

# Domoticz Home Automation Workbook

---

*Home Automation & Information System*

**Robert W.B. Linn**

**15.12.2020**

## **DISCLAIMER**

THIS DOCUMENT IS PROVIDED BY THE AUTHOR “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# Table of Contents

Table of Contents .....	1
Purpose .....	15
Objectives .....	15
Remarks .....	15
Functions .....	15
Explore .....	16
Credits .....	17
Abbreviations .....	17
Components .....	18
Hardware .....	18
Domoticz Production System .....	18
Domoticz Development System .....	19
Software .....	19
External Services .....	19
Setup .....	20
Development Device .....	20
Raspberry Pi .....	20
Installation .....	20
Network Static IP Address .....	21
Wired Network .....	22
WLAN Power Save Mode .....	23
Raspian Check Version & Update .....	24
Samba .....	25
Persistent USB Devices .....	29
Domoticz .....	30
Installation .....	30
Settings .....	31
Folders .....	32
Events .....	32
Functions .....	33
Introduction .....	33
Air Pressure (BME280) .....	34
Purpose .....	34
Wiring .....	34
Circuit .....	34

Domoticz Configuration.....	36
Barometer Widget .....	37
Automation Script .....	37
Alert Message.....	39
Purpose .....	39
Solution .....	39
Automation Script .....	40
Alert Message Reset.....	41
Database Tables.....	41
Ambient Light (BH1750).....	43
Purpose .....	43
Parts List.....	43
Wiring.....	43
Circuit.....	43
Device Setup Domoticz.....	43
Device Setup ESP Easy.....	45
Trigger Switch Light .....	46
Android App atHome .....	48
Purpose .....	48
Solutions.....	48
Anemometer (TFA 30.3168).....	50
Purpose .....	50
Device.....	50
Setup.....	50
Battery Status .....	51
Purpose .....	51
Solution .....	51
Domoticz .....	52
RaspberryMatic .....	54
Blinds (Somfy RTS Pure) .....	58
Purpose .....	58
Domoticz Configuration.....	58
Dashboard & Switches Widget .....	59
MQTT Messages .....	59
BME280 Temp+Hum+Baro.....	60
Purpose .....	60
Solution .....	60

Wiring.....	60
Circuit.....	60
Domoticz Configuration.....	63
Automation Script .....	65
Climate Environment (DWD Service) .....	67
Purpose .....	67
Solution .....	67
UV Index .....	67
Pollen.....	71
Modified Solution .....	75
Coffee Machine Monitor.....	76
Purpose .....	76
Solution .....	76
Automation Script .....	77
Contact Detection (433 Mhz).....	78
Purpose .....	78
Solution Sensor PB-62R.....	78
Domoticz Configuration.....	79
Alarm Detection .....	79
Automation Script .....	80
Alarm State Reset Switch.....	81
MQTT.....	82
Custom Settings.....	84
Purpose .....	84
Solution .....	84
Installation .....	85
HTML File .....	86
Enhancements.....	90
Days-To-Go.....	91
Purpose .....	91
Solution .....	91
Atomation Script .....	92
Device Status.....	94
Purpose .....	94
Solution .....	94
Installation .....	95
HTML File .....	96

Electric Usage .....	99
Purpose .....	99
Solution .....	99
Electric Usage House (volkszaehler) .....	100
Purpose .....	100
Solution .....	100
Domoticz Configuration.....	101
Polling .....	102
Automation Script .....	104
MQTT.....	105
HTTP API Request.....	105
Log Examples.....	107
Database Table .....	107
volkszaehler.....	109
Electric Usage Rooms/Devices .....	124
Purpose .....	124
Solution Homematic IP .....	124
Solution Revolt SF-436.m.....	126
Email Control .....	131
Purpose .....	131
Solution .....	132
Event Monitor .....	139
Purpose .....	139
Domoticz Configuration.....	139
Automation Script .....	140
Maintenance .....	141
Garage Door Monitor (HMIP-SWDM).....	142
Purpose .....	142
Solution .....	142
Homematic Script.....	143
Automation Script .....	144
Hardware Monitor Raspberry Pi .....	145
Purpose .....	145
Solution .....	146
RPi volkszaehler.....	147
RPi Domoticz Production.....	150
RPi Status Indicator.....	150

RPi LCD Hardware Info .....	152
Indoor Air Quality (Tinkerforge, Script).....	153
Purpose .....	153
Solution .....	153
Tinkerforge Setup.....	155
Domoticz Configuration.....	157
Pseudo Code.....	159
Python Script.....	160
Automation Script .....	163
Enhancements.....	167
Indoor Air Quality (Tinkerforge, Plugin).....	168
Purpose .....	168
Solution .....	168
Info Message .....	170
Purpose .....	170
Solution .....	170
Automation Script .....	170
Key Dates (JSON).....	171
Purpose .....	171
Solution .....	171
Domoticz Configuration.....	172
Automation Script .....	173
Roomplan .....	175
Key Dates (CSV).....	177
Purpose .....	177
Domoticz Configuration.....	177
Concept.....	178
CSV Input File .....	178
Automation Script .....	179
Python CSV File Generation Script .....	180
Music Player (Kodi) .....	184
Purpose .....	184
Solution .....	184
Installation .....	185
Overview Devices & User Variables .....	186
Roomplan and Floorplan .....	187
Remote Control Functions .....	188

Automation Script .....	189
Notes.....	190
Purpose .....	190
Solution .....	190
Domoticz Configuration.....	193
Export Notes.....	194
Database Records .....	196
Philips Hue .....	198
Purpose .....	198
Hue Bridge Configuration.....	198
Hue Devices .....	198
Hue Light Add New .....	199
Hue Control Tests.....	200
Hue Timed Switch Lights .....	202
Hue Lights Control Node-RED.....	204
Hue Direct Access .....	206
Hue Color Setting.....	214
Postbox Notifier (HmIP-SWDO).....	218
Purpose .....	218
Solutions.....	218
Solution Alert .....	220
Solution Alert + Switch.....	223
Solution Alert + Switch + Voltage.....	227
Radiator Thermostat (HmIP-eTRV).....	232
Purpose .....	232
Solution Thermostat SetPoint Device .....	233
Solution Selector Switch & Thermostat.....	241
Solution Plugin .....	246
Raspberry Pi Monitor .....	253
Purpose .....	253
Solution .....	253
Monitoring.....	254
RaspberryMatic Duty-Cycle Monitor .....	256
Purpose .....	256
Solution .....	257
HTTP XML-API Requests.....	257
Domoticz Configuration.....	258

Automation Script .....	259
Remote Control (HmIP-RC8) .....	260
Purpose .....	260
Solution .....	260
RaspberryMatic Configuration.....	262
Solution Custom Event.....	264
Solution Selector Switch .....	265
River Elbe Tide (WSV Service).....	268
Purpose .....	268
Solution .....	268
Domoticz Configuration.....	269
Automation Script .....	269
Outlook.....	270
Roomplans.....	271
Purpose .....	271
Define Roomplan .....	271
Define Device Names.....	272
Overview Roomplans .....	273
RFXCOM RFXtrx433E .....	275
Purpose .....	275
Prepare RFXtrx433E .....	275
Domoticz Configuration.....	277
Troubleshooting .....	278
Packet Analysis .....	279
Soil Moisture Monitor (Tinkerforge, Plugin).....	280
Purpose .....	280
Solution .....	280
Stock Quotes (Service, Alpha Vantage).....	281
Purpose .....	281
Domoticz Configuration.....	282
Solution Automation Script.....	283
Solution Node-RED .....	285
Notifications .....	289
Enhancements.....	291
Hints .....	292
Temperature & Humidity (TFA TS34C) .....	293
Purpose .....	293

Device.....	293
Setup.....	293
Replace Batteries.....	294
Time Control .....	295
Purpose .....	295
Solution .....	296
Domoticz Configuration.....	296
Various .....	300
Timers.....	301
Purpose .....	301
Rule .....	301
Device Timers .....	301
Timerplans.....	304
Automation Events .....	305
Utility Dashboard (Grafana, Flowcharting) .....	308
Purpose .....	308
Dashboard.....	308
Solution .....	310
Design Approach.....	311
Process Flow Diagram .....	312
Domoticz Configuration.....	318
Grafana .....	323
Domoticz Custom Page.....	330
Automation Scripts.....	334
Domoticz Roomplan .....	335
Weather Forecast (Service, Meteorologisk) .....	337
Purpose .....	337
Solution .....	337
Domoticz Configuration.....	339
Custom Page.....	341
Installation .....	342
Enhancements.....	344
Web App Site Control .....	346
Purpose .....	346
Web UI Quick Access Mobile .....	348
Purpose .....	348
Solution .....	348

Get Started.....	353
Web Radio (Volumio).....	355
Purpose .....	355
Volumio Setup.....	356
Solution Options.....	356
Solution Automation Script.....	358
Solution Node-RED .....	362
Favorites List.....	367
Wind (TFA 30.3168).....	368
Purpose .....	368
Solution .....	368
Automation Script .....	369
User Variables .....	370
Syntax.....	370
List.....	370
SQL .....	370
Python.....	372
Usage .....	373
Scripts.....	373
Browser Interactive .....	374
Node-RED .....	375
Node-RED Dashboard.....	376
Overview API Requests .....	377
Custom Page.....	378
Purpose .....	378
Solution .....	378
HTML File User_Variables .....	380
Custom Icons.....	384
Purpose .....	384
Create Icons.....	384
Icon List File .....	384
Icon Archive .....	384
Icon Import .....	385
Icon Import (2 <sup>nd</sup> example) .....	386
Additional Information.....	387
Get List of Icons .....	387
Location Custom Images.....	387

Explore .....	389
API Interaction .....	389
Purpose .....	389
Option HTTP .....	389
Option MQTT .....	391
Recommendation.....	393
Example Syntax HTTP vs MQTT .....	393
Custom Pages .....	395
Purpose .....	395
Prepare.....	396
Alert Message (Form) .....	409
Quick Message (Bootbox).....	411
Raspberry Pi CPU Dashboard (JustGage & Bootstrap Grid).....	413
Raspberry Pi Monitor (Badges).....	416
Raspberry Pi Quick Check (Bootbox).....	418
Switch Hue Light (Form) .....	420
Custom Pages Charts.....	422
Purpose .....	422
Get Started.....	424
Sources .....	428
Concepts .....	429
Custom Page.....	431
Test Scenario .....	436
Highcharts .....	438
Google Charts.....	441
Installation .....	446
Data Sources.....	447
Custom UI .....	450
Purpose .....	450
Approach .....	450
Domoticz Configuration.....	451
Custom UI HTML File.....	452
Appendix: Solution Options Libraries .....	458
Appendix: Automation Event Simulator.....	459
Appendix: Older Solution .....	460
ESP8266 .....	465
Purpose .....	465

Experiment .....	465
Prototype .....	465
Prepare Arduino IDE.....	465
Parts List.....	466
Wiring.....	466
Circuit.....	466
Source Code .....	467
ESP Easy .....	468
Purpose .....	468
Overview Experiments.....	468
Flash ESP .....	469
Update ESP .....	470
Domoticz Controller.....	471
ESP Easy Rules .....	472
GPIO Commands .....	474
ESP Push-Button switching ESP LED.....	475
Domoticz Switch ESP LED .....	477
ESP Ambient Light to Domoticz .....	479
ESP controlling Hue Brightness.....	483
ESP BMP280 to Domoticz.....	488
Events .....	491
Purpose .....	491
Event Execution Order .....	491
Event Scheduling & Trigger .....	492
Event Database Tables .....	492
Event Script Viewer .....	492
Event Development .....	493
Example Basement Humidity Monitor .....	494
Events dzVents Lua.....	503
Device Properties .....	503
Script .....	503
Property List.....	504
External Modules .....	504
Shared Helper Functions .....	508
Fast Event Less One Minute .....	509
Asynchronous HTTP requests.....	523
Grafana & InfluxDB.....	524

Purpose .....	524
Test Scenario .....	525
InfluxDB .....	525
Grafana Installation .....	529
Domoticz Configuration .....	531
Line Chart Single Device .....	537
Line Chart Multiple Devices .....	538
Gauge Chart Multiple Devices .....	539
Gauge & Line Chart .....	540
GPIO .....	541
Purpose .....	541
Notes .....	542
RPi.GPIO .....	545
GPIO Zero .....	566
I2C LCD Display .....	594
Node-RED .....	607
Kodi Music Player .....	608
Purpose .....	608
Parts List .....	608
Installation .....	608
Kodi JSON-RPC API Communication .....	610
Domoticz Automation Events .....	611
Domoticz Explore .....	614
Appendix: Kodi JSONRPC Command (Selection) .....	626
Appendix: Various Hints .....	629
MQTT .....	632
Purpose .....	632
Install .....	632
Mosquitto .....	634
Python .....	636
Node-RED .....	641
Purpose .....	641
Install & Update .....	641
Access Flows & Dashboard UI .....	643
Start, Stop, Log .....	643
Manage Node Packages .....	643
Folder Locations .....	644

Example Flows .....	645
Example Dashboard UI Flows .....	651
Plugin Development.....	656
Purpose .....	656
Hints .....	658
Plugin Traffic Light (Tinkerforge Building Blocks) .....	662
RaspberryMatic .....	669
Purpose .....	669
Solution .....	669
Installation.....	669
Addon XML-API .....	670
Addon CUxD.....	689
Plugin Considerations .....	694
CCU Device(s) Information .....	695
CCU Devices HomematicIP.....	697
Experiments.....	699
SQL .....	715
Purpose .....	715
SQLite Shell .....	716
Python Scripts .....	731
Tinkerforge .....	737
Purpose .....	737
Hardware .....	737
Software .....	737
Concepts .....	738
Appendix Domoticz Hints.....	754
Start, Stop, Status .....	754
Lost Username and Password .....	754
Create Backup .....	755
Change Log Level .....	755
Troubleshooting .....	756
Device Widget turns yellow/red.....	756
Check Domoticz Status .....	757
GLIBC Version not found.....	757
Script Python Code 32512.....	758
HTTP API Requests .....	759
Rename Device .....	759

Various .....	760
CLI Commands .....	760
String Length.....	761
Hide Devices .....	761
Appendix Domoticz Build Source .....	762
Appendix volkszaehler Setup.....	764
Hardware .....	764
Software .....	764
Setup Raspberry Pi .....	765
Option 1 WLAN static IP address.....	765
Option 2 Ethernet static IP address .....	765
volkszaehler.....	767
PowerMeter Setup .....	767
USB Configuration.....	768
Define Channels .....	769
vzlogger Setup.....	770
Appendix Tools .....	777
Domoticz Internal Script Viewer .....	777
Purpose .....	777
Domoticz MQTT Logger .....	778
Purpose .....	778
Solution .....	778
Alternative Logger .....	781
RaspberryMatic Statelist .....	783
Purpose .....	783
Solution .....	783
Node-RED .....	783

# Purpose

To build a Home Automation & Information System, based on the [Domoticz](#) Home Automation System, running on the single-board computer Raspberry Pi.

## Objectives

- ✓ Explore & learn Domoticz & Scripting.
- ✓ Build a Home Automation & Information System.
- ✓ Write up & share experiences during development.
- ✓ Use this workbook as a supplemental reference.
- ✓ Having fun writing down experiences & share.

## Remarks

- ❖ This is a working document
  - ... concept changes, new idea's & to-do's whilst progressing.
- ❖ There might be better solutions for what is shared
  - ... updates or changes depending learning curve.
- ❖ Source code for scripts, flows & apps not fully shared or have changed
  - ... check the [GitHub](#) "src" folder for the latest sources.
- ❖ Automation events created as dzVents (Domoticz Easy Events) Lua event scripts.
- ❖ Domoticz Hardware Plugin development in Python.
- ❖ Custom pages with JavaScript.
- ❖ Some of the functions make initially use of Node-RED flows
  - ... but target is to replace by dzVents Lua scripts if possible.
- ❖ Functions that have been replaced by another solution are kept in this document as reference – could be of use.
- ❖ Screenshots might not be updated as functionality is evolving.
- ❖ To-do's are tagged with <TODO>.The To-do list, with prefix, i.e. NEW, UPD ..., are captured in the file TODO.md.
- ❖ Device names and other are in cases in German as the Domoticz system is used in Germany.

## Functions

- Air Quality monitor [[Plugin](#)].
- Ambient light (from ESP8266 running ESP Easy) with threshold.
- Battery Status daily check & notify for all devices (Domoticz & External Systems) the battery level against a threshold set via user variable.
- Charts for selective weather items, room temperature & humidity.
- Climate Environment UV Index & Category, Pollen Index from DWD.
- Coffee machine monitor start and end time, info message.
- Control Somfy roller shutters with RTS motors in rooms.
- Control Volumio music player whilst listening to web radio.
- Custom icons.

- Custom Pages simple to more complex examples.
- Custom Settings custom page to change the value of selective user variables.
- Device Status custom page to instantly view all device properties of a used device.
- Display temperature & humidity measured in rooms.
- Electricity power & energy for selected devices / rooms (washer, MakeLab etc.).
- Electricity power & energy for the house from “volkszaehler” with charts.
- Event monitor for selected devices.
- Garage door open | close state monitor.
- Hardware Monitor for the Raspberry Pi's used (see HardwareMonitor.zip).
- Homematic (RaspberryMatic) CCU3 integration with variety of devices and plugins [[HmIP-eTRV](#)], [[HmIP-PSM](#)].
- Indoor Air Quality measure the IAQ Index VOC, assign to a custom sensor (VOC) and set the color of a Hue White & Color Ambiance bulb accordingly.
- Key Dates management and information organised in a roomplan.
- MQTT Logger.
- MQTT subscribe & publish messages to trigger actions or information.
- Monitor stock quotes.
- Music Player LibreELEC with Kodi.
- Notes with a subject & content and save, copy, or notify.
- Philips Hue Lighting System control via Hue Bridge for ZigBee devices.
- Philips Hue Light controlled by ESP8266 with slide potentiometer & 4-digit-7-segment-display.
- Postbox Notifier.
- RaspberryMatic DutyCycle monitor.
- RaspberryMatic Statelist with devices and datapoints.
- Raspberry Pi system information with charts and threshold email notification.
- Remote Control via Homematic IP device (HmIP-RC8) for 8 specific functions.
- Remote Control Domoticz devices using Email client – incl. simple Android test app.
- Security door & window wireless contact detectors.
- Soil Moisture monitor for plants [[Plugin](#)].
- Time Control to track & control the time spent in hours, per activity block (with start- & end-time) and total/day, on a generic task.
- Timers for single devices or complex tasks (Automation events dzVents script).
- Weather 9-DAYS forecast from Norwegian Meteorological Institute (Meteorologisk) as a Custom Page with Highcharts Custom Chart.
- Web UI Quick Access Mobile [[Info](#)] customized web frontend to control dedicated functions mainly accessed from mobile smartphones devices.
- Web App Site Control [[Info](#)] Node-RED alternative to the standard Domoticz Web UI (Not developed further as replaced by Web UI Quick Access Mobile).

## Explore

- Domoticz running on a Raspberry Pi (setup, configure).
- Scripting dzVents (Automation), JavaScript (Custom Pages), Lua, Python (Plugin).
- Domoticz RESTful API generating JSON returns from HTTP requests.
- Domoticz Python Framework (mainly with Tinkerforge Building Blocks).
- MQTT messaging.
- Custom Pages with Highcharts & Google Charts for single & multiple devices.

- Node-RED as an alternative script engine and User Interface.
- SQLite3 to read the Domoticz Database.
- ESP8266, ESP Easy.
- Raspberry Pi GPIO libraries RPi.GPIO and GPIOZero.
- RFXCOM RFXtrx433E USB RF Transceiver for
  - Temperature & Humidity devices.
  - External Wind device (only for RFXCOM tests).
  - Other 433Mhz devices, i.e. door & window contacts.
- Philips Hue Light Control.
- HomematicIP using RaspberryMatic CCU and integrate into Domoticz.
- External services.
- Domoticz Android App (native client, to be determined).
- Advanced User Interfaces, i.e. Node-RED, Bootstrap ...

## Credits

To the developers of Domoticz and to all sharing information about Domoticz. Without these, it would not be possible to build this project and write the workbook.

Drawings are created with [Fritzing](#).

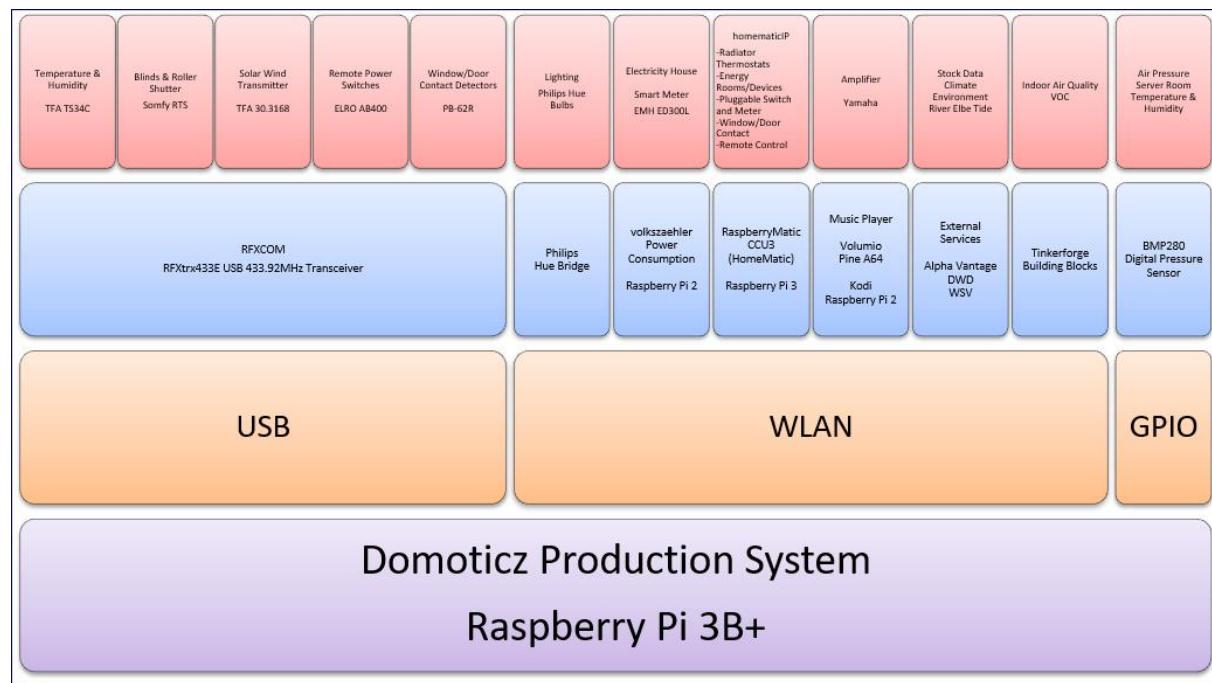
## Abbreviations

Some of the abbreviations used.

<b>CLI</b>	Command Line Interface
<b>domoticz-ip</b>	Domoticz system IP address
<b>~domoticz</b>	Path to the Domoticz folder (Home Directory)
<b>dzVents</b>	Automation events using dzVents Lua scripts
<b>GUI</b>	Domoticz UI in browser
<b>HmIP</b>	HomematicIP
<b>RaspMatic</b>	RaspberryMatic
<b>RPi</b>	Raspberry Pi
<b>TF</b>	Tinkerforge

# Components

Hardware and software used for this project.  
More details to be found in chapters [Setup](#), [Functions](#) and other.



## Hardware

### Domoticz Production System

- Raspberry Pi 3B+ v1.2 (Domoticz Production System)
- RFXCOM RFXtrx433E USB 433.92MHz Transceiver
  - TFA Dostmann TS34C (temperature & humidity)
  - TFA Dostmann 30.3168 Wind Meter (speed, direction, temperature)
  - Somfy Blinds RTS Pure (blinds & roller shutter)
  - Revolt SF-436 (electricity rooms/devices – tested only = not used)
  - PB-62R (door & window wireless contact detectors)
  - ELRO AB400 (remote power switches)
- Philips Hue Bridge
  - Hue Bulbs
- Smart Meter EMH ED300L connected to a Raspberry Pi 2 running volkszaehler
- RaspberryMatic CCU (Raspberry Pi 3B+) with variety of connected devices like HmIP-PSM, HmIP-eTRV-B, HmIP-eTRV-2, HmIP-SDWO, HmIP-SDWM
- Pine A64 1GB - Volumio Music Player
- Raspberry Pi GPIO - BMP280 Barometer (I2C)
- Raspberry Pi 2B - Kodi Music Player

## Domoticz Development System

- Raspberry Pi 4B for testing & plugin development
  - Domoticz latest BETA version to ensure up-to-date with development
  - Interfacing, i.e. GPIO, RaspberryMatic; Automation events dzVents scripts
  - Node-RED, Tinkerforge and more

## Software

- Domoticz Production System:
  - Raspberry Pi OS (32-bit) (Debian Buster, Linux dodev 4.19.118-v7+)
  - Domoticz 2020.2 Stable
- Domoticz Development System:
  - Raspberry Pi OS (32-bit) (Debian Buster, Linux dodev 4.19.118-v7+)
  - Domoticz latest BETA version (i.e. 2020.2 build 12252)
- RFXCOM RFXflash Programmer 8.0.0.0 - to update the firmware on the RFXtrx433E
- RFXCOM RFXmngr 18.0.0.18 - to test and manage RFXtrx433E connected devices
- RaspberryMatic 3.51.6.20200621 (full compatible Homematic CCU3 firmware)
- Node-RED v1.0.6 – various functions
- WinSCP - to manage & exchange files between development devices and the Raspberry Pi's
- PuTTY - to run terminal commands
- SSH - to remote connect

*Note*

The software versions are subject to change. Recommend checking out for newer versions available – newer versions are mentioned in the related chapters.

## External Services

External Services are used to request information

- Alpha Vantage (Stock Data)
- DWD (Climate Environment)
- WSV (River Elbe Tide)

# Setup

Various setup steps prior starting to build the solution in Domoticz.

## Development Device

A **Notebook**, running **Windows 10**, is used to support the setup and as development device.  
Additional software installed:

- [RealVNC](#) - Remote access Raspberry Pi Desktop
- [Python](#) - Script development and testing
- [Notepad++](#) - Text and source code editor

## Raspberry Pi

### Installation

The Raspberry Pi running as the Domoticz system, has a monitor, keyboard and mouse connected. This to ease the setup.

After initial installation, the Raspberry Pi is running headless and VNC is used to remote access the Raspberry Pi desktop.

#### *NOOBS*

Download the Raspberry Pi NOOBS version from [here](#).

NOOBS Version 2.8.2, 2018-06-27, Zip archive NOOBS\_v2\_8\_2.zip.

#### On the Development PC

- insert a new SD card (used 32GB)
- format the SD card (used SDFormatter)
- unzip NOOBS\_v2\_8\_2.zip to the SD card

#### On the Raspberry Pi

- insert the SD card (ensure no power connected)
- connect power to the Raspberry Pi
- from the initial installation menu
  - select Raspian
  - select expand partition (extra 512 MB)
  - set WiFi connection
  - press install
  - installation procedure starting, wait for completion (~10 minutes)
  - reboot
  - complete the Assistant steps incl. Check for Updates (~20 minutes)

Additional configuration Desktop Menu Preferences > Raspberry Pi Configuration >  
Interfaces enabled SSH, VNC, SPI, I2C, 1-Wire

Any other software for the project [Functions](#) are described under Functions.

## Network Static IP Address

It is useful to define a static IP address for the Raspberry Pi running Domoticz, i.e. fix IP address to access Domoticz from a browser (<http://domoticz-ip:8080>) and for a Samba shared folder used i.e. to edit Scripts from the Development PC or backup or read the Domoticz database.

For release Raspbian GNU/Linux 8 (stretch), set a fixed IP address using the Desktop. This is done by right clicking on the LAN icon at the top right of the desktop and select menu:

### Wireless & Wired Network Settings.

```
Wireless
Select Menu Configure > Interface > wlan0:
set IPv4: NNN.NNN.N.NNN, Router: NNN.NNN.N.NNN

Wired
Select Menu Configure > Interface > eth0:
set IPv4: NNN.NNN.N.NNN, Router: NNN.NNN.N.NNN

Apply and Close.
```

#### *Note*

Just as an FYI, the file `/etc/dhcpcd.conf` is updated by the Raspberry Pi Desktop application.

```
sudo nano /etc/dhcpcd.conf
interface wlan0
inform NNN.NNN.N.NNN
static routers=NNN.NNN.1.1

interface eth0
inform NNN.NNN.N.NNN
static routers=NNN.NNN.1.1
static domain_name_servers=NNN.NNN.1.1
```

### WLAN Router SSID

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=DE

network={
    ssid="*****"
    psk="*****"
    key_mgmt=WPA-PSK
}
```

## Reboot the Raspberry Pi

Check, using command ifconfig, the WLAN address (wlan0) or Ethernet address (eth0)

```
ifconfig
Output:
wlan0      Link encap:Ethernet  HWaddr b8:27:eb:fa:44:fc
           inet addr:NNN.NNN.N.NN  Bcast:NNN.NNN.0.255  Mask:255.255.255.0
...
eth0:  flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet NNN.NNN.N.NNN  netmask 255.255.255.0  broadcast NNN.NNN.1.255
...
```

### Note

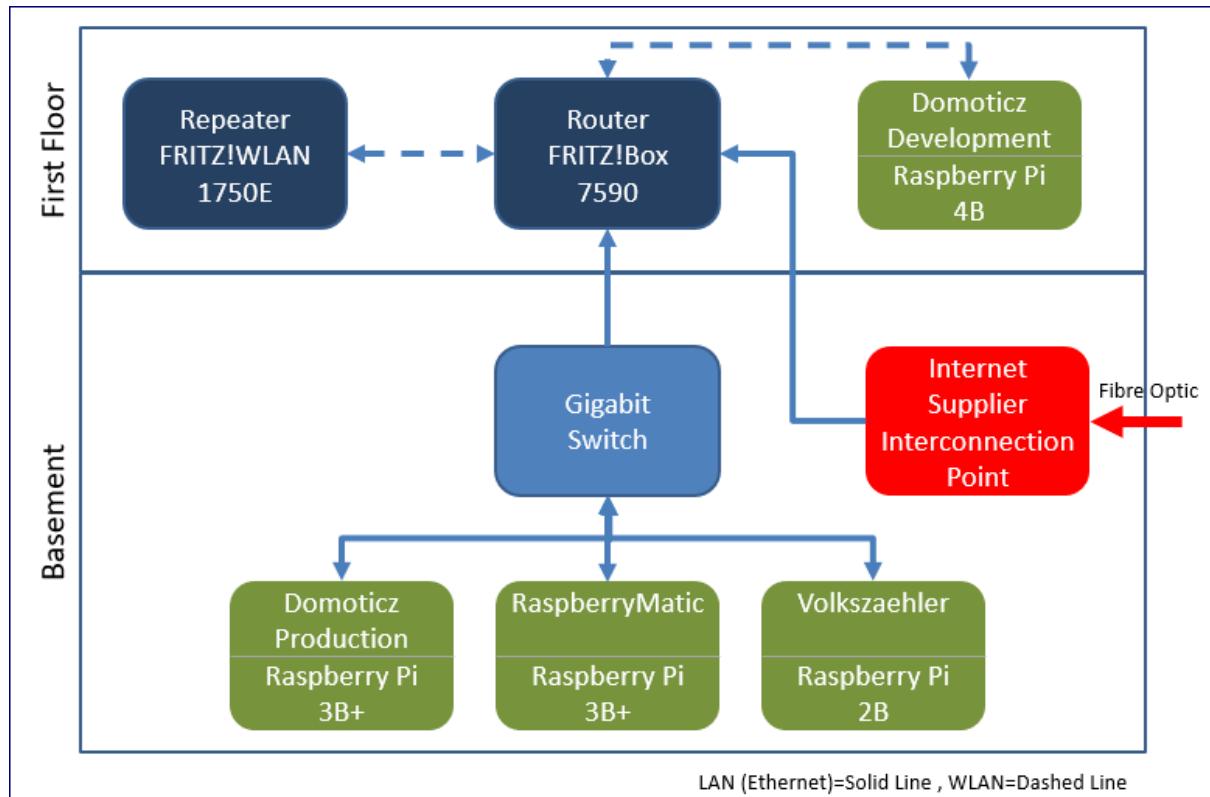
In the documentation the IP address of the Raspberry Pi Domoticz system (production and development) is referred to as: <http://domoticz-ip:8080>

## Wired Network

Whilst this Home Automation solution is evolving, decided to use wired connections for the several production Raspberry Pi's in the network.

The reason is that the hardware is placed in the basement near the power & interconnection distribution = need to ensure highest available network speed.

A FRITZ!WLAN Repeater 1750E is also installed to provide WLAN in the far end of the backyard.



## WLAN Power Save Mode

If the Domoticz GUI is not refreshing, the cause could be that the Raspberry Pi WLAN Power Save Mode is ON. After certain time, the network connection drops.

### Check the Raspberry Pi WLAN power save mode

```
iw wlan0 get power_save
```

Output

```
Power save: on
```

*Note*

The command *iwconfig* provides this information also, i.e. shows Power Management: off

To turn the Raspberry Pi WLAN “Power Save Mode” OFF, run

```
sudo iw wlan0 set power_save off
```

*Note*

Run as sudo else error message: command failed: Operation not permitted (-1)

Check again the Power Save Mode:

```
iwconfig or iw wlan0 get power_save
```

The “Power Save Mode” is set back to default ON when the Raspberry Pi reboots.

To turn the Raspberry Pi WLAN “Power Save Mode” OFF during reboot, add to crontab

```
sudo crontab -e  
# Disable wlan power save  
@reboot sudo iw wlan0 set power_save off
```

Reboot the Raspberry Pi and check power save mode

```
iw wlan0 get power_save  
Power save: off
```

# Raspian Check Version & Update

## Raspian Check version installed

OS Release Notes	Debian Version	Kernel Version
cat /etc/os-release	cat /etc/debian_version	uname -a
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)" NAME="Raspbian GNU/Linux" VERSION_ID="10" VERSION="10 (buster)" VERSION_CODENAME=buster ID=raspbian ID_LIKE=debian HOME_URL="http://www.raspbian.org/" SUPPORT_URL="http://www.raspbian.org/RaspbianForums" BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"	10.4	Linux dodev 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l GNU/Linux

## Raspian Update to the latest version

Recommend performing regular updates, i.e. once a week.

```
Desktop Select Menu > Preferences > Add / Remove Software > Options > Check for Updates
```

Or run alternative from a terminal:

```
sudo apt-get update && sudo apt-get upgrade
```

### Notes

If packages are kept back, re-install, like

```
sudo apt-get install --reinstall raspberrypi-ui-mods
```

Regular clean and purge:

```
sudo apt clean  
sudo apt purge
```

## Samba

Start by reading [here](#).

### Hint

Do not forget to set the password and restart Samba.

```
sudo smbpasswd -a pi
sudo systemctl restart smbd.service
```

## Domoticz Home Folder as a share

The Domoticz home folder /home/pi/domoticz can also be set as a shared folder.

### Prework Raspberry Pi Domoticz System

The shared folder, DoProDomoticz, uses samba and is configured with:

```
sudo nano /etc/samba/smb.conf
```

### Content

```
[DoProDomoticz]
Comment = Raspberry Pi Domoticz Production folder
Path = /home/pi/domoticz
Browseable = yes
Writable = Yes
only guest = no
create mask = 0777
directory mask = 0777
Public = yes
Guest ok = yes
```

### *Details Samba settings:*

[DoProDomoticz]	Name of the share
comment	Text is displayed as Comments in the Share detail view
path	Specifies the folder with the shared files
browsable	yes no - Set share visible when running <i>net view</i> command and browsing network shares.
writable	yes no - Allows user to add/modify files and folders. Default samba shares are readonly
guest ok	yes no - Allows “non authenticated” users to access the share

**After configuration:** Perform following steps after the Samba configuration.

Ensure to set the directory mask for the domoticz folder to 0777:

```
sudo chmod 777 /home/pi/domoticz
```

Restart samba:

```
sudo systemctl restart smbd.service
```

Restart domoticz:

```
sudo service domoticz.sh restart
```

## Prework Windows

In Windows (Version 10 is used) connect to the shared folder using the Windows Explorer > Connect Network:

Path for drive Z: is \\ domoticz-ip\DoProDomoticz  
(DoProDomoticz is previous defined in the samba configuration)

To connect from the Windows explorer:

Network drives:

Add new drive with URL: \\domoticz-ip\DoProDomoticz

Username: pi, Password: \*\*\*\*\*

## Example Music Shared Folder

Another example in setting up a music shared folder “MyMusic” pointing to folder /home/pi/music.

Create the folder music:

```
mkdir /home/pi/music
```

Add to the Samba configuration file, /etc/samba/smb.conf

```
[MyMusic]
comment=Raspberry Pi Music Share
path=/home/pi/music
browseable=Yes
writeable=Yes
only guest=no
create mask=0777
directory mask=0777
public=yes
```

Ensure to set the directory mask for the folder to 0777:

```
sudo chmod 777 /home/pi/music
```

Restart Samba:

```
sudo /etc/init.d/samba restart
```

Restart Domoticz:

```
sudo service domoticz.sh restart
```

Connect from Windows Explorer using:

```
URL: \\domoticz-ip\MyMusic
Username: Pi
Password: *****
```

## Example Samba Installation with Share

Full dump terminal commands used to install on a Raspberry Pi (running Raspian Debian Linux Buster) Samba and set the share /home/pi (to access all folders and files of home).

```
# Ensure to update & upgrade (optional) Raspbian
sudo apt-get update && sudo apt-get upgrade
# Install Samba
sudo apt-get install samba samba-common-bin
# Check if Samba is running with output like "active(running) ...
sudo service smbd status
sudo service nmbd status
# Move the org smb config or make the changes in this large file
pi@dodev:~ $ sudo mv /etc/samba/smb.conf /etc/samba/smb.conforg
# Create a new smb config file - if not using the org file
sudo nano /etc/samba/smb.conf
With content:
[global]
workgroup = WORKGROUP
security = user
encrypt passwords = yes
client min protocol = SMB2
client max protocol = SMB3
# Test the config file & restart Samba - ok message: Loaded services file OK.
testparm
sudo service smbd restart
sudo service nmbd restart
# Define the shared folder - using /home/pi
sudo nano /etc/samba/smb.conf
With content:
[DoProDomoticz]
Comment = Raspberry Pi Domoticz Production folder
Path = /home/pi
Browseable = yes
Writable = Yes
only guest = no
create mask = 0777
directory mask = 0777
Public = yes
Guest ok = yes
# Test the config file & restart Samba - ok message: Loaded services file OK.
testparm
sudo service smbd restart
sudo service nmbd restart
# Change the Samba password
sudo smbpasswd -a pi
sudo service smbd restart
sudo service nmbd restart
sudo chmod 777 /home/pi
sudo service smbd restart
sudo service nmbd restart
# Connect from a device to the shred folder, i.e. Windows 10
Drive X: = \\domoticz-ip\DoProDomoticz
```

### *Note*

Related to the [global] configuration in file /etc/samba/smb.conf

This is a basic global configuration as set by the section [global]. All settings apply to the defined shares.

- Key "workgroup" has been used in the past in Windows networks to structure groups. It is not really used anymore and left for compatibility purposes.
- Key "security" set the security level when accessing the shares. The value "user" sets the user/management of the server, means for accessing the Samba shares the server user rights are used.
- Key "encrypt passwords" should be "yes" to ensure passwords are encrypted during transfer.

Both "security" and "encrypt passwords" ensure that in case wrong userdata is used, the login is as "guest".

- Keys client min protocol = SMB2 and client max protocol = SMB3.  
For Windows 10, the protocol SMB1 is not secure and disabled, therefor two new parameter are required with the lowest & highest smb protocol version.

A second share has been defined to access the Domoticz Development System.

```
[DoDevDomoticz]
Comment = Raspberry Pi Domoticz Development folder
Path = /home/pi
Browseable = yes
Writeable = Yes
only guest = no
create mask = 0777
directory mask = 0777
Public = yes
Guest ok = yes
```

Y: = \\domoticz-ip\DoDevDomoticz

Example of y:\domoticz folder on the Domoticz Development System:

y [dodevdomoticz] 19.948.808 k of 27.784.728 k free			
y:\domoticz\*.*			
Name	Ext	Size	Date
..		<DIR>	02/02/2020 10:07
[Config]		<DIR>	30/01/2020 12:53
[d2vents]		<DIR>	30/01/2020 12:53
[plugins]		<DIR>	30/01/2020 12:53
[scripts]		<DIR>	30/01/2020 12:53
[www]		<DIR>	30/01/2020 12:53
domoticz	db-wal	255.472	03/02/2020 10:09
domoticz	db-shm	32.768	03/02/2020 10:08
domoticz	db	937.984	03/02/2020 10:05
domocookie	txt	370	03/02/2020 05:00
domoticz_crash	log	4.243	02/02/2020 18:39
domoticz		13.712.512	02/02/2020 12:23
updatebeta		610	30/01/2020 12:53
updaterelease		627	30/01/2020 12:53
domoticz	sh	4.433	30/01/2020 12:53
History	txt	96.326	30/01/2020 12:53
License	txt	35.147	30/01/2020 12:53
server_cert	pem	3.414	30/01/2020 12:53

## Persistent USB Devices

On the Raspberry Pi, the RFXtrx433E device is connected to an USB port. If multiple USB devices connected, the device order could change after reboot, resulting in Domoticz using the wrong devices. To ensure the right USB Port is used by Domoticz, a symbolic link (**symlink**) is assigned for the USB Port associated to the RFXtrx433E.

### USB Symlink for RFXCOM RFXtrx344E

#### Create symlink:

1. Do not plug in the RFXtrx433E.
2. List the devices: `ls /dev/tty*`
3. Plug in the RFXtrx433E to an USB port.
4. List the devices again to check out the new USB device: `ls /dev/tty*`
5. A new device should be listed, i.e: **/dev/ttyUSB0**

To define the symlink, USB device information is required: *idVendor*, *idProduct*, *iSerial*.

#### Get USB device information

```
sudo lsusb -v | grep 'idVendor\|idProduct\|iProduct\|iSerial'
```

#### Seek the RFXCOM RFXtrx433 entry listed under iProduct, i.e. RFXtrx433

```
Bus 001 Device 008: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART)
IC
Device Descriptor:
...
idVendor      0x0403 Future Technology Devices International, Ltd
idProduct     0x6001 FT232 USB-Serial (UART) IC
iProduct       2 RFXtrx433
iSerial        3 A1YQCEQY
...
```

The data required to define the symlink are the idVendor (0x0403), idProduct (0x6001), iSerial (A1YQCEQY).

#### Create the symlink

On the Raspberry Pi, edit the file `/etc/udev/rules.d/99-usb-serial.rules` and reboot!

```
sudo nano /etc/udev/rules.d/99-usb-serial.rules
Add:
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", ATTRS{serial}=="A1YQCEQY",
SYMLINK+="ttyUSB-RFX433E-A", MODE="0666"
```

**Reboot** the Raspberry Pi (`sudo shutdown -r now`) and **check** the USB Port after boot, check the USB port

```
ls -l /dev/ttyUSB-RFX433E-A
Output:
lrwxrwxrwx 1 root root 7 Aug 29 19:24 /dev/ttyUSB-RFX433E-A -> ttyUSB0
```

Preparation of the RFXtrx433E is completed, next to install & configure the RFXCOM device in Domoticz.

# Domoticz

## Installation

The Domoticz installation is initiated by running terminal command:

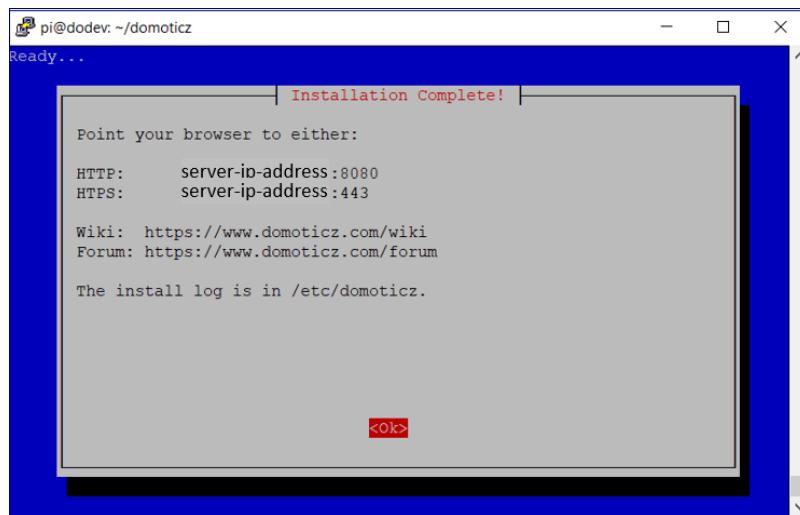
```
curl -L install.domoticz.com | sudo bash
```

Answers to the questions

Question	Answer
Services	http, Port: 8080
Domoticz Folder	/home/pi/domoticz
Domoticz URL	http://domoticz-ip:8080 or http://localhost:8080 (when using the monitor connected to Raspberry Pi)

If everything went ok, enter the Domoticz URL in a browser and ... the Domoticz UI is shown. The Domoticz version is displayed as V4.9700 (or newer).

*Example Domoticz Installation Complete Message (for the development system)*



Now READY to configure the Domoticz Home Automation System solution further.

## Settings

Changes made to the default Domoticz settings:

System	
User Interface	Language: English Theme: Default
Location	Name: Pinneberg (Pro) Latitude: 53.636470 Longitude: 9.798251 <i>Note</i> The browser shows as tab Pinneberg (Pro) 
Dashboard	Mode: Mobile (for access via Smartphone)
Local Networks	Networks: NNN.NNN.1.*
Software Updates	Check Release Channel: beta
<b>Log History</b>	Set to 1 days
<b>Notifications</b>	See under Functions where applicable
<b>Email</b>	
Email enabled	username <email-address>
Send Email notification alerts	Enabled
<b>Meters / Counters</b>	
Wind Meter	Display:Beaufort
<b>Floorplan</b>	See under Functions where applicable
<b>Other</b>	
Event System	Enabled
dzVents	Enabled
	<i>Note</i> Automation events are developed as dzVents Lua scripts only.

## Folders

Folder	Path
Domoticz Home	/home/pi/domoticz
Scripts	/home/pi/domoticz/scripts
Bash scripts -created new folder, make sure all scripts are executable, i.e. sudo chmod +x myscripts.sh	/home/pi/domoticz/scripts/bash

## Events

Events are developed using mainly dzVents (Domoticz Easy Events), which is Domoticz Next Generation Lua scripting (described [here](#)).

The Domoticz internal event editor is used.

External Lua modules have been defined, to provide common functions across the dzVents scripts.

In the meantime, these modules are replaced by a dzVents script global\_data, maintained using the internal event editor.

There are a few exceptions, where an event is written in Python and triggered via crontab. These events are not yet converted to Lua (planned).

# Functions

## Introduction

This project has a modular setup, which are called **functions**.

Each **function** has a specific purpose, makes use of required hardware & software and Domoticz configuration settings.

### Function examples are

- RFXCOM - wireless receive & transmit between devices/sensors at 433.92Mhz.  
(Temperature Sensors, Wind Sensors, Light/Switch Devices)
- Philips Hue Light Control (Light/Switch Devices)
- Waste Calendar – inform about waste dates for residual waste, organic waste, paper waste, plastic waste (Utility Sensors)
- Days since / to calculation (Utility Sensors)
- Raspberry Pi system information (Utility Sensors)

Just a few functions to mention:

- ▷ RFXCOM RFxTrx433E
- ▷ Temperature & Humidity
- ▷ Anemometer
- ▷ Wind
- ▷ Somfy Blinds
- ▷ Airpressure BMP280
- ▷ Hue
- ▷ Remote Switch
- ▷ Waste Calendar
- ▷ Days-To-Go
- ▷ **Raspberry Pi Monitoring**
- ▷ Electric Usage
- ▷ Coffee Machine Monitor
- ▷ Alert Message
- ▷ User Variables

# Air Pressure (BME280)

## Purpose

To measure the Air Pressure using a Waveshare BME280 Environmental Sensor Module connected to the GPIO of the Raspberry Pi.

### Important

This function is replaced by [BME280 Temp+Hum+Baro](#) and left in this document for reference – this chapter is not updated anymore.

## Wiring

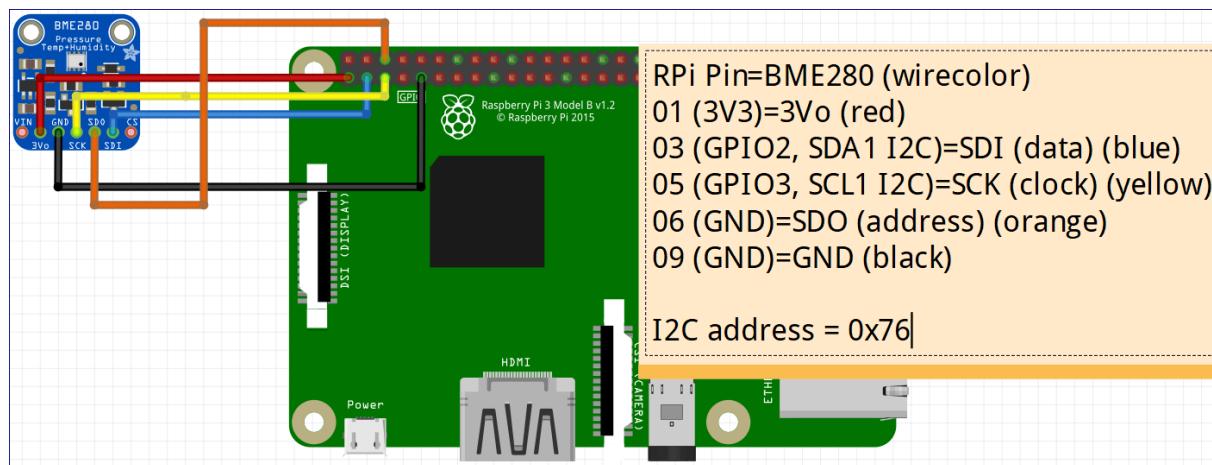
BME280	Raspberry Pi
3V	Pin #1 3V3
GND	Pin #9 GND
SDI Data	Pin #3 GPIO2, SDA1 I2C
SCL Clock	Pin #5 GPIO3, SCL1 I2C
SDO Address	Pin #6 GND
GND	Pin #9 GND

### Note

The I2C address used is 0x76

## Circuit

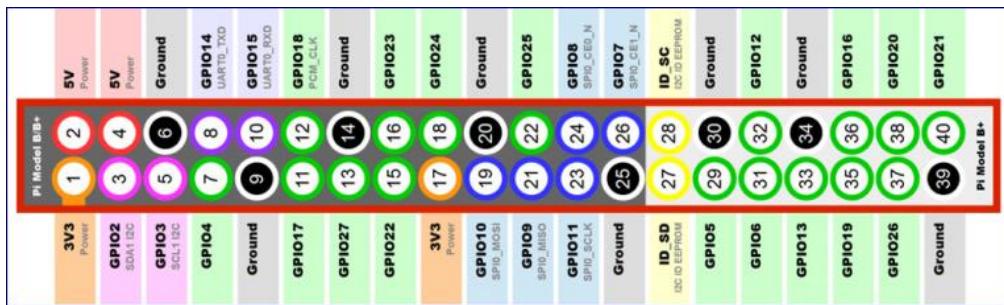
The BME280 sensor (Waveshare) connected to the Raspberry Pi:



### Notes

- By connecting SDO to Ground, the default I2C Address 0x76 is set.  
Use the terminal command: “i2cdetect -y 1” to determine the I2C address.
- Domoticz uses the default address 0x76.
- Adding multiple I2C sensors, requires a unique address for each sensor and two 1K resistors between VCC and the Data (SDA) & Clock (SCL) lines.

## Raspberry Pi B+ GPIO Header



## I2C

Check if the I2C module (i2c\_bcm2835) is loaded using the terminal command “lsmod”.

If not, activate the I2C interface in the Raspberry Pi preferences.

Below list is only an extract of the various command outputs.

```
$lsmod
Module           Size Used by
i2c_bcm2835      16384  0
spi_bcm2835      16384  0
w1_gpio          16384  0
wire              40960  1 w1_gpio
i2c_dev          16384  0
ip_tables        24576  0
x_tables          32768  1 ip_tables
ipv6             434176  58

$ls -l /dev/i2c-
crw-rw---- 1 root i2c 89, 1 Aug 30 12:08 /dev/i2c-1

$i2cdetect -l
i2c-1  i2c          bcm2835 I2C adapter
i2cdetect -y 1          I2C adapter
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: -----
20: -----
30: -----
40: -----
50: -----
60: -----
70: ----- 76 --
```

## Domoticz Configuration

GUI > Setup > Hardware > Select Type, give Name, set path I2C bus > Add

Enabled:

Name: BME280

Type: I2C sensors

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
Do not enable this option for devices that do not receive data!

SubType: I2C sensor BME280 Temp+Hum+Baro

Path to I2C bus: /dev/i2c-1 (etc. /dev/i2c-1, or empty for autodetection)

**Add**

### Note

The path to the I2C Bus is given by this terminal command:

```
$ls -l /dev/i2c-*
crw-rw---- 1 root i2c 89, 1 Aug 30 12:08 /dev/i2c-1
```

### Check if the hardware has been added to the hardware list

Name	Enabled	Type	Address	Port	Data Timeout
BME280	Yes	I2C sensor BME280 Temp+Hum+Baro	I2C	/dev/i2c-1	Disabled

### Check the Domoticz log after adding the hardware

```
2018-08-31 11:34:50.758 Status: I2C Start HW wif ID: 13 Name: I2C_BME280 Address: 0 Port: /dev/i2c-1
Invert:0
2018-08-31 11:34:50.758 Status: I2C_BME280: Worker started...
```

### Add new Device

Setup > Settings > Hardware/ Devices > Allow new Devices 5 minutes

A new device is added with IDX=115 (Setup > Devices).

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
115	BME280	0001	1	Server Room	Temp + Humidity + Baro	Weather Station	22.6 C, 53 %, 1009.7 hPa

Add the device, named BME280 and check the widgets.

TemperatureWidget	Weather Widget
<p>BME280  22.5° C / 53%</p> <p>Comfortable, Barometer: 1010 hPa, Prediction: Cloudy Dew Point: 12.45° C Last Seen: 2019-07-18 15:49:08</p> <p> Log Edit Notifications</p>	<p>BME280  1010 hPa</p> <p>Prediction: Cloudy Last Seen: 2019-07-18 15:49:38</p> <p> Log Edit Notifications</p>

## Barometer Widget

The BME280 measures pressure, temperature, and humidity.

As the BME280 is directly connected to the Raspberry Pi, the temperature and the humidity are used to monitor the Server Room (holding various hardware).

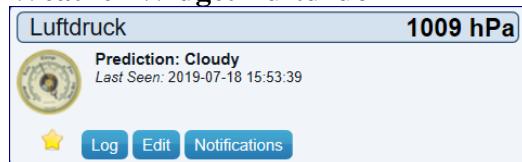
In the Domoticz GUI, a Barometer Widget holds the value taken from the BME280.

The solution is to create a Virtual Sensor Type Barometer named “Luftdruck”.

### Luftdruck Virtual Sensor Device List Entry (IDX=116)

116	VirtualSensors	00082116	1	Luftdruck	General	Barometer	1027.6 hPa
-----	----------------	----------	---	-----------	---------	-----------	------------

### Weather Widget Luftdruck



## Automation Script

Created a dzVents script to update the pressure and pressure forecast from the BME280 property “barometer”.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
115	BMP280	0001	1	BME280	Temp + Humidity + Baro	Weather Station	22.5 C, 53 %, 1010.0 hPa

The Data holds: TEMP = Temperature C, HUM = Humidity %, BAR = Barometric pressure.

### dzVents Lua Script

Event name: airpressure\_timer\_update

```
--[[[airpressure_timer_update.dzvents
  Timer trigger to update the Virtual Sensor Barometer named Air Pressure (idx=116) from the BME280
  device (idx=115) property barometer.
  The BME280 is connected to the Domoticz Production system GPIO and is placed in the server room.
  20200908 rwbl
]]--]

-- Idx of the devices
-- Idx, Hardware, Name, Type, SubType, Data
-- 115, BMP280, Server Room, Temp + Humidity + Baro, Weather Station, 20.9 C, 58 %, 1021.0 hPa
local IDX_BME280 = 115;
-- 116, VirtualSensors, Luftdruck, General, Barometer, 1021 hPa
local IDX_AIRPRESSURE = 116;
local DEF_BME280_COMPENSATION = 0

return {
  on = {
    timer = { 'every 15 minutes' },
  },
  execute = function(domoticz, item)
    -- Get the airpressure
    local airpressure = domoticz.devices(IDX_BME280).barometer;
    -- Round the pressure
    airpressure = math.floor(airpressure) + DEF_BME280_COMPENSATION
    -- Get the forecast
    local forecastString = string.upper(domoticz.devices(IDX_BME280).forecastString);
    -- Forecast from the BMP device to the Barometer device
  end
}
```

```
-- See source code RFXNames.cpp, BMP_Forecast_Desc
-- Map the BMP_Forecase_Desc string to dzEvents Forecast value:
-- domoticz.BARO_STABLE, BARO_SUNNY, BARO_CLOUDY, BARO_UNSTABLE, BARO_THUNDERSTORM
local forecast = "UNKNOWN"
if (forecastString == 'CLOUDY') then forecast = domoticz.BARO_CLOUDY end
if (forecastString == 'CLOUDY/RAIN') then forecast = domoticz.BARO_CLOUDY end
if (forecastString == 'STABLE') then forecast = domoticz.BARO_STABLE end
if (forecastString == 'SUNNY') then forecast = domoticz.BARO_SUNNY end
if (forecastString == 'THUNDERSTORM') then forecast = domoticz.BARO_THUNDERSTORM end
if (forecastString == 'UNKNOWN') then forecast = domoticz.BARO_STABLE end
if (forecastString == 'UNSTABLE') then forecast = domoticz.BARO_UNSTABLE end
-- Update the virtual sensor Luftdruck
domoticz.devices(IDX_AIRPRESSURE).updateBarometer(airpressure, forecast)
domoticz.log(string.format('Device %s updated:%.0f %s', domoticz.devices(IDX_BME280).name,
airpressure, forecast), domoticz.LOG_INFO)
end
}
```

### Note

The dzVents script airpressure\_device\_update.dzvents is not used, because running rather often, up-to 3 times/minute (and even more), based on device data changes.

### Domoticz Log

```
2020-09-09 08:45:00.499 Status: dzVents: Info: ----- Start internal script: airpressure_timer_update:, trigger: "every 15 minutes"
2020-09-09 08:45:00.515 Status: dzVents: Info: Device Server Raum updated:1018 cloudy
2020-09-09 08:45:00.515 Status: dzVents: Info: ----- Finished airpressure_timer_update
```

# Alert Message

## Purpose

To display an Alert Message on the GUI > Dashboard and GUI > Utility.

If the Alert Level is 4 (ALERTLEVEL\_RED), send an email notification to the recipients defined in the GUI > Setup > Settings > Email.

The Alert Message is populated by various dzEvents, i.e.

- Raspberry Pi monitor (Script: rpi\_monitor.lua)
- Hue Lights living room timed switch (Script: hue\_wz\_timer.lua)
- More see the dedicated functions

## Solution

Add a Virtual Device

<b>Idx,Hardware,Name,Type,SubType,Data</b>
55,VirtualSensors,Alert Messages,General,Alert, Message Text

Add a User Variable

<b>Idx,Name,Type,Value</b>
6,IDX_ALERTMSG,Integer,55

*Note*

The value contains the idx of the virtual device “Alert Messages”.

## Dashboard

A screenshot of the Domoticz Dashboard. At the top, there's a header bar with 'Utility Sensors' and a small icon. Below it, a status bar shows '08.07.2019 14:41 - Postbox opened!' with a red warning icon. The main area shows a single alert message card:

## Widget

A screenshot of a 'Alert Messages' widget. It displays a single log entry:

- Icon: Red exclamation mark
- Date: 08.07.2019 14:41
- Message: Postbox opened!
- Details: Last Seen: 2019-07-08 14:41:39, Type: General, Alert
- Buttons: Log, Edit, Notifications

## Log Entries

A screenshot of the 'Alert Messages' log entries table. It shows the following data:

Date	Data
2019-07-08 14:41:39	08.07.2019 14:41 - Postbox opened!
2019-07-07 23:00:00	Hue Lampen aus 23:00
2019-07-07 21:21:00	Hue Lampen an 21:21
2019-07-07 20:51:01	Hue Lampen gehen an 21:21

## Automation Script

Monitor the level of the Alert Messages Device with IDX=55.

If the level (nValue) > 4 (or any other value set by uservar TH\_ALERTTOEMAIL), send the Alert Message to Email recipients.

Event name: alertmsg\_monitor

```
local IDX_ALERTMSG = 55
local IDX_TH_ALERTTOEMAIL = 14

return {
  on = {
    devices = { IDX_ALERTMSG }
  },
  execute = function(domoticz, device)
    -- Send email notification in case level = 4 (or other see user var TH_ALERTTOEMAIL)
    if (device.nValue == domoticz.variables(IDX_TH_ALERTTOEMAIL).value) then
      domoticz.notify('ALERT:' .. device.text, device.text, domoticz.PRIORITY_HIGH)
    end
  end
}
```

Switch the Hue MakeLab ON and update the Alert Messages Device with a message.

Event name: hue\_control\_makelab

Depends on: global\_data

```
local IDX_HUE_MAKELAB = 118;

return {
  on = {
    devices = { IDX_HUE_MAKELAB }
  },
  execute = function(domoticz, device)
    domoticz.log('device.name .. ' changed to ' .. device.state, domoticz.LOG_INFO)
    if (device.state == 'On') then
      local msg = device.name .. ' switched on at ' .. domoticz.helpers.isnow(domoticz)
      domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, msg)
      domoticz.log(msg, domoticz.LOG_INFO)
    end
  end
}
```

For all events, the helper functions are defined in the dzVents Lua script “global\_data”.

For this example, the helper functions “isnow” and “alertmsg” are used which are referenced as

- domoticz.helpers.isnow(domoticz)
- domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL\_YELLOW, msg)

# Alert Message Reset

## Purpose

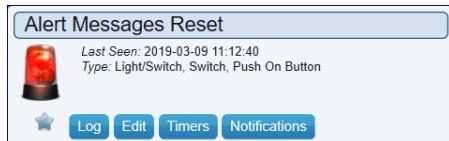
To reset the Alert Message to a default text using a Switch.

## Solution

Create Switch Device (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
119,VirtualSensors,Alert Messages Reset,Light/Switch,Switch,On/Off,Off

If the switch is turned ON, the text of the alert sensor (Type=General, SubType=Alert, IDX=55) is set to the value of the User Variable DEF\_ALERTMSG is set and the switch is turned OFF again. This is handled using a dzVents script.



## Create User Variable

Idx,Name,Type,Value
7,DEF_ALERTMSG,String,Keine

## dzVents Script

Event name: alertmsg\_reset

Depends on: global\_data

```
local IDX_ALERTMSG_RESET = 119

return {
  on = {
    devices = { IDX_ALERTMSG_RESET }
  },
  execute = function(domoticz, device)
    domoticz.log('Device ' .. device.name .. ' was changed ' .. device.state)
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).text, domoticz.LOG_INFO)

    if (device.state == 'On') then
      -- only change if the current text differs from the default text
      if (domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).text ~= domoticz.variables('DEF_ALERTMSG').value) then
        local message = domoticz.variables('DEF_ALERTMSG').value
        domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_GREY, message)
        domoticz.log(message)
      end
    end
  end
}
```

## Database Tables

The Alert Message device (IDX=55) is stored in the Domoticz database “domoticz.db”, table “DevicesStatus”.

ID	Har...	Dev...	Unit	Name	Used	Type	Sub...	Swit...	Fav...	Sign...	Batt...	nVal...	sValue	LastUpdate	Order	Addj...	Addj...	Addj...	Addj...	StrP...	StrP...	Last...	Prot...	Cust...	Des...	Opt...	
55	6	82054	1	Alert Meldung	1	243	22	0	1	12	255	0	Reset	2018-09-13 13:57:03	55	0.0	1.0	0.0	1.0	1.0	StrP...	StrP...	Last 0	Prot 0	Cust 0	Des 0	Opt

The Alert Message device log entries are stored in the table “LightingLog”.  
The field DeviceRowID is the IDX of the device, i.e. 55.

SELECT * FROM LightingLog WHERE DeviceRowID=55;				
Devi...	nVal...	sValue	Date	User
55	2	[ERROR] Powerconsumption: 7	2018-09-13 11:50:00	
55	2	[ERROR] Powerconsumption: 7	2018-09-13 11:55:00	
55	2	[ERROR] Electric usage: 7	2018-09-13 12:00:00	
55	2	[ERROR] Powerconsumption: 7	2018-09-13 12:00:00	
55	2	[ERROR] Electric usage: 7	2018-09-13 12:05:00	
55	2	[ERROR] Powerconsumption: 7	2018-09-13 12:05:00	
55	2	[ERROR] Powerconsumption: 7	2018-09-13 12:10:00	
55	0	Keine	2018-09-13 13:50:36	
55	0	Keine	2018-09-13 13:50:42	
55	0	Reset	2018-09-13 13:57:03	

To delete entries, option are:

The GUI > Utility > Alert Message Widget > Log > Clear

The [SQLite Shell](#) (stop Domoticz, delete the rows using an SQL statement, start Domoticz).  
This is an example using another IDX=12 with some dummy data.

```
$sudo service domoticz.sh stop
$cd /home/pi/domoticz
$sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
sqlite> .open domoticz.db
sqlite> select * from lightinglog where devicerowid=12;
12|2|20|Admin|2020-03-02 10:28:18
12|2|10|Admin|2020-03-02 10:28:21
12|2|20|Admin|2020-03-02 10:28:22
12|2|10|Admin|2020-03-02 10:28:24
12|2|20|Admin|2020-03-02 10:28:25
sqlite> delete from lightinglog where devicerowid=12;
sqlite> select * from lightinglog where devicerowid=12;
sqlite> .exit
$sudo service domoticz.sh start
```

# Ambient Light (BH1750)

## Purpose

To measure, using an ESP8266 microcontroller running [ESP Easy](#), the Ambient Light in Lux with an BH1750 sensor and send the value to a Domoticz Device. A light bulb is switched ON when the Lux value is below a threshold.

The BH1750 sensor communicates with the ESP8266 microcontroller via I2C.

## Parts List

- 1x ESP8266 = used a NodeMCU v1.0 with 4M
- 1x BH1750FVI Sensor

## Wiring

BH1750	NodeMCU
VCC	3V3
GND	GND
SCL	D1 (GPIO5)
SDA	D2 (GPIO4)

The BH1750 sensor supports two addresses depending the ADDR pin connection:

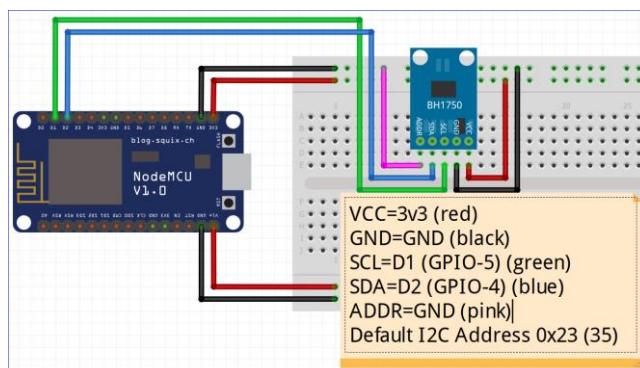
BH1750 Pin	ESP8266 Pin	I2C Address
ADDR	A3 or GND	0x23 (Default)
ADDR	Vin	0x5C

### Note

This BH1750 sensor uses the default address 0x23 (ADDR pin is connected to GND).

## Circuit

The circuit shows the NodeMCU with BH1750 connected to I2C pins.



## Device Setup Domoticz

Create a virtual sensor: Sensor Type Lux with Name “ESPEasy Ambient Light”.

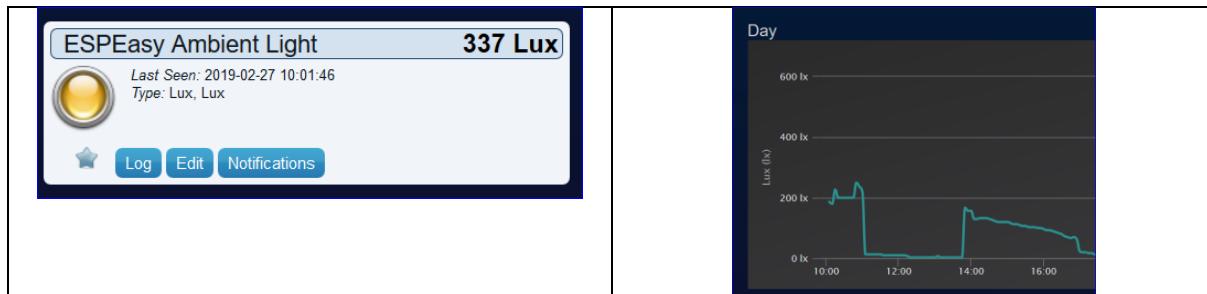
After adding the new device, the idx is required for the ESP device settings: IDX=46.

## Devices List Entry

 46 VirtualSensors 82045 1 ESPEasy Ambient Light Lux Lux 177 Lux 9
---

## Widget

The widget and day chart showing actual values.



## Device Setup ESP Easy

Define the device via a browser by opening:

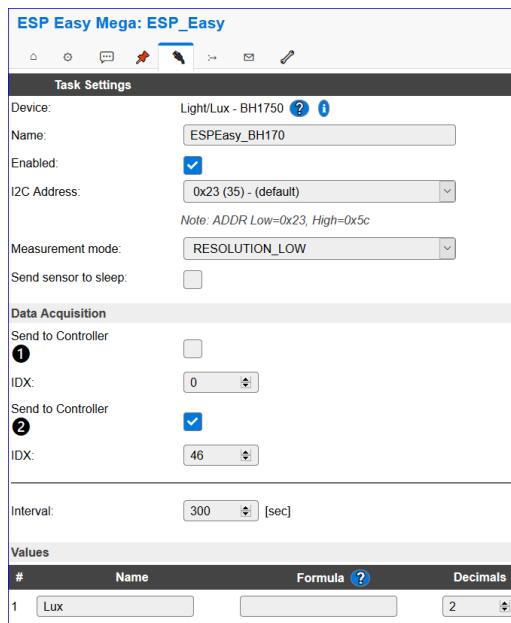
<http://esp-device-ip> > goto Devices > select Task Edit and configure.

The Domoticz MQTT controller is used and Domoticz device with IDX=46 (as previous defined).

The Lux value is updated every 300 seconds (5 minutes) to Domoticz.

### Domoticz Log

```
2019-02-26 10:10:08.783 MQTT: Topic: domoticz/in, Message:
{"idx":46,"RSSI":10,"nvalue":0,"svalue":"183.33"}
```



## ESP Easy Devices List Entry

<input type="button" value="Edit"/>	4	<input checked="" type="checkbox"/>		Light/Lux - BH1750	ESPEasy_BH170			<input checked="" type="checkbox"/> (46)	GPIO-4 GPIO-5	Lux: <input type="text" value="290.00"/>
-------------------------------------	---	-------------------------------------	--	--------------------	---------------	--	--	--	------------------	--

## Trigger Switch Light

If the Domoticz Ambient Light device (IDX=46) Lux value is below threshold, then switch on a light. The light used, is a Philips Hue named MakeLab with IDX=118. Instead a light as switch, other switches could be used as well.

### Threshold

Defined as Domoticz user variable: TH\_AMBIENT\_LIGHT with value integer 100.  
(Setup > More Options > User variables)

Idx	Variable name	Variable type	Current value
10	TH_AMBIENT_LIGHT	Integer	100

### Automation Script

Listen to device changes (device IDX=46), check if the value is below threshold (user variable IDX=9), switch light (device IDX=118).

Event name: ambient\_light\_control\_espeasy

```
--[[[ambient_light_control_espeasy.lua
Measure the ambient light Lux value and if below threshold switch on hue light.
]]--]

-- External modules:
local msgbox = require('msgbox')

-- Idx of the devices used
local IDX_AMBIENT_LIGHT = 46
local IDX_HUE_MAKELAB = 118
local IDX_TH_AMBIENT_LIGHT = 10

return {
  on = {
    devices = {
      IDX_AMBIENT_LIGHT
    }
  },
  execute = function(domoticz, device)
    -- get the threshold
    local althreshold = domoticz.variables(IDX_TH_AMBIENT_LIGHT).value

    domoticz.log('Device ' .. device.name .. ' changed to ' .. tostring(device.lux) .. ' / ' ..
    tostring(althreshold), domoticz.LOG_INFO)

    -- check threshold
    if (device.lux < althreshold) and (domoticz.devices(IDX_HUE_MAKELAB).state == 'Off') then
      domoticz.log('Ambient Light below threshold. Switched on light', domoticz.LOG_INFO)
      domoticz.devices(IDX_HUE_MAKELAB).switchOn()
      -- optional check leveland set to default 20%
      if domoticz.devices(IDX_HUE_MAKELAB).level < 20 then
        domoticz.devices(IDX_HUE_MAKELAB).dimTo(20)
      end
    end
  end
}
```

### Domoticz Log

The Lux value is below threshold, resulting in switching on the light.

```
2019-02-27 10:52:05.762 MQTT: Topic: domoticz/in, Message:  
{"idx":46,"RSSI":3,"nvalue":0,"svalue":"23.33"}  
2019-02-27 10:52:05.933 Status: dzVents: Info: Handling events for: "ESPEasy Ambient Light", value:  
"23.33"  
2019-02-27 10:52:05.933 Status: dzVents: Info: ----- Start internal script:  
ambient_light_control_espeasy: Device: "ESPEasy Ambient Light (VirtualSensors)", Index: 46  
2019-02-27 10:52:05.934 Status: dzVents: Info: Device ESPEasy Ambient Light changed to 23.33 / 100  
2019-02-27 10:52:05.934 Status: dzVents: Info: Ambient Light below threshold. Switched on light  
2019-02-27 10:52:05.935 Status: dzVents: Info: ----- Finished ambient_light_control_espeasy
```

# Android App atHome

## Purpose

To develop an Android & Windows Desktop App displaying & controlling the Domoticz favourite devices, i.e. the devices shown in the Domoticz GUI Tab Dashboard.

## Solutions

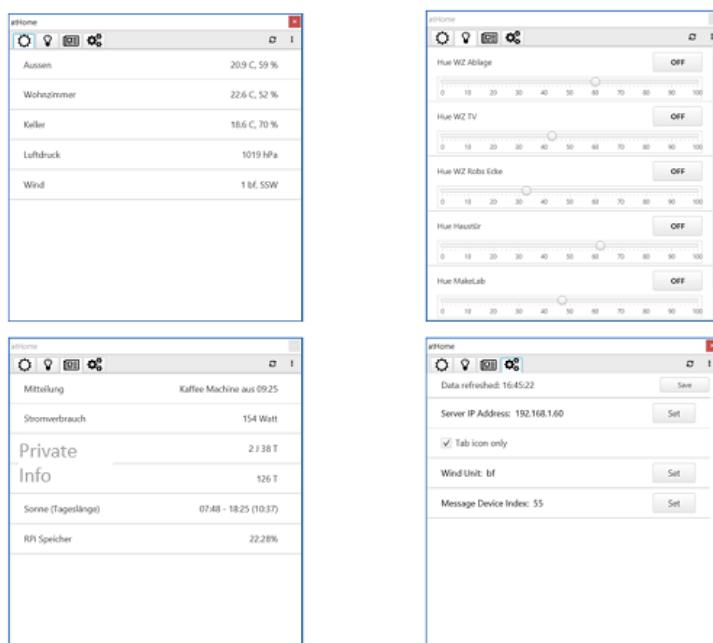
Android app developed with [B4A](#) v8.5.

Read the B4A Forum *Share My Creations* post [here](#).



Windows Desktop application developed with [B4J](#) v6.8.

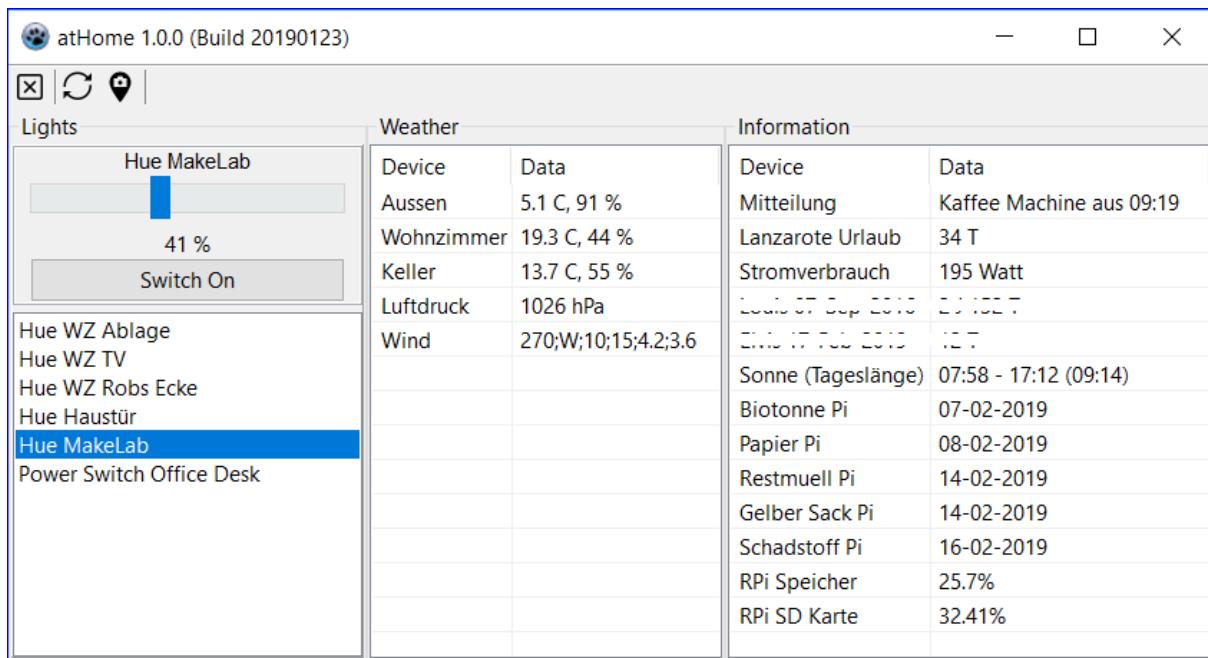
Read the B4J Forum *Share My Creations* post [here](#).



Note: current version in use has many more devices.

## Windows Desktop application developed with [Lazarus](#) v1.8.4.

Not published yet.



# Anemometer (TFA 30.3168)

## Purpose

To measure the wind speed (m/s, bft), direction and air temperature.

### Note

Only the wind speed & direction of this device are used and assigned to a Virtual Sensor Wind (see Function Wind) which is used for the Dashboard.

## Device

Device (433MHz): Anemometer **TFA Dostmann 30.3168 Windmeter**

Location: Garden shed (“Gartenhaus”).

The anemometer is connected to the RFXCOM RFXtrx433E.

## Setup

1. Open the device battery flip and take out the battery blocker
2. The device will be recognized automatically and listed under the devices list
3. Select the green arrow to give the device the name and the name will be shown in the list (see below)

### Anemometer Device List Entry (IDX=28)

28	RFXtrx433e	9D16	0	Windmesser	Wind	TFA	293.00;WNW;9;9;14.6;14.6
----	------------	------	---	------------	------	-----	--------------------------

### Anemometer Widgets (Tabs Utility & Temperature)



# Battery Status

## Purpose

To daily check the battery low level status of Domoticz used devices and external system devices, like RaspberryMatic.

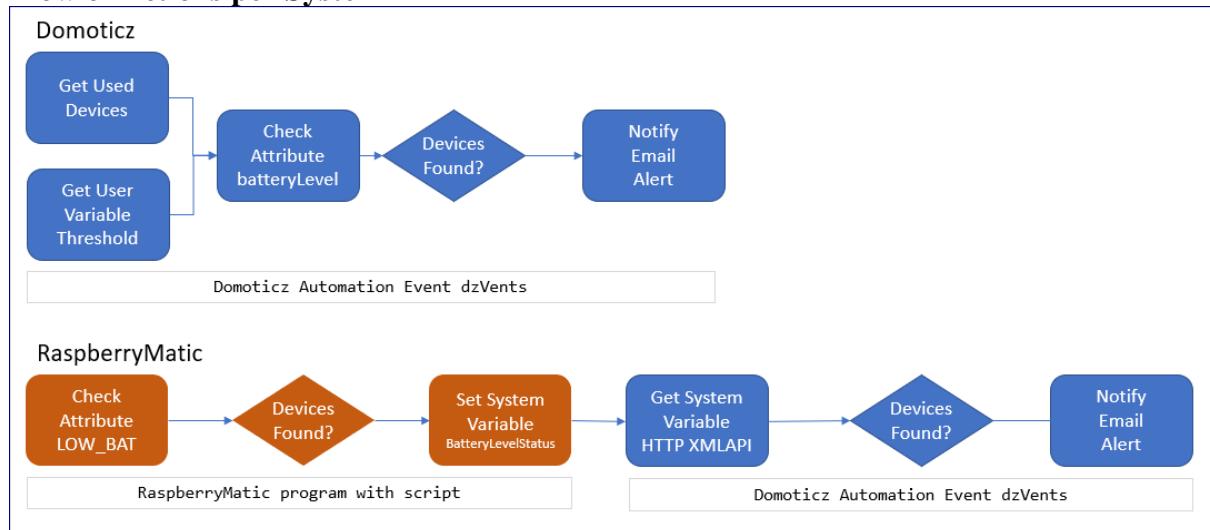
To notify, in case devices found having battery low level, via email (per system, i.e. Domoticz, RaspberryMatic ...) and update an Alert message.

To be able to disable the daily battery low level status checks.

## Solution

Each system has its own solution to check the Battery Low Level status but resulting in a common way to notify via email & [Alert Message](#).

### Flow of Actions per System



Brief description of the flow.

### Domoticz

Daily at 01:00 the battery level for all devices is checked, via a Domoticz Automation Event dzVents against a threshold (defined as Domoticz User Variable).

If the battery level for a device is below threshold, the device name & battery level are added to a message.

After check completion, the message with the devices found is sent via email and to an Alert device. No action is taken if there are no devices found.

### Example Email and Alert Message

From	Subject	Meldungen	Domoticz Batterie Prüfung: Keller Großraum: 0%; Windmesser: 0%;
rwbliinn@outlook.de	Domoticz Batterie Prüfung Keller Großraum: 0%; Windmesser: 0%; <end>		

## RaspberryMatic

Daily at 00:00 a RaspberryMatic program executes a script which loops over all devices to check the device attribute LOW\_BAT. If there are devices found, the device name is added to a message. If there are no devices found, the message contains the text OK. The message is used to update a RaspberryMatic system variable BatteryLevelStatus.

At 01:00 a Domoticz Automation Event dzVents requests via an HTTP XMLAPI request the value of the RaspberryMatic system variable BatteryLevelStatus via its datapoint id (attribute ise\_id). If the value does not contain the text OK, the message with the devices found is sent via email and to an Alert device.

## Domoticz

### Configuration

#### User Variable

Idx	Variable name	Variable type	Current value
24	TH_BATTERY_LEVEL	Integer	20

The current value is the battery level threshold in percent.

#### Disable Battery Level Check

Set the value to 0 to disable the daily battery level check

#### Note

The user variable can also be set via the Custom page [Custom Settings](#).

## Automation Event

Source: battery\_status\_domoticz.dzvents

```
-- [[
battery_status_domoticz
Once per day at 01:00, check the battery level of all used Domoticz devices
against a threshold set by a user variable.
Threshold = 0 battery check is disabled.
Battery level below threshold, add the device name and battery level to a message.
If devices found, send an email, and update an alert sensor.
20201023 rwbl
]]-- 

-- Subject of the message for email and alert
local MSGSUBJECT = 'Domoticz Batterie Stand Niedrig'
-- Idx of the user variable holding the battery level threshold (n %)
local IDX_TH_BATTERYLEVEL = 24
-- Set a default threshold in case the user var is not found
local DEF_THRESHOLD = 20
-- Timer Rule
local TIMER_RULE = 'at 01:00'
-- local TIMER_RULE = 'every minute' -- TESTS

return {
```

```
on = {
    timer = {
        TIMER_RULE,
    },
    logging =
    {
        level = domoticz.LOG_INFO,
        marker = 'DOMOTICZBATTERYSTATUS',
    },
},
execute = function(domoticz, timer)
    -- Get the battery level threshold from the user variable
    local threshold = domoticz.variables(IDX_TH_BATTERYLEVEL).value
    -- If no user variable found set default threshold
    if threshold == nil then threshold = DEF_THRESHOLD end
if threshold > 0 then
    -- Loop over all devices and check the property batteryLevel
    -- !Ensure the right case for the device attribute -- see dzVents documentation
    local message = 'OK'
    domoticz.devices().forEach(function(device)
        if device.batteryLevel ~= nil then
            if message == 'OK' then message = '' end
            if device.batteryLevel < threshold then
                message = message .. ('%s:%d%'):format(device.name, device.batteryLevel)
                -- message = message .. ('%s;'):format(device.name)
                -- domoticz.log(('Device %s Battery Level %d%'):format(device.name,
device.batteryLevel),domoticz.LOG_ERROR)
            end
        end
    end)
    domoticz.log(message, domoticz.LOG_INFO)
    if message ~= 'OK' then
        -- Send notification via mail
        domoticz.notify(MSGSUBJECT, message, domoticz.PRIORITY_HIGH)
        -- update the alert message with level orange (level = 3)
        -- If user var TH_ALERTTOEMAIL = 4 then no email is sent because level < 4
        domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW,
        ('%s:%s'):format(MSGSUBJECT, message))
    end
end
}
}
```

# RaspberryMatic

## Program

The RaspberryMatic program runs daily at 00:00 to check the device LOW\_BAT attribute against value true. The system variable BatteryLevelStatus is updated either with value OK or list of devices with LOW\_BAT.

The screenshot shows the RaspberryMatic web interface under the 'Programs and connections' tab. A single program is listed:

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Battery Status Domoticz	Check the battery LOW_BAT state for all devices.	Time: Daily at 00:00 o'clock starting at 21.10.2020 trigger at point of time	Script: ... immediately run	<input type="checkbox"/> intrinsic

**Condition: If...**

- Time control: Daily at 00:00 o'clock starting at 21.10.2020 trigger at point of time
- AND
- OR

**Activity: Then...**  Stop all current delays before performing the activity (e.g. retriggering).

Script: !battery\_status\_raspberrymatic.script! Once per day, loop ... Immediately

**Activity: Else...**  Stop all current delays before performing the activity (e.g. retriggering).

## System Variable

The system variable BatteryLevelStatus definition and log.

The screenshot shows the RaspberryMatic web interface under the 'Settings' tab, specifically the 'System variables' section. A table lists system variables, and a modal dialog is open to edit the 'BatteryLevelStatus' variable.

Name	Description	Variable type	Values	Unit of measurement	Channel assignment	Action	Connections
Filter		Filter		Filter	Filter		
Alarm zone 1	Alarm message alarm zone 1	Alarm	true = triggered false = not triggered			<input type="button" value="Delete"/> <input checked="" type="checkbox"/> visible <input type="button" value="Edit"/> <input type="checkbox"/> logged	<input type="button" value="Programs"/>

**Edit system variable**

Name	Description	Variable type	Values	Unit of measurement	Channel assignment
BatteryLevelStatus	Battery level status checke	Character string			<input type="radio"/> without <input type="radio"/> with <input type="checkbox"/> Channel selection

**Buttons:** Cancel, OK

Name	Description	Last modified	Status
Filter			
BatteryLevelStatus	Battery level status checked daily	22.10.2020 17:47:41	OK

## Script

Source: battery\_status\_raspberrymatic.script

```
! battery_status_raspberrymatic.script
! Once per day, loop over all devices to check the value of datapoint LOW_BAT against true (means the
battery of the device is low)
! Update system variable BatteryLevelStatus with devices found or OK if all devices are fine.
! 20201023 rwbl

! System variabe holding the text of the battery level check.
string BATTERYLEVELSTATUS = "BatteryLevelStatus";
! Message text to save in the system variable BatteryLevelStatus
string text = "";
! Number of devices with LOW_BAT = true
integer counter = 0;

! Get all devices objects
var objIDs = dom.GetObject(ID_DEVICES).EnumUsedIDs();

! Loop over all datapoints for the devices found
string id;
foreach(id, objIDs){
    !Get datapoint
    var object = dom.GetObject(id);
    ! Name: Briefkasten Status: ID=2530
    ! WriteLine("Name: " # object.Name() # ": ID=" # object.ID());

    ! Get the device name from the object
    string DeviceName = object.Name();
    ! Get the the device object from the device name, i.e. Briefkasten Status
    var Device = dom.GetObject(DeviceName);
    string id;
    ! Loop over all channels to get the datapoint LOW_BAT
    foreach(id, Device.Channels()) {
        ! Get the channel
        var ch = dom.GetObject(id);
        ! Get the datapoint for the attribute LOW_BAT from the channel
        var dp = ch.DPByHssDP("LOW_BAT");
        if (dp) {
            ! dpValue() true|false. If true the battery is low
            if (dp.Value() == true) {
                counter = counter + 1;
                text = text # DeviceName # ";";
                ! WriteLine(DeviceName # ", Battery LOW = " # dp.Value());
            }
        }
    }
}

! If no devices with low battery found, set text to OK
if (counter == 0) {
    text = "OK";
}

! Update the systemvariable
var objBatteryLevelStatus = dom.GetObject(BATTERYLEVELSTATUS);
objBatteryLevelStatus.Variable(text);
```

```
! Log for tests the content of the system variable
! WriteLine(objBatteryLevelStatus.Variable());
```

## Automation Event

Source: battery\_status\_raspberrymatic.dzvents

```
--[[
battery_status_raspberrymatic
Once per day, check the battery level of all raspberrymatic (homematic ip) devices.
A raspberrymatic script runs every night and sets a system variable BatteryLevelStatus type string.
The raspberrymatic device datapoint LOW_BAT is checked true (battery is low) or false (battery ok).
Via HTTP XMLAPI request sysvar.cgi for system variable datapoint, the system variable value
(string) is retrieved.
If the value does not contain "OK", there are devices with low battery state.
Threshold = 0 battery check is disabled.
If devices found, sent an email, and update an alert sensor.
20201023 rwbl
]]--


-- Subject of the message for email and alert
local MSGSUBJECT = 'RaspberryMatic Batterie Prüfung'
-- Idx of the user variable holding the battery level threshold (n %)
local IDX_TH_BATTERYLEVEL = 24
-- Set a default threshold in case the user var is not found
local DEF_THRESHOLD = 20
-- RaspberryMatic url http xml-api request system variable. The ise_id is set in the function openURL
local URL_RASPMATIC = 'http://raspmatic-ip/addons/xmlapi/sysvar.cgi?ise_id=';
-- define the unique (across all dzVents) callback
local RES_RASPMATIC = "RES_RASPMATIC_BATTERYSTATUS";
-- set the datapoint ise_id of the system variable "BatteryLevelStatus" with ise_id 1390
-- get the id from http://raspmatic-ip/addons/xmlapi/sysvarlist.cgi
-- <systemVariable name="BatteryLevelStatus" variable="OK" value="OK" value_list="" ise_id="9175"
min="" max="" unit="" type="20" subtype="11" logged="false" visible="true" timestamp="1603381661"
value_name_0="" value_name_1="" />
local DATAPOINT_ISE_ID = 9175;
-- Timer Rule
local TIMER_RULE = 'at 01:00'
-- local TIMER_RULE = 'every minute' -- TESTS

return {
on = {
    timer = {
        TIMER_RULE,
    },
    httpResponses = {
        RES_RASPMATIC,
    },
    logging =
    {
        level = domoticz.LOG_INFO,
        marker = 'RASPBERRYMATICBATTERYSTATUS',
    },
},
execute = function(domoticz, item)

    -- Check if the item is timer, then request RASPBERRYMATIC HTTP XMLAPI datapoint
    if (item.isTimer) then
        -- Get the battery level threshold from the user variable
        local threshold = domoticz.variables(IDX_TH_BATTERYLEVEL).value
        -- If no user variable found set default threshold
        if threshold == nil then threshold = DEF_THRESHOLD end
        if threshold > 0 then
            domoticz.openURL({url = URL_RASPMATIC .. DATAPOINT_ISE_ID, method = 'GET', callback =
RES_RASPMATIC,})
            end
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then
            local message = ''
            if (item.statusCode == 200) then
                -- Select the callback - in this case there is only one (but for any next developments
here might be more)
                if (item.callback == RES_RASPMATIC) then

```

```
-- get the attribute value from the http response and assing as message
-- <systemVariable name="BatteryLevelStatus" variable="OK" value="OK" value_list=""
value_text="" ise_id="9175" min="" max="" unit="" type="20" subtype="11" timestamp="1603381661"
value_name_0="" value_name_1=""/>
message =
domoticz_applyXPath(item.data,'//systemVariable[@ise_id="'.DATAPOINT_ISE_ID..'" ]/@value')
end
else
    domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
end
    domoticz.log(message, domoticz.LOG_INFO)
if message ~= 'OK' then
    -- Send notification via mail
    domoticz.notify(MSGSUBJECT, message, domoticz.PRIORITY_HIGH)
    -- update the alert message with level orange (level = 3)
    -- If user var TH_ALERTTOEMAIL = 4 then no email is sent because level < 4
    domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW,
(':%s'):(format(MSGSUBJECT, message)))
    end
end
}
}
```

# Blinds (Somfy RTS Pure)

## Purpose

To control (open, close, stop) Somfy Blinds (RTS Pure) using the RFXtrx433E Transceiver.

## Domoticz Configuration

1. Open Domoticz URL (like <http://domoticz-ip:8080>)
2. Domoticz GUI Tab Switches > Select Manual Switches
3. Select Manual/Light Switch
4. Define the properties (see below picture)

### Do NOT press Add Device

5. On the Remote RTS Pure Telis 1 (or other Somfy remotes):  
Press the Prog Button (red button at the back of the remote) for 3 seconds.  
The Blind will move short (=react).
6. Click Test. The blind should react short.
7. Click Add Device
8. The new device is added to the Switches

#### *Note*

When adding more devices define a new ID and set the Unit Code, i.e. 2.

The existence of a device is checked by its unique ID + Unit Code

9. The device is listed under Settings > Devices (see below, idx 13)

Idx	Hardware	ID	Unit	Name	Type	Sub Type	Data
321	RFXtrx433e	0C0000	3	WZ Rechts	RFY	RFY	Closed
319	RFXtrx433e	0A0000	3	WZ Links	RFY	RFY	Closed
320	RFXtrx433e	0B0000	4	WZ Mitte	RFY	RFY	Closed
13	RFXtrx433e	1E1E1E	2	Markise SZ	RFY	RFY	Stopped
12	RFXtrx433e	1F1F1F	1	Markise WZ	RFY	RFY	Stopped

### Notes

The ID and the Unit are randomly selected to test various values. Each ID must be unique. If the address is also used in Node-RED with the node-red-contrib-rfxcom nodes, then the max Unit value is 4 for the address setting RFY/address/unit (i.e. RFY/0x0E1E1E/2).

# Dashboard & Switches Widget

## Dashboard



Domoticz GUI Tab Switches Widgets Blind Living & Bedroom



## MQTT Messages

Example MQTT Messages for Topic="Domoticz/out", IDX=13, name="Blind Bedroom".

The messages are related to switching states Open > Stop > Close > Stop, which is Payload Property nvalue.

Open	Stop	Close	Open
domoticz/out { "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "RFY", "id" : "1E1E1E", "idx" : 13, "name" : "Blind Bedroom", "nvalue" : 1, "stype" : "RFY", "switchType" : "Blinds", "unit" : 2 }	domoticz/out { "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "RFY", "id" : "1E1E1E", "idx" : 13, "name" : "Blind Bedroom", "nvalue" : 0, "stype" : "RFY", "switchType" : "Blinds", "unit" : 2 }	domoticz/out { "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "RFY", "id" : "1E1E1E", "idx" : 13, "name" : "Blind Bedroom", "nvalue" : 3, "stype" : "RFY", "switchType" : "Blinds", "unit" : 2 }	domoticz/out { "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "RFY", "id" : "1E1E1E", "idx" : 13, "name" : "Blind Bedroom", "nvalue" : 0, "stype" : "RFY", "switchType" : "Blinds", "unit" : 2 }

# BME280 Temp+Hum+Baro

## Purpose

To measure the Air Pressure, Temperature and Humidity using a Waveshare BME280 Environmental Sensor Module connected to the GPIO of the Raspberry Pi running as Domoticz Production system.

As the BME280 is connected to the Domoticz Production system, the Temperature and Humidity are used to monitor the server room with the various devices.

The Air Pressure (Barometer) is used for weather information.

## Solution

In Domoticz, three devices are part of the solution:

Hardware	Type	SubType
BME280	Temp + Humidity + Baro	Weather Station
VirtualSensors	Temp + Humidity	THGN122/123/132, THGR122/228/238/268
VirtualSensors	General	Barometer

The BME280 provides the temperature, humidity, and barometer data. The data is then assigned to the respective virtual sensor. This is handled by an Automation Event dzVents Lua in regular intervals.

Because the data is assigned to other devices, the BME280 device can be hidden.

## Wiring

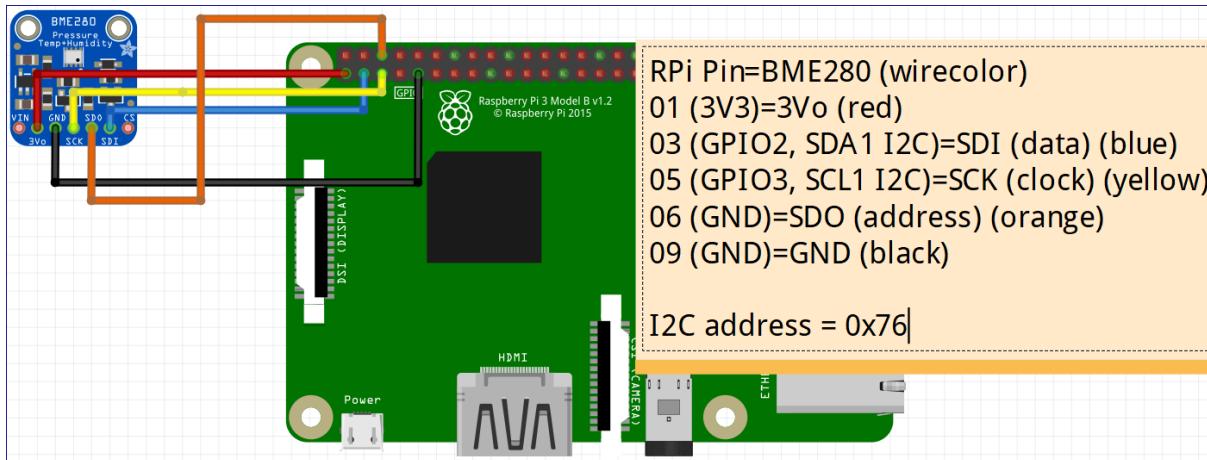
BME280	Raspberry Pi
3V	Pin #1 3V3
GND	Pin #9 GND
SDI Data	Pin #3 GPIO2, SDA1 I2C
SCL Clock	Pin #5 GPIO3, SCL1 I2C
SDO Address	Pin #6 GND
GND	Pin #9 GND

*Note*

The I2C address used is 0x76.

## Circuit

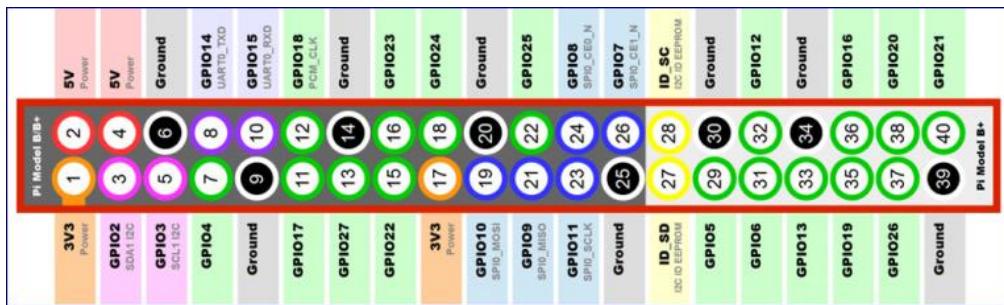
The BME280 sensor (Waveshare) connected to the Raspberry Pi:



### Notes

- By connecting SDO to Ground, the default I2C Address 0x76 is set.  
Use the terminal command: “i2cdetect -y 1” to determine the I2C address.
- Domoticz uses the default address 0x76.
- Adding multiple I2C sensors, requires a unique address for each sensor and two 1K resistors between VCC and the Data (SDA) & Clock (SCL) lines.

## Raspberry Pi B+ GPIO Header



## I2C

Check if the I2C module (`i2c_bcm2835`) is loaded using the terminal command “`lsmod`”.

If not, activate the I2C interface in the Raspberry Pi preferences.

Below list is only an extract of the various command outputs.

```
$lsmod
Module           Size  Used by
i2c_bcm2835      16384  0
spi_bcm2835      16384  0
w1_gpio          16384  0
wire              40960  1 w1_gpio
i2c_dev          16384  0
ip_tables        24576  0
x_tables          32768  1 ip_tables
ipv6             434176  58

$ls -l /dev/i2c-
crw-rw---- 1 root i2c 89, 1 Aug 30 12:08 /dev/i2c-1

$i2cdetect -l
i2c-1  i2c          bcm2835 I2C adapter
i2cdetect -y 1          I2C adapter
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -----
10: -----
20: -----
30: -----
40: -----
50: -----
60: -----
70: ----- 76 --
```

# Domoticz Configuration

## BME280 Hardware

### Installation

Domoticz GUI > Setup > Hardware > Select Type, give Name, set path I2C bus > Add



### Note

The path, i.e. /dev/i2c-1, to the I2C Bus is obtained by the terminal command:

```
$ls -l /dev/i2c-*
crw-rw---- 1 root i2c 89, 1 Aug 30 12:08 /dev/i2c-1
```

## Check if the hardware has been added to the hardware list

Name	Enabled	Type	Address	Port	Data Timeout
BME280	Yes	I2C sensor BME280 Temp+Hum+Baro	I2C	/dev/i2c-1	Disabled

## Check the Domoticz log after adding the hardware

```
2018-08-31 11:34:50.758 Status: I2C Start HW wif ID: 13 Name: I2C_BME280 Address: 0 Port: /dev/i2c-1
Invert:0
2018-08-31 11:34:50.758 Status: I2C_BME280: Worker started...
```

## Add new Device

Setup > Settings > Hardware/ Devices > Allow new Devices 5 minutes

A new device is added with IDX=115 (Setup > Devices).

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
115	BME280	0001	1	Server Room	Temp + Humidity + Baro	Weather Station	22.6 C, 53 %, 1009.7 hPa

Add the device, named BME280 and check the widgets.

TemperatureWidget	Weather Widget
-------------------	----------------

### Important

As mentioned earlier, the BME280 is not required anymore to be visible, so the device is hidden by changing the device name to “\$BME280 Server Raum”.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
115	BME280	0001	1	\$BME280 Server Raum	Temp + Humidity + Baro	Weather Station	20.9 C, 64 %, 1015.7 hPa

## Barometer Device

The barometer device gets the barometer value and forecast from the BME280.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
116	VirtualSensors	00082116	1	Luftdruck	General	Barometer	1015 hPa

## Temperature and Humidity Device

The temp+humdevice gets the temperature and humidity values from the BME280.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
351	VirtualSensors	141AF	1	Server Raum	Temp + Humidity	THGN122/123/132, THGR122/228/238/268	20.9 C, 65 %

## Automation Script

The automation event dzVents Lua updates, in regular intervals, the virtual sensors holding the data barometer, temperature & humidity.

Event name: bme280\_devices\_update

```
--[[  
bme280_devices_update.dzvents  
bme280 to update, triggered by timer, two vitualsensor devices Temp+Hum and Barometer.  
The BME280 is connected to the Domoticz Production system GPIO and is placed in the server room.  
Device trigger NOT USED because running rather often (n times/minute).  
]]--  
local IDX_BME280 = 115;  
local IDX_AIRPRESSURE = 116;  
local IDX_TEMPHUM = 351;  
local DEF_BME280_COMPENSATION = 0  
  
return {  
    on = {  
        timer = {'every 15 minutes'},  
    },  
    execute = function(domoticz, item)  
        -- Get the airpressure (and round), forecast  
        local barometer = domoticz.devices(IDX_BME280).barometer  
        barometer = math.floor(barometer) + DEF_BME280_COMPENSATION  
        local forecast = domoticz.devices(IDX_BME280).forecast -- Number,4  
        local forecastString = string.upper(domoticz.devices(IDX_BME280).forecastString) -- String,CLOUDY  
        -- Workaround for the forecast as the previous mapping did not work, resultig in unknown.  
        -- forecast = domoticz.devices(IDX_BME280).forecast  
        -- Forecast can be domoticz.BARO_STABLE, BARO_SUNNY, BARO_CLOUDY, BARO_UNSTABLE, BARO_THUNDERSTORM.  
        -- Barometer Forecast: 0 = No Info,1 = Sunny,2 = Paryly Cloudy,3 = Cloudy,4 = Rain  
        forecast = domoticz.BARO_STABLE  
        if (forecastString == 'CLOUDY') then forecast = domoticz.BARO_CLOUDY end  
        if (forecastString == 'CLOUDY/RAIN') then forecast = domoticz.BARO_CLOUDY end  
        if (forecastString == 'STABLE') then forecast = domoticz.BARO_STABLE end  
        if (forecastString == 'SUNNY') then forecast = domoticz.BARO_SUNNY end  
        if (forecastString == 'THUNDERSTORM') then forecast = domoticz.BARO_THUNDERSTORM end  
        if (forecastString == 'UNKNOWN') then forecast = domoticz.BARO_STABLE end  
        if (forecastString == 'UNSTABLE') then forecast = domoticz.BARO_UNSTABLE end  
        -- Update the virtual sensor  
        domoticz.devices(IDX_AIRPRESSURE).updateBarometer(barometer, forecast)  
        domoticz.log(string.format('Device %s updated:%.0f %s',  
            domoticz.devices(IDX_AIRPRESSURE).name, barometer, forecast), domoticz.LOG_INFO)  
  
        -- Get the temp and hum + status  
        local temperature = domoticz.devices(IDX_BME280).temperature;  
        local humidity = domoticz.devices(IDX_BME280).humidity;  
        local humiditystatusvalue = domoticz.devices(IDX_BME280).humidityStatusValue  
        -- Update the virtual sensor  
        domoticz.devices(IDX_TEMPHUM).updateTempHum(temperature, humidity, humiditystatusvalue)  
        domoticz.log(string.format('Device %s updated:%.0f %s %s',  
            domoticz.devices(IDX_TEMPHUM).name, temperature, humidity, humiditystatusvalue), domoticz.LOG_INFO)  
    end  
}
```

### Domoticz Log

2020-09-09 15:45:00.356 Status: dzVents: Info: ----- Start internal script: bme280_devices_update:, trigger: "every 15 minutes"
2020-09-09 15:45:00.372 Status: dzVents: Info: Device Luftdruck updated:1015 cloudy
2020-09-09 15:45:00.373 Status: dzVents: Info: Device Server Raum updated:21 65 3
2020-09-09 15:45:00.373 Status: dzVents: Info: ----- Finished bme280_devices_update

### Note

A timer trigger is used instead of a device trigger, because the BME280 updates rather often, up-to 3 times/minute (and even more), based on device data changes.

2020-09-09 15:45:57.754 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)
2020-09-09 15:46:27.809 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)

2020-09-09 15:46:57.862 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)
2020-09-09 15:47:27.914 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)
2020-09-09 15:47:57.971 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)
2020-09-09 15:48:28.022 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)
2020-09-09 15:48:58.074 (BME280) Temp + Humidity + Baro (\$BME280 Server Raum)

# Climate Environment (DWD Service)

## Purpose

To show daily

- the UV Index & UV Category for a given city and
- the pollen dispersal for selected pollen.

## Solution

This solution applies to Germany because it makes use of an external service provided by the [“Deutsche Wetterdienst”](#) (DWD).

The data is requested via HTTP Requests from the DWD services.

The HTTP Response is returned in JSON format, which is parsed to get the respective data to update Domoticz devices (Virtual Sensors).

Automation scripts dzVents Lua are handling the HTTP requests, parsing the HTTP response, and updating the Domoticz devices.

The DWD data is provided by the [Open Data Server](#).

The data used are the [UV Index](#) and [Pollen](#).

## UV Index

The UV Index is obtained by sending an HTTP request to the DWD service:

```
https://opendata.dwd.de/climate_environment/weather/alerts/uvi.json
```

with URL Response (extract for city “Hamburg” from the full JSON response)

```
{
    "next_update": "2020-07-30T10:00:00",
    "forecast_day": "2020-07-29",
    "sender": "Deutscher Wetterdienst - Medizin-Meteorologie",
    "last_update": "2020-07-29T10:00:00",
    "content": [
        {
            "forecast": {
                "dayafter_to": 7,
                "tomorrow": 6,
                "today": 5
            },
            "city": "Hamburg"
        }
    ],
    "name": "UV-Gefahrenindex"
}
```

### Notes

- The data is updated once a day at 12:00 (2 hrs after the value of key “next\_update”)
- The content keys “city” & “forecast” are used to update Domoticz devices

## Domoticz Configuration

Two Domoticz virtual devices are created and updated by an Automation Event dzVents Lua:

- Type UV to show the UV Index & UV Category for the city
- Type Alert to indicate the UV Index & UV Data

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
19	VirtualSensors	14063	1	UV Index Hamburg - moderate	UV	UVN128,UV138	5.0 UVI
20	VirtualSensors	00082020	1	UV Data Hamburg	General	Text	Today: 5 (moderate), Tomorrow: 6 (high), Day After: 7 (high)

Idx, Hardware, Name, Type, SubType, SwitchType, Data
19,VirtualSensors,UV Index,UV,UVN128,UV138,,Data
21,VirtualSensors,UV Alert,General,Alert,,Data



### Note

The name of the devices are updated by the Automation Script:

- UV Index amended with the City – UV Index Category
- UV Alert amended with the City

### Alert Device

The alert levels are mapped to the UV Index:

UV Index	Alert Level	
Not used	0 = gray	
0-2	1 = green	
3-5	2 = yellow	
6-7	3 = orange	
8-10	4 = red	Alert level 4 used for UV Index 8-10 and > 11
> 11	4 = red	

### User Variable

A user variable is used to set the city to obtain UV Index data.

Idx	Variable name	Variable type	Current value	Last update
1	DEF_UV_CITY	String	Hamburg	2020-07-29 12:02:11

### Automation Script

Source: dwd\_uvi.dzvents

(contains comments = stripped out below)

```
local IDX_UV_INDEX = 19
local IDX_UV_ALERT = 21
local UV_URL = 'https://opendata.dwd.de/climate_environment/weather/alerts/uvi.json';
local UV_RES = 'DWDUVINDEX';
local IDX_UV_CITY = 1
```

```

local DOM_URL = 'http://localhost:8080' -- /json.htm?type=setused&idx=%d&name=%s&used=true'

local function getUVCategories(uvindex)
    local uvcategories = "unknown"
    if uvindex >=0 and uvindex <=2 then uvcategories = "low" end
    if uvindex >=3 and uvindex <=5 then uvcategories = "moderate" end
    if uvindex >=6 and uvindex <=7 then uvcategories = "high" end
    if uvindex >=8 and uvindex <=10 then uvcategories = "very high" end
    if uvindex >=11 then uvcategories = "extreme" end
    return uvcategories
end

local function getUVAlert(uvindex)
    local uvalert = 0
    if uvindex >=0 and uvindex <=2 then uvalert = 1 end      -- green
    if uvindex >=3 and uvindex <=5 then uvalert = 2 end      -- yellow
    if uvindex >=6 and uvindex <=7 then uvalert = 3 end      -- orange
    if uvindex >=8 and uvindex <=10 then uvalert = 4 end     -- red
    if uvindex >=11 then uvalert = 4 end
    return uvalert
end

return {
    on = { timer = {'at 12:00'}, httpResponses = {UV_RES} },
    execute = function(domoticz, item)
        local uvcity = domoticz.variables(IDX_UV_CITY).value
        if (item.isTimer) then
            domoticz.openURL({ url = UV_URL, method = 'GET', callback = UV_RES, })
        end
        if (item.isHTTPResponse) then
            if (item.ok and item.callback == UV_RES) then
                if (item.isJSON) then
                    local uvdata = item.json.content
                    local cityfound = 0
                    for i,line in pairs(uvdata) do
                        local city = line.city
                        if city == uvcity then
                            cityfound = 1
                            local forecast = line.forecast
                            local uvindex = forecast.today
                            local uvcategories = getUVCategories(uvindex)
                            local uvdata = string.format("Today: %d (%s), Tomorrow: %d (%s)",
                                forecast.today, uvcategories, forecast.tomorrow, getUVCategories(forecast.tomorrow))
                            domoticz.devices(IDX_UV_INDEX).updateUV(uvindex)
                            domoticz.devices(IDX_UV_ALERT).updateAlertSensor(getUVAlert(uvindex), uvdata)
                            local newname = domoticz.utils.urlEncode(string.format("UV Index %s - %s", uvcity,
                                uvcategories))
                            local url = string.format("%s/json.htm?type=setused&idx=%d&name=%s&used=true",DOM_URL,
                                IDX_UV_INDEX, newname)
                            domoticz.openURL({url = url})
                            newname = domoticz.utils.urlEncode(string.format("UV Alert %s", uvcity))
                            url = string.format("%s/json.htm?type=setused&idx=%d&name=%s&used=true",DOM_URL,
                                IDX_UV_ALERT, newname)
                            domoticz.openURL({url = url}).afterSec(1)
                        end -- if city
                    end
                    if cityfound == 0 then
                        domoticz.log(string.format('UV data for %s not found.', UVI_CITY), domoticz.LOG_ERROR)
                    end
                end
            else
                domoticz.log(string.format('Problem handling the request: %s',UVI_URL), domoticz.LOG_ERROR)
            end
        end
    end
}
}

```

### Notes

Ensure in the settings (GUI > Setup > Settings > Local Networks (no username/password)), the local networks are defined to be able to openURL on localhost for renaming the devices.

**Local Networks (no username/password):**

Networks: 127.0.0.\*;192.168.1.\*  
(Separate by a semicolon, For Example: 127.0.0.\*;192.168.0.\*)

**Domoticz Log**

```
2020-07-30 12:00:00.168 Status: dzVents: Info: ----- Start internal script: dwd_uvi:, trigger: "at 12:00"
2020-07-30 12:00:00.168 Status: dzVents: Info: ----- Finished dwd_uvi
2020-07-30 12:00:00.168 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-07-30 12:00:00.418 Status: dzVents: Info: Handling httpResponse-events for: "DWDUVINDEX"
2020-07-30 12:00:00.418 Status: dzVents: Info: ----- Start internal script: dwd_uvi: HTTPResponse: "DWDUVINDEX"
2020-07-30 12:00:00.423 Status: dzVents: Info: UVI Hamburg: 5 = moderate
2020-07-30 12:00:00.423 Status: dzVents: Info: Today: 5 (moderate), Tomorrow: 7 (high)
2020-07-30 12:00:00.435 Status: dzVents: Info: ----- Finished dwd_uvi
```

## Pollen

The Pollen information is obtained by sending an HTTP request to the DWD service:

```
https://opendata.dwd.de/climate_environment/weather/alerts/s31fg.json
```

with URL Response (extract for region\_name": "Schleswig-Holstein und Hamburg" from the full JSON response)

```
{
  "next_update": "2020-07-29 11:00 Uhr",
  "name": "Pollenflug-Gefahrenindex fÃ¼r Deutschland ausgegeben vom Deutschen Wetterdienst",
  "sender": "Deutscher Wetterdienst - Medizin-Meteorologie",
  "legend": {
    "id1": "0", "id1_desc": "keine Belastung",
    "id2": "0-1", "id2_desc": "keine bis geringe Belastung",
    "id3": "1", "id3_desc": "geringe Belastung",
    "id4": "1-2", "id4_desc": "geringe bis mittlere Belastung"
    "id5": "2", "id5_desc": "mittlere Belastung",
    "id6": "2-3", "id6_desc": "mittlere bis hohe Belastung",
    "id7": "3", "id7_desc": "hohe Belastung",
  },
  "last_update": "2020-07-28 11:00 Uhr",
  "content": [
    {
      "partregion_name": "Geest, Schleswig-Holstein und Hamburg",
      "region_name": "Schleswig-Holstein und Hamburg",
      "Pollen": {
        "Ambrosia": {"today": "0", "dayafter_to": "-1", "tomorrow": "0"},
        "Beifuss": {"today": "1", "tomorrow": "1", "dayafter_to": "-1"},
        "Birke": {"dayafter_to": "-1", "tomorrow": "0", "today": "0"},
        "Erle": {"tomorrow": "0", "dayafter_to": "-1", "today": "0"},
        "Esche": {"dayafter_to": "-1", "tomorrow": "0", "today": "0"},
        "Graeser": {"tomorrow": "1-2", "dayafter_to": "-1", "today": "1-2"},
        "Hasel": {"today": "0", "tomorrow": "0", "dayafter_to": "-1"},
        "Roggen": {"tomorrow": "0", "dayafter_to": "-1", "today": "0" }
      },
      "region_id": 10,
      "partregion_id": 12
    }
  ]
}
```

### Notes

- The data is updated once a day at 12:00 (1 hr after the value key next\_update)
- The content keys “region\_name” & “Pollen” are used to update Domoticz devices

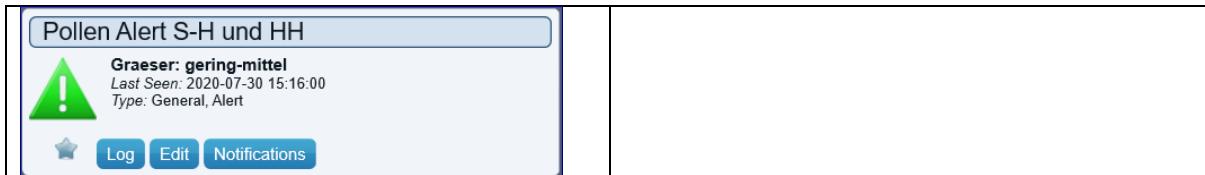
## Domoticz Configuration

A Domoticz virtual device is created and updated by an Automation Event dzVents Lua:

- Type Alert to indicate the region & pollen alerts

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
22	VirtualSensors	82022	1	Pollen Alert S-H und HH	General	Alert	Graeser: gering-mittel

Idx, Hardware, Name, Type, SubType, SwitchType, Data
22, VirtualSensors, Pollen Alert, General, Alert,, Data



### Note

The name of the device is updated by the Automation Script with the key partregion\_name (truncated by using a user variable DEF\_POLLEN\_PARTREGION\_SHORT else does not fit on the GUI widget title).

The alert text is listing all pollen which have an index 1-2 or higher. The pollen description is truncated to ensure the max string length of 200 characters for the text is not exceeded.

### Alert Device

The alert levels are mapped to the Pollen Index:

UV Index	Alert Level
0	0 = gray
1-2	1 = green
2	2 = yellow
2-3	3 = orange
3	4 = red

### User Variable

Two user variables are used to for the part region to obtain pollen data.

Idx	Variable name	Variable type	Current value
3	DEF_POLLEN_PARTREGION_SHORT	String	S-H und HH
2	DEF_POLLEN_PARTREGION_NAME	String	Geest, Schleswig-Holstein und Hamburg

### Automation Script

Source: dwd\_pollen.dzvents

(contains comments = stripped out below)

```

local IDX_POLLEN_ALERT = 22
local POLLEN_URL = 'https://opendata.dwd.de/climate_environment/weather/alerts/s31fg.json';
local POLLEN_RES = 'DWDPOLLEN';
local IDX_POLLEN_PARTREGION_NAME = 2
local IDX_POLLEN_PARTREGION_SHORT = 3
local DOM_URL = 'http://localhost:8080' -- /json.htm?type=setused&idx=%d&name=%s&used=true'

local function getPollenIndexDesc(id)
    local iddesc = "nicht bekannt"
    if id == "0" then iddesc = "kein" end
    if id == "0-1" then iddesc = "kein-gering" end
    if id == "1" then iddesc = "gering" end
    if id == "1-2" then iddesc = "gering-mittel" end
    if id == "2" then iddesc = "mittel" end
    if id == "2-3" then iddesc = "mittel-hoch" end
    if id == "3" then iddesc = "hoch" end
    return iddesc
end

local function setPollenAlertLevel(id, oldlevel)
    local newlevel = oldlevel
    local level = 0
    if id == "1-2" then level = 1 end
    if id == "2" then level = 2 end
    if id == "2-3" then level = 3 end
    -- gray
    -- green
    -- yellow
    -- orange
end

```

```

if id == "3" then level = 4 end          -- red
if level > oldlevel then newlevel = level end
return level
end

return {
on = { timer = {'at 12:00'},httpResponses = {POLLEN_RES} },
execute = function(domoticz, item)
local partregion = domoticz.variables(IDX_POLLEN_PARTREGION_NAME).value
local partregionshort = domoticz.variables(IDX_POLLEN_PARTREGION_SHORT).value

if (item.isTimer) then
    domoticz.openURL({ url = POLLEN_URL, method = 'GET', callback = POLLEN_RES, })
end
if (item.isHTTPResponse) then
    if (item.ok and item.callback == POLLEN_RES) then
        if (item.isJSON) then
            local content = item.json.content
            local partregionfound = 0
            for i,line in pairs(content) do
                local partregionname = line.partregion_name
                if partregionname == partregion then
                    partregionfound = 1
                    local pollenlist = line.Pollen
                    local pollenalertlevel = 0  -- no alert
                    local pollenalertinfo = ""
                    local count = 0
                    for k, v in next, pollenlist do
                        if v.today == "1-2" or v.today == "2" or v.today == "2-3" or v.today == "3" then
                            count = count + 1
                            if count > 1 then
                                pollenalertinfo = pollenalertinfo .. ", "
                            end
                            pollenalertinfo = pollenalertinfo .. string.format("%s: %s", k,
getPollenIndexDesc(v.today))
                            pollenalertlevel = setPollenAlertLevel(v.today, pollenalertlevel)
                        end
                    end
                    if count == 0 then pollenalertinfo = "keine Belastung" end
                    domoticz.devices(IDX_POLLEN_ALERT).updateAlertSensor(pollenalertlevel, pollenalertinfo)
                    local newname = domoticz.utils.urlEncode(string.format("Pollen Alert %s", partregionshort))
                    local url = string.format("%s/json.htm?type=setused&idx=%d&name=%s&used=true",DOM_URL,
IDX_POLLEN_ALERT, newname)
                    domoticz.openURL({url = url}).afterSec(1)
                end -- if regionfound
            end
            if partregionfound == 0 then
                domoticz.log(string.format('Pollen data for %s not found.', partregion), domoticz.LOG_ERROR)
            end
        end
    else
        domoticz.log(string.format('Problem handling the request: %s',POLLEN_URL), domoticz.LOG_ERROR)
    end
end
end
}

```

## Domoticz Log

Showing testlog triggered every minute instead at 12:00

```
2020-07-30 15:32:00.569 Status: dzVents: Info: ----- Start internal script: dwi_pollen:, trigger:  
"every minute"  
2020-07-30 15:32:00.570 Status: dzVents: Info: ----- Finished dwi_pollen  
2020-07-30 15:32:00.570 Status: EventSystem: Script event triggered:  
/home/pi/domoticz/dzVents/runtime/dzVents.lua  
2020-07-30 15:32:00.834 Status: dzVents: Info: Handling httpResponse-events for: "DWDPOLLEN"  
2020-07-30 15:32:00.834 Status: dzVents: Info: ----- Start internal script: dwi_pollen: HTTPResponse:  
"DWDPOLLEN"  
2020-07-30 15:32:00.853 Status: dzVents: Info: Partregion: Geest, Schleswig-Holstein und Hamburg  
2020-07-30 15:32:00.853 Status: dzVents: Info: Graeser: gering-mittel  
2020-07-30 15:32:00.864 Status: dzVents: Info: ----- Finished dwi_pollen  
2020-07-30 15:32:00.865 Status: EventSystem: Script event triggered:  
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```



## Modified Solution

The previous shared initial solution has been created on the Domoticz Development System. The next, slightly modified, solution runs on the Domoticz Production System with two virtual sensor devices (both type Alert) and three user variables.

The usage of two Alert devices for UV Index and Pollen is easier to read out.

The content contains the date (format DD-MM) of the information as provided by the DWD service. The data is updated once per day at 1200.

A roomplan has been defined for quick dashboard access.

The data is in German.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
▲ 323	VirtualSensors	82323	1	Pollen	General	Alert	24-08: Beifuss: Gering-Mittel
▲ 322	VirtualSensors	82322	1	UV Index	General	Alert	24-08: Gering, 25-08: Mittel

Idx	Variable name	Variable type	Current value
20	DEF_POLLEN_PARTREGION_SHORT	String	S-H und HH
19	DEF_POLLEN_PARTREGION_NAME	String	Geest, Schleswig-Holstein und Hamburg
18	DEF_UV_CITY	String	Hamburg

Roomplan	Dashboard
<p>The Roomplan interface displays two sections: 'Room plans' and 'Devices'. The 'Room plans' section shows a single entry for 'Climate'. The 'Devices' section lists two virtual sensors: 'UV Index' and 'Pollen'.</p>	<p>The Dashboard interface shows a summary for 2020-08-24 18:28:31 with current values: S-H und HH for Pollen, Geest, Schleswig-Holstein und Hamburg for Pollen, and Hamburg for UV Index. Below this, a 'Utility Sensors' section lists 'UV Index' and 'Pollen' with their respective current values: 24-08: Gering, 25-08: Mittel for UV Index and 24-08: Beifuss: Gering-Mittel for Pollen.</p>

## Automation Scripts

These events are enhanced based on the previous described solution.

See the source codes “dwd\_uvi\_v2.dzvents” and “dwd\_pollen\_v2.dzvents”.

# Coffee Machine Monitor

## Purpose

To monitor, the start & end time of the Coffee Machine running early morning.

If the coffee machine is tuned on, an [Alert Message](#) is set.

The Coffee Machine switches OFF after two hours.

## Solution

Monitor the value of the Electric Usage House Device (IDX=171, ).

If it is above a threshold (defined as User Variable, Integer) between timeframe.

If above threshold, then set the Alert Message.

Monitoring is handled by a dzVents Lua script event for device changes and timer.

The script uses a script level persistent variable “notified” to handle the Alert Message notifications (see dzvents script).

The threshold is set by a User Variable (i.e. 1100 Wh):

Idx	Variable name	Variable type	Current value
15	TH_COFFEEMACHINEMONITOR	Integer	1100

### Notes

1. The actual Wh is determined from the House Power Measurement. If another device is turned on prior coffee machine resulting in power usage above threshold, then the Coffee Machine Alert Message is sent – which is misleading.  
A solution, is to connect a power plug to the Coffee Machine (see [Electric Usage Rooms/Devices](#)).
2. The alertlevel orange (value 3) is used. if the uservar “th\_alerttoemail” is set to 3 then an email is sent if the electricusage threshold is reached (see dzVents Script).

# Automation Script

Event name: coffee\_machine\_monitor

```
-- Idx of the device kWh
local IDX_ELECTRICUSAGEHOUSE = 171
-- Idx of the user variable for the threshold (as integer)
local IDX_TH_COFFEE MACHINE MONITOR = 15
-- Monitor the whactual between 7 and 9 if above threshold
local MONITOR_START_END = 'at 07:00-09:00'

return {
  on = {
    -- monitor the whactual between start end time
    timer = {
      MONITOR_START_END
    },
  },
  data = {
    -- set a flag if already notified within timeframe
    notified = { initial = 0 }
  },
  execute = function(domoticz)
    -- Check if below threshold and notified flag set
    -- Action: reset notified flag
    if (domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhActual <
domoticz.variables(IDX_TH_COFFEE MACHINE MONITOR).value) and (domoticz.data.notified == 1)
    then
      domoticz.data.notified = 0
    end

    -- Check if above threshold and notified flag not set
    -- Action: set notified flag and alert message
    if (domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhActual >
domoticz.variables(IDX_TH_COFFEE MACHINE MONITOR).value) and (domoticz.data.notified == 0)
    then
      -- set notified flag
      domoticz.data.notified = 1
      -- get the time on and log
      timeon = os.date("*t")
      message = ('Coffee Machine On %02d:%02d'):format(timeon.hour, timeon.min)
      domoticz.log(message, domoticz.LOG_INFO)
      -- calculate the time off = timeon + 2 hours
      timeoff = os.date("*t", os.time() + 2*60*60)
      -- define message to notify the time coffee machine switches off
      message = ('Coffee Machine Off %02d:%02d'):format(timeoff.hour, timeoff.min)
      -- update alert message
      domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_ORANGE, message)
      domoticz.log(message, domoticz.LOG_INFO)
    end
  end
}
```

# Contact Detection (433 Mhz)

## Purpose

To monitor opening of doors/windows (or other like monitoring the postbox) and trigger a “contact open” notification action.

## Solution Sensor PB-62R

This solution makes use of the wireless 433Mhz contact detector PB-62R (from KOBERT-GOODS) connected to RFXCOM RFxTrx433E.

Tested two devices:

PB-62R	PB-67R
<p>From: KOBERT-GOODS            Costs: ~10EUR            Battery: 12V (MN21/23) with duration ~4 month.</p>  <ul style="list-style-type: none"> <li>Left = Magnet</li> <li>Right = Detector with               <ul style="list-style-type: none"> <li>top left red LED (lights up if the contact is “broken”),</li> <li>Middle right reset button used for Domoticz to learn the device.</li> </ul> </li> </ul>	<p>(Status 20190521)  <b>Tested but not supported by RFXCOM RFxTrx433E.</b>            The RFXCOM Manager recognized the switch as:            Packettype = Lighting5            ERROR: Unknown Sub type for Packet type=14: 11            Signal level = 7</p>
<p><b>Ways-of-Working</b>            If the state of the contact switch changes to “Alarm + Tamper” and a notification is sent to inform.            If the door is closed again, the contact switch does not send a change state, which means a manual or timer trigger is required to reset the state to “Normal”.</p>	<p>&lt;TODO&gt;            Watch for RFXCOM updates and check if this device is supported.</p>

## Domoticz Configuration

Learn the new device:

Domoticz GUI > Setup > Settings > Hardware/Devices > Allow for 5 minutes

Press and hold for a few seconds the reset button on the device.

Check the devices list for the new device:

Domoticz GUI > Setup > Devices > Scroll down to the last list entry.

A new device is listed:

Idx,Hardware,Name,Type,SubType,Data
160,RFXtrx433e,Unknown,Security,X10 security,Alarm + Tamper

*Note*

After adding the device, set the name to: Security Main Door.

## Alarm Detection

 <p>Security Main Door      <b>Alarm + Tamper</b></p> <p>Last Seen: 2019-05-20 09:43:46 Type: Security, X10 security, Security</p> <p>Log Edit Notifications</p>	<p>If door is opened, the alarm state is set to <b>Alarm + Tamper</b>. The state is not reset by the device itself and requires a trigger, i.e. Button or dzVents timer script!</p>
 <p>Security Main Door      <b>Normal</b></p> <p>Last Seen: 2019-05-20 09:36:00 Type: Security, X10 security, Security</p> <p>Log Edit Notifications</p>	<p>State reset to Normal via HTTP API Request:  <a href="http://domoticz-ip:8080/json.htm?type=command&amp;param=switchlight&amp;idx=160&amp;switchcmd=Normal">http://domoticz-ip:8080/json.htm?type=command&amp;param=switchlight&amp;idx=160&amp;switchcmd=Normal</a></p>

# Automation Script

Solution to reset all PB-62R devices (applies to X10 switches with state Alarm + Tamper) to state **Normal** by using a dzVents Lua script event.

If a device state changes to **Alarm + Tamper**, notify email and set user variable DEF\_SECURITY\_ALARM to monitor state to 1.

Reset, via timer every 5 minutes, the state of the devices defined in the array SECURITYDEVICES and set the user variable to 0.

Event name: security\_monitor

```
-- Array holding the idx for the security devices:security main door (160), ...
local SECURITYDEVICES = {160}
-- User variable holding the state of the alarm (0=Normal, 1=Alarm)
local IDX_DEF_SECURITY_ALARM = 13

return {
  on = {
    devices = SECURITYDEVICES
    ,
    timer = {'Every 5 minutes'}
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
      if (item.state ~= 'Normal') then
        domoticz.email('SECURITY ALERT atHome', item.name .. ' state ' .. item.state, , 'email-address')
        domoticz.variables(IDX_DEF_SECURITY_ALARM).set(1)
      end
    end

    if (item.isTimer) then
      if (domoticz.variables(IDX_DEF_SECURITY_ALARM).value == 1) then
        domoticz.variables(IDX_DEF_SECURITY_ALARM).set(0)
        for i, idx in ipairs(SECURITYDEVICES) do
          domoticz.devices(idx).setState('Normal')
        end
      end
    end
  end
}
```

## User Variable

Idx	Variable name	Variable type	Current value
13	DEF_SECURITY_ALARM	Integer	0

## Alarm State Reset Switch

Solution to reset all security devices (mainly X10) to state Normal by using a switch which triggers a dzVents Lua script event “security\_reset”.

Create Switch Device (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
162, VirtualSensors, Security Reset, Light/Switch, Switch, Push On Button, On

Create Automation Event

Event name: security\_reset

```
-- security_reset.lua
-- The push-button to reset the contacts
local IDX_SECURITY_RESET = 162
-- Array of the idx for the security devices: security front door, ...
local securitydevices = {160}
return {
  on = {
    devices = {IDX_SECURITY_RESET}
  },
  execute = function(domoticz, device)
    domoticz.log('SECURITY RESET: ' .. device.name, domoticz.LOG_INFO)
    -- Loop over the array
    for i, idx in ipairs(securitydevices) do
      domoticz.devices(idx).setState('Normal')
    end
  end
}
```

## MQTT

Explored MQTT payload when the X10 security device state changes.

### Alarm State Change to "Alarm + Tamper"

#### MQTT Result

```
{
  "Battery" : 100,   "RSSI" : 5,    "description" : "",    "dtype" : "Security",   "id" : "5C0700",
  "idx" : 160,     "name" : "Main Door State",   "nvalue" : 130,    "stype" : "X10 security",   "switchType"
  : "On/Off",      "unit" : 0}
```

#### Note

If the state is in Alarm, the nvalue is greater 0, in this case 130 for state "Alarm + Tamper".  
The MQTT JSON string shows the "switchType" : "On/Off", means the switchlight command/param can be used.

#### Domoticz Log

```
2019-05-18 11:17:04.661 (RFXtrx433e) Security (Main Door State)
2019-05-18 11:17:04.816 Status: dzVents: Info: Handling events for: "Main Door State", value: "Alarm +
Tamper"
2019-05-18 11:17:04.816 Status: dzVents: Info: ----- Start internal script: event_monitor: Device:
"Main Door State (RFXtrx433e)", Index: 160
2019-05-18 11:17:04.817 Status: dzVents: Info: ----- Finished event_monitor
2019-05-18 11:17:04.818 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

### Alarm State Reset to "Normal"

#### HTTP API Request Test

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=160&switchcmd=Normal
```

#### HTTP JSON Result

```
{"status" : "OK","title" : "SwitchLight"}
```

#### MQTT JSON Result

```
{
  "Battery" : 100,   "RSSI" : 12,    "description" : "",    "dtype" : "Security",   "id" : "5C0700",
  "idx" : 160,     "name" : "Main Door State",   "nvalue" : 0,    "stype" : "X10 security",   "switchType"
  : "On/Off",      "unit" : 0}
```

#### Note

If the state is Normal (so not in Alarm), the nvalue is 0.

#### Domoticz Log

```
2019-05-18 11:14:43.559 (RFXtrx433e) Security (Main Door State)
2019-05-18 11:14:43.552 Status: User: Admin initiated a switch command (160/ Main Door State/Normal)
2019-05-18 11:14:43.711 Status: dzVents: Info: Handling events for: " Main Door State", value: "Normal"
2019-05-18 11:14:43.711 Status: dzVents: Info: ----- Start internal script: event_monitor: Device:
"Main Door State (RFXtrx433e)", Index: 160
2019-05-18 11:14:43.712 Status: dzVents: Info: ----- Finished event_monitor
2019-05-18 11:14:43.713 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

## Another HTTP API Request Test

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=160&switchcmd=Alarm
```

The MQTT nvalue is 2.

*Note*

For the state "Alarm + Tamper" the nvalue is 130.

# Custom Settings

## Purpose

To (quickly) change selective user variables via a dedicated custom web page instead using the Domoticz default GUI > Setup > More Options > User variables.

## Solution

Custom webpage "Settings.html" with edit fields & set button to change the value of the user variable.

### Screenshot

GUI Custom Dropdown Menu entries and the Custom Page Settings with three settings. The field Name is read-only.

Custom Settings

**Battery Level Check Threshold**

Name: TH\_BATTERY\_LEVEL

Value: 20 Set

(%, Min = 0, Max = 100, 0 = Disabled)

**RaspberryMatic DutyCycle Threshold**

Name: TH\_RASPBERRYMATIC\_DUTYCYCLE

Value: 10 Set

(%, Min = 0, Max = 20, 0 = Disabled)

**Thermostat Default Temperature On**

Name: DEF\_THERMOSTAT\_ON

Value: 23 Set

(°C, Min = 6, Max = 23)

20201024 by rwbl

### Note

The page layout is inspired by the Domoticz page:

[~domoticz/www/views/uservariables.html](http://domoticz/www/views/uservariables.html)

## Installation

Copy the file "Settings.html" to following Domoticz folder:

```
~domoticz/www/templates
```

The Domoticz GUI > Tab Custom shows the new custom menu entry “Settings” in the dropdown list:



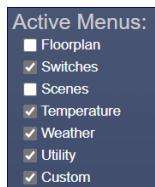
Direct page access from a browser:

```
http://domoticz-ip:port/#/Custom/Settings
```

*Note*

Ensure to enable the Domoticz GUI Custom Menu (checkmark the entry Custom):

```
GUI > Setup > Settings > System > Active Menus > Custom
```



## HTML File

Source: Settings.html

The source is well documented and outlines how to add a new user variable.

```
<!--
domoticz-homeautomation-workbook function Custom Settings
Domoticz custom page:
Settings.html (/home/pi/domoticz/www/templates)
Purpose:
To change the value of selected user variables, i.e. used as custom settings
Any other settings, to be set via Domoticz GUI Setup > Settings (the standard way).
20201024 rwbl
-->
<style>
/* hint style text small */
.hint {
  font-size: smaller;
  color: white;
}
</style>

<!--
HTML
-->
<div class="container">
  <table class="page-header-small" border="0" cellpadding="0" cellspacing="0" width="100%">
    <tr>
      <td align="left">
        <h4 data-i18n="Custom Settings"></h4>
      </td>
    </tr>
  </table>
<!--
  Each user variable consists out of a table with 4 fields variablename (readonly), variablevalue
  (mandatory), variableidx (readonly), variablename (readonly).
  The id of the table must set and unique, i.e. table id="raspmaticdutycycle".
  The field data is initially set when loading/refeshing the page.
  Any new user variables copy & paste the first entry <table id=...>...</table> and change the id.
  Changes required:
  * Set the table id, i.e. batterylevelcheck
  * Set the selector in function updateUserVariable(), i.e. updateUserVariable('#batterylevelcheck');
  * Set the input type, i.e. type="number"
  * Set the min/max values, i.e. min="0", max="100"
  * Amend the document load function getUserVariable('#uservariable', uservariableidx), i.e.
getUserVariable('#batterylevelcheck', 24);
-->

<!--
  Battery Level Check (idx=24)
-->
<table id="batterylevelcheck" class="display" border="0" cellpadding="0" cellspacing="20">
  <tr>
    <td align="right" style="width:110px"></td>
    <td><h3 data-i18n="Battery Level Check Threshold"></h3></td>
  </tr>
  <tr>
    <td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span></label></td>
    <td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
  </tr>
  <tr>
    <td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span></label></td>
    <td>
      <input id="variablevalue" type="number" min="0" max="100" style="width: 50px; padding: .2em;" class="text ui-widget-content ui-corner-all" required />
      <a align="right" class="btnstyle3" onclick="updateUserVariable('#batterylevelcheck');" data-
i18n="Set"></a>
    </td>
  </tr>
  <tr>
    <td align="right" style="width:110px"></td>
```

```

        <td class="hint">(% Min = 0, Max = 100, 0 = Disabled)</td>
    </tr>
    <!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden -->
    <tr hidden>
        <td align="right" style="width:110px"><label for="variableidx"><span data-
i18n="Idx"></span>:</label></td>
        <td><input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;">
<span data-
i18n="Type"></span>:</label></td>
        <td><input id="variabletype" type="number" style="width: 50px; padding: .2em; " class="text ui-
widget-content ui-corner-all" readonly /></td>
        <!-- 0=Integer,1:Float,2=String3=Date4=Time5=DateTime -->
    </tr>
    <tr hidden>
        <td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span>:</label></td>
        <td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
    </tr>
    <tr>
        <td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span>:</label></td>
        <td>
            <input id="variablevalue" type="number" min="0" max="20" style="width: 50px; padding: .2em;">
<span data-
i18n="Type"></span>:</label></td>
            <a class="btnstyle3" onclick="updateUserVariable('#raspmaticdutycycle');" data-i18n="Set"></a>
        </td>
    </tr>
    <tr>
        <td align="right" style="width:110px"></td>
        <td class="hint">(% Min = 0, Max = 20, 0 = Disabled)</td>
    </tr>
    <!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden -->
    <tr hidden>
        <td align="right" style="width:110px"><label for="variableidx"><span data-
i18n="Idx"></span>:</label></td>
        <td><input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;">
<span data-
i18n="Type"></span>:</label></td>
        <td><input id="variabletype" type="number" style="width: 50px; padding: .2em; " class="text ui-
widget-content ui-corner-all" readonly /></td>
        <!-- 0=Integer,1:Float,2=String3=Date4=Time5=DateTime -->
    </tr>
    </table>
    <br>

    <!--
        Thermostat Temperature On (idx=21)
    -->
    <table id="thermostaton" class="display" border="0" cellpadding="0" cellspacing="20">
        <tr>
            <td align="right" style="width:110px"></td>
            <td><h3 data-i18n="Thermostat Default Temperature On"></h3></td>
        </tr>
        <tr>

```

```

<td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span></label></td>
<td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
</tr>
<tr>
<td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span></label></td>
<td>
<input id="variablevalue" type="number" min="6" max="23" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" required />
<a class="btnstyle3" onclick="updateUserVariable('#thermostaton');" data-i18n="Set"></a>
</td>
</tr>
<tr>
<td align="right" style="width:110px"></td>
<td class="hint">(°C, Min = 6, Max = 23)</td>
</tr>
<!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden --&gt;
&lt;tr hidden&gt;
&lt;td align="right" style="width:110px"&gt;&lt;label for="variableidx"&gt;&lt;span data-
i18n="Idx"&gt;&lt;/span&gt;&lt;/label&gt;&lt;/td&gt;
&lt;td&gt;&lt;input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" readonly /&gt;&lt;/td&gt;
&lt;/tr&gt;
&lt;tr hidden&gt;
&lt;td align="right" style="width:110px"&gt;&lt;label for="variabletype"&gt;&lt;span data-
i18n="Type"&gt;&lt;/span&gt;&lt;/label&gt;&lt;/td&gt;
&lt;td&gt;&lt;input id="variabletype" type="number" style="width: 50px; padding: .2em;" 
class="text ui-
widget-content ui-corner-all" readonly /&gt;&lt;/td&gt;
<!-- 0=Integer,1:Float,2=String3=DateTime4=DateTime --&gt;
&lt;/tr&gt;
&lt;/table&gt;
&lt;br&gt;

&lt!--
  Add More
--&gt;

&lt!-- Version info --&gt;
&lt;div id="pageversion" class="hint"&gt;&lt;/div&gt;
&lt;/div&gt;

&lt;!--
  JavaScript
--&gt;
&lt;script type="text/javascript" charset="utf-8"&gt;

// Update the value of a user variable
// Get UV Data: json.htm?type=command&amp;param=getuservariable&amp;idx=IDX
// Update UV Value:
json.htm?type=command&amp;param=updateuservariable&amp;vname=USERVARIABLENAME&amp;vtype=USERVARIABLETYPE&amp;vvalue=USE
RVARIABLEVALUE
function updateUserVariable(variableSelector) {
  var uvIdx = $(variableSelector + ' ' + '#variableidx').val();
  var uvName = $(variableSelector + ' ' + '#variablename').val();
  var uvType = $(variableSelector + ' ' + '#variabletype').val();
  var uvValue = $(variableSelector + ' ' + '#variablevalue').val();
  var requestUrl = "json.htm?type=command&amp;param=updateuservariable&amp;idx=" + uvIdx + "&amp;vname=" + uvName
+ "&amp;vtype=" + uvType + "&amp;vvalue=" + uvValue;
  console.log("updateUserVariable - URL: " + requestUrl);
  $.ajax({
    url: requestUrl,
    async: false,
    dataType: 'json',
    success: function (data) {
      if (typeof data != 'undefined') {
        if (data.status == "OK") {
          bootbox.alert($.t('User variable ' + uvName + ' updated.'));
        }
        else {
          ShowNotify($.t(data.message), 2500, true);
        }
      }
    }
  });
}

// Update the value of a user variable
// Get UV Data: json.htm?type=command&amp;param=getuservariable&amp;idx=IDX
// Update UV Value:
json.htm?type=command&amp;param=updateuservariable&amp;vname=USERVARIABLENAME&amp;vtype=USERVARIABLETYPE&amp;vvalue=USE
RVARIABLEVALUE
function updateUserVariable(variableSelector) {
  var uvIdx = $(variableSelector + ' ' + '#variableidx').val();
  var uvName = $(variableSelector + ' ' + '#variablename').val();
  var uvType = $(variableSelector + ' ' + '#variabletype').val();
  var uvValue = $(variableSelector + ' ' + '#variablevalue').val();
  var requestUrl = "json.htm?type=command&amp;param=updateuservariable&amp;idx=" + uvIdx + "&amp;vname=" + uvName
+ "&amp;vtype=" + uvType + "&amp;vvalue=" + uvValue;
  console.log("updateUserVariable - URL: " + requestUrl);
  $.ajax({
    url: requestUrl,
    async: false,
    dataType: 'json',
    success: function (data) {
      if (typeof data != 'undefined') {
        if (data.status == "OK") {
          bootbox.alert($.t('User variable ' + uvName + ' updated.'));
        }
        else {
          ShowNotify($.t(data.message), 2500, true);
        }
      }
    }
  });
}
</pre>

```

```

    },
    error: function () {
        ShowNotify($.t('Problem saving user variable!'), 2500, true);
    }
});

// Get the user variable properties
// Get UV Data: json.htm?type=command&param=getuservariable&idx=IDX
// Parameter:
// variableSelector (String): #raspmaticdutycycle
// uvIdx (Integer): 23
function getUserVariable(variableSelector, uvIdx) {
    // var uvIdx = $(variableSelector + ' ' + '#variableidx').val();
    // Define the http api url to get uservariable properties name and type used to set the new value
    var requestUrl = "json.htm?type=command&param=getuservariable&idx=" + uvIdx;
    console.log("setUserVariable - Get data: " + requestUrl);
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            if(typeof data.result == "undefined"){
                console.log("setUserVariable - no data found (status: " + data.status + ", title: " +
data.title + ")");
                return false;
            }
            // console.log("setUserVariable - Succesfully retrieved (status: " + data.status + ", title: " +
data.title + ")");
            // Get the uservariable name and type
            var uvName = data.result[0].Name;
            var uvType = data.result[0].Type;
            var uvValue = data.result[0].Value;
            console.log("getUserVariable Idx,Name,Type,Value: " + uvIdx + "," + uvName + "," + uvType + "," +
uvValue);
            // Update the user value properties
            $(variableSelector + ' ' + '#variablename').val(uvName);
            $(variableSelector + ' ' + '#variablevalue').val(uvValue);
            $(variableSelector + ' ' + '#variableidx').val(uvIdx);
            $(variableSelector + ' ' + '#variabletype').val(uvType);
        },
        error: function () {
            console.log("[ERROR] Status all devices. Can not communicate to the Domoticz server!");
            return false;
        }
    });
}

// Sleep for ms
function sleep(milliseconds) {
    const date = Date.now();
    let currentDate = null;
    do {
        currentDate = Date.now();
    } while (currentDate - date < milliseconds);
}

function setVersion(pageVersion) {
    $('#pageversion').html(pageVersion);
}

// Load the document
$(document).ready(function(){
    setVersion("20201024 by rwbl");
    // Get the user variable data with selector, idx.
    // Short sleep after each uservariable to ensure HTTP api request gets completed
    getUserVariable('#batterycheck', 24);
    sleep(50);
    getUserVariable('#raspmaticdutycycle', 23);
    sleep(50);
    getUserVariable('#thermostaton', 21);
    sleep(50);
});
</script>

```

## Enhancements

Use HTML elements like checkboxes, buttons to set user variables type integer with value 0 (=OFF) and 1(=ON). This concept would avoid creating virtual devices.

# Days-To-Go

## Purpose

To calculate, once a day around midnight, the days-to-go for several Virtual Devices.  
Beside days-to-go, days-left are also calculated depending date given.  
Update the Control Message after updating the devices.

### Widget Sample



#### Note

Changed the name and the days calculated use different dates then the script below.

## Solution

Run a dzVents Lua script event “days\_to\_go\_update”, once a day at 00:15, to calculate the date difference in days and update the devices.

The dzVents Lua script event is written using the Domoticz Event Editor (GUI > Setup > More Options > Events).

# Atomation Script

Event name: days\_to\_go\_update  
 (code stripped)

```
-- Idx of the devices
local IDXDAYSX = 29;
local IDXDAYSY = 57;
local IDXDAYSZ = 97;
local IDXCONTROLMSG = 52;

-- Calculate the date difference in days between now and the target date
-- Return days
local function datediffnow(d,m,y)
  local targetdate = os.time{day=d, month=m, year=y}
  local daysdiff = os.difftime(targetdate, os.time()) / (24 * 60 * 60)
  return math.floor(daysdiff)
end

-- Get the current date & time from the domoticz instance with time object
-- Return date & time now, i.e. 2018-09-05 09:09:00
local function isnow(domoticz)
  return domoticz.time.rawDate .. ' ' .. domoticz.time.rawTime
end

return {
  on = {
    timer = {
      -- 'every minute',
      'at 00:15'
    }
  },
  execute = function(domoticz, timer)
    domoticz.devices(IDXDAYSX).updateText(datediffnow(1,9,2021) .. ' Tage')
    domoticz.devices(IDXDAYSY).updateText(datediffnow(1,9,2016) .. ' Tage')
    domoticz.devices(IDXDAYSZ).updateText(datediffnow(1,9,2019) .. ' Tage')
    domoticz.devices(IDXCONTROLMSG).updateText('Days-to-go updated ' .. isnow(domoticz))
  end
}
```

## Domoticz Log

### *Script saved*

```
2018-09-04 14:44:35.496 Status: EventSystem: reset all events...
2018-09-04 14:44:35.497 Status: dzVents: Write file:
/home/pi/domoticz/scripts/dzVents/generated_scripts/days_to_go_update.lua
```

### *Script running*

(for tests running every minute instead of once per day)

```
2018-09-04 15:36:00.068 Status: dzVents: Info: ----- Start internal script: days-to-go-update:, trigger: every minute
2018-09-04 15:36:00.085 Status: dzVents: Info: ----- Finished days-to-go-update
```

## Addition

A script addition is made to calculate an age in xx Years yy Days.

```
ageyearsdays(7,9,2016)
```

### *Lua Functions*

```
-- Calculate the date difference in days between now and the target date
-- Return days
local function datediffnow(d,m,y)
    -- Set the target date from the parameter
    local targetdate = os.time{day=d, month=m, year=y}

    -- Get the time diff between now and the target date in seconds in a day
    local daysdiff = os.difftime(targetdate, os.time()) / (24 * 60 * 60)

    -- Return the days
    return math.floor(daysdiff)
end

-- Calculate the age in years + days
-- Return age as string years J days T
local function ageyearsdays(dbirth,mbirth,ybirth)
    -- get the actual date
    t = os.date ("*t")

    -- get the year, month, day for now
    ynow = t.year
    mnow = t.month
    dnow = t.day

    -- year difference between now year and the target year
    ydiff = ynow - ybirth
    -- days difference between now and the target day+month for the new year
    -- i.e. dtarget+mtarget+ynow
    ddiff = math.abs(datediffnow(dbirth, mbirth, ynow))

    -- build the age string to return
    age = ydiff .. ' J ' .. ddiff .. ' T'
    -- print(age)
    return age
end
```

# Device Status

## Purpose

To view all device attributes of a used device.

To be used for a quick lookup either

- to check the data of a device
- to review device attributes during the development of scripts/plugins/custom pages etc..

## Solution

A **Domoticz Custom Page** with HTML tags:

- **Select**: holding all devices with status used and not hidden, sorted by property Name
- **Button**: Refresh the data. Select shows the text “Select...” and the TextArea is empty
- **Label**: Showing the device attributes idx and Data
- **TextArea**: listing the selected device attributes
- **Label**: showing the version of the custom page

**Screenshot Domoticz GUI with Custom Menu and Device Status entry**



**Screenshot Custom Page Device Status with attributes for the device with IDX=28**

The screenshot shows a custom Domoticz page titled "Device Status". The top navigation bar is identical to the one in the previous screenshot. Below the title, there's a dropdown menu showing "Windmesser - Wind" and "Idx: 28 | Data: 168.00;SSE;12;47;16.8;16.8". The main content area contains a large block of text listing device attributes:

```

AdjMulti: 1
AdjMulti2: 1
AdjValue: 0
AdjValue2: 0
BatteryLevel: 0
Chill: 16.8
CustomImage: 0
Data: 168.00;SSE;12;47;16.8;16.8
Description:
Direction: 168
DirectionStr: SSE
Favorite: 1
Gust: 3
HardwareID: 4
HardwareName: RFXtrx433e
HardwareType: RFXCOM - RFXtrx433 USB 433.92MHz Transceiver
HardwareTypeval: 1
HaveTimeout: false
ID: 9D16
LastUpdate: 2020-10-24 11:26:11

```

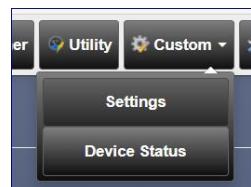
At the bottom left, it says "20201024 by rwbl".

## Installation

Copy the file "Device Status.html" to following Domoticz folder:

```
~domoticz/www/templates
```

The Domoticz GUI > Tab Custom shows the new custom menu entry “Device Status” in the dropdown list:



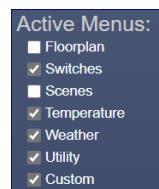
Direct page access from a browser:

```
http://domoticz-ip:port/#/Custom/Device%20Status
```

*Note*

Ensure to enable the Domoticz GUI Custom Menu (checkmark the entry Custom):

```
GUI > Setup > Settings > System > Active Menus > Custom
```



*Note*

### Custom Page Name

The filename contains a space, which means when editing the file from the CLI with for example with “nano”, the filename must be called like nano "Device Status.html".

```
pi@dodev:~ $ cd domoticz/www/templates  
pi@dodev:~/domoticz/www/templates $ nano "Device Status.html"
```

# HTML File

Source: Device Status.html

```
<!--
domoticz-homeautomation-workbook function Device Status
Domoticz custom page:
Device Status.html (/home/pi/domoticz/www/templates)
Purpose:
To quickly check the status, i.e. all properties of a device.
20201024 rwbl
-->
<style>
/* hint style text small */
.hint {
  font-size: smaller;
  color: white;
}
/* Textarea listing the device properties */
.taDevice {
  width: 100%;
  padding: 5px 5px;
  box-sizing: border-box;
  border: 2px solid #ccc;
  border-radius: 4px;
  background-color: #f8f8f8;
  font-size: 14px;
  resize: none;
}
</style>
<!--
HTML
-->
<div class="container">
  <table class="page-header-small" border="0" cellpadding="0" cellspacing="0" width="100%">
    <tr>
      <td align="left">
        <h4 data-i18n="Device Status"></h4>
      </td>
      <td align="right">
        <a class="btnstyle3" onclick="refreshDevicesList();"><span data-i18n="Refresh"></span></a>
      </td>
    </tr>
  </table>
  <table class="display" border="0" cellpadding="0" cellspacing="0" width="100%">
    <tr>
      <td align="left">
        <select id="deviceslist" class="combobox ui-corner-all" style="width: auto;" onchange="setDeviceProperties(statusDevices)">
          </select>
        </td>
      </tr>
      <tr>
        <td align="left">
          <label>
            <span id="labeldeviceproperties" data-i18n="Status" class="text"></span>
          </label>
        </td>
      </tr>
      <tr>
        <td align="left">
          <textarea id="deviceproperties" class="taDevice" rows="20" spellcheck="false"></textarea>
        </td>
      </tr>
    </table>
    <div id="pageversion" class="hint"></div>
  <!--
  JavaScript
  -->
<script type="text/javascript" charset="utf-8">
  // Sort an array of objects by key

```

```

// Example: data.result.sortBy(el => el.Name);
Array.prototype.sortBy = function(key_func, reverse=false){
    return this.sort( (a, b) => {
        var keyA = key_func(a),
            keyB = key_func(b);
        if(keyA < keyB) return reverse? 1: -1;
        if(keyA > keyB) return reverse? -1: 1;
        return 0;
    });
}

// Get all devices data from the domotiz database and create the select dropdown entries
// Hint requestUrl parameterSet:
// If this custom page is accessing on the same system, then the requesturl starts with json.htm...
// "json.htm?type=devices&used=true&displayhidden=0";
// No need to add the domoticz ip and port: "http://domoticz-
ip:port?json.htm?type=devices&used=true&displayhidden=0";
function getData(title, requestUrl) {
    // console.log(requestUrl);
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            if(typeof data.result == "undefined"){
                // console.log(data);
                console.log("Status all devices - no devices found (status: " + data.status + ", title: " + data.title + ")");
                return false;
            }
            console.log("Status all devices - Succesfully retreived (status: " + data.status + ", title: " + data.title + ")");
            // Array of all device objects used to retrieve the properties via the select dropdown option
            value idx
            statusDevices = [];
            // String holding all devices option tag
            var optionList = "";
            optionValue = '<option value="0">Select...</option>';
            optionList += optionValue;
            // Sort the device object array by key Name
            data.result.sortBy(el => el.Name);
            // Loop over the device entries to build the option and push to the device objects array
            data.result.forEach(function(entry) {
                entryType = entry.Type;
                if (entry.Type === "General") { entryType = entry.SubType; }
                optionValue = '<option value="'+entry.idx+'">' + entry.Name+ ' - ' + entryType +'';
                optionList += optionValue;
                statusDevices.push(entry);
                // console.log("Added " + entry.idx + ", " + entry.Name + ", " + entry.Data);
            });
            //
            console.log("Status all devices - Done");
            // Update selector
            document.getElementById("deviceslist").innerHTML = optionList;
        },
        error: function () {
            console.log("[ERROR] Status all devices. Cannot communicate to the Domoticz server!");
            return false;
        }
    });
}

// Set the device property information in the textarea
function setDeviceProperties(statusDevices) {
    var idx = document.getElementById("deviceslist").value;
    // console.log("Device idx " + idx + " selected.");
    // Find the specific device object, by its idx, in an array of device objects
    var deviceObj = statusDevices.find(obj => obj.idx === idx.toString());
    // if the device is found, get the properties
    if (typeof deviceObj !== 'undefined') {
        // Update the textarea label
        document.getElementById("labeldeviceproperties").innerHTML = "Idx: " + deviceObj.idx + " | Data:
" + deviceObj.Data;
        // Get the deviceproperties from the object
        deviceProperties = "";
    }
}

```

```
for (const [key, value] of Object.entries(deviceObj)) {
    deviceProperties += key + ":" + value + "\n";
}
// Update the textarea content
document.getElementById("deviceproperties").innerHTML = deviceProperties;
}

// Refresh the page. Assigned to a button.
function refreshDevicesList() {
    window.location.reload();
    console.log("Devices list refreshed");
}

function setVersion(pageVersion) {
    $('#pageversion').html(pageVersion);
};

// Load the document
// Get status of all used devices as array of object
$(document).ready(function(){
    setVersion("20201024 by rwbl");
    getData(
        "Domoticz Device Status",
        "json.htm?type=devices&used=true&displayhidden=0");
});
</script>
```

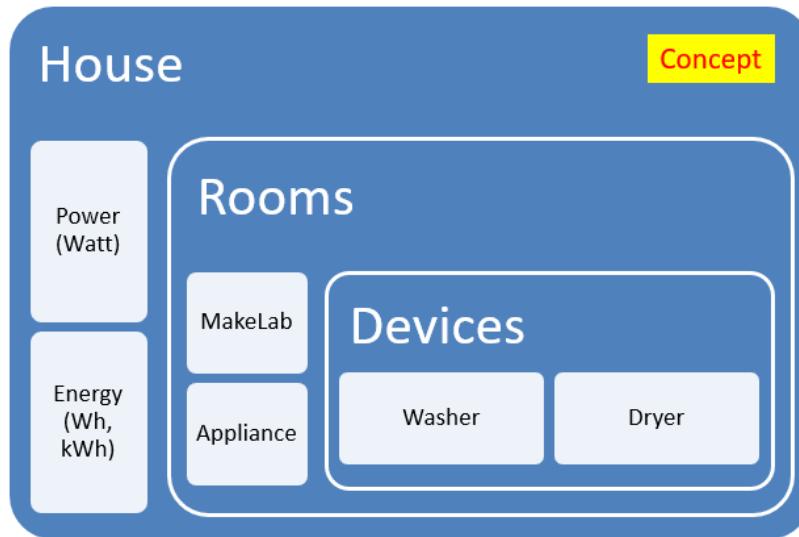
# Electric Usage

## Purpose

To monitor the Power (Watt) and Energy (kWh) consumption for

- the house (as total)
- selected rooms
- selected devices

## Solution



The solution makes use of several functions, described in the dedicated chapters:

- [Electric Usage House](#)
- [Electric Usage Rooms/Devices](#)

<TODO> Add more

# Electric Usage House (volkszaehler)

## Purpose

To measure the total Power (Watt) and Energy (Wh, kWh) for the house.

## Solution

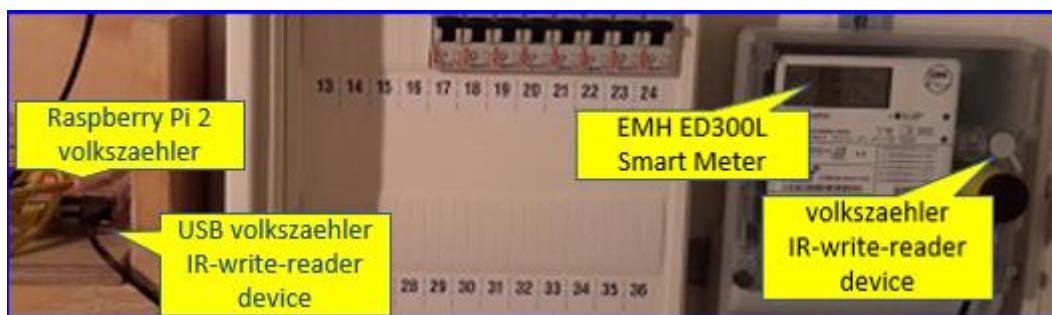
The house has a Smart Meter (Type EMH ED300L) which records consumption of electric energy used for monitoring and billing.

A [volkszaehler](#) (Open Source Smart Meter solution) IR-write-reader device sends data, via USB, to a Raspberry Pi 2 running the volkszaehler middleware.

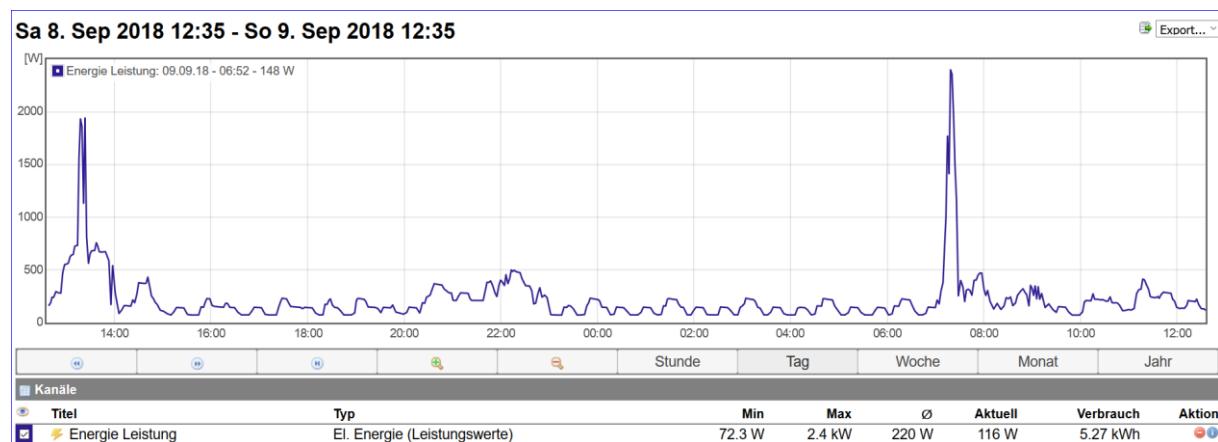
The Domoticz system polls the volkszaehler every minute for average power data, which is assigned to a Domoticz devices.

For more information, read the [Appendix volkszaehler Setup](#), but some screenshots of the volkszaehler setup and web frontend.

### Hardware Setup volkszaehler



### Web Frontend volkszaehler



# Domoticz Configuration

## Device

Create a Virtual Sensor with Name **Electric Usage House** and Type **Electric (Instant+Counter)**.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
171	VirtualSensors	00082171	1	Electric Usage House	General	kWh	333.939 kWh

The widget is added to the Utility tab.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
171	VirtualSensors	00082171	1	Electricity House	General	kWh	3.775 kWh

The screenshot shows the 'Edit Device' dialog and the utility tab. The utility tab displays a widget for 'Electric Usage House' showing current power usage of 2317 Watt and total energy usage of 3.775 kWh. The 'Edit Device' dialog shows the device configuration: Name: Electric Usage House, Description: Energy (Wh) & Power (Watt) provided, every minute, by the volkszaehler Raspberry Pi. The Energy is computed from the Power., Type: Usage, and Energy read: Computed (radio button selected).

The widget shows:

- Top Right: Current Power Usage (Watt) obtained from volkszaehler data.average property.
- Under Today: Energy Usage (kWh) calculated by Domoticz.

The widget is edited to set the property “Energy read” to computed, means Domoticz calculates the Energy (Wh, kWh) using the Power Usage (Watt).

## RFXMeter

To calculate the Energy costs, following settings used:  
(Domoticz GUI > Setup > Settings > RFXMeter/Counter Dividers)

RFXMeter/Counter Dividers:		Costs T1: 0.2217 / kWh		Costs T2: 0.2217 / kWh	
Energy: 1000		Costs R1: 0	/ kWh	Costs R2: 0	/ kWh

T1 = energy usage meter tariff 1 (Night)

T2 = energy usage meter tariff 1 (Day)

The costs can be obtained from the Device Log > Report. The info below uses initial data just to show the calculation, i.e. June 1.86 EUR = 8.403 kWh x 0.2217 EUR/kWh.

Electricity House 2019			
Total Usage: 10.057 kWh	Counter: 10.085 kWh	Year Cost: 2.23	
Month	Usage	Costs	
05. May	1.654	0.37	=
06. June	8.403	1.86	↑

## Polling

To get the electricity data from the volkszaehler server and update the Domoticz device, a dzVents Lua script event “electricity\_usage\_update” is executed every minute (Timer):

1. Send, every minute – polling interval, an HTTP GET Request to the Raspberry Pi running the volkszaehler.  
The request response is handled using callback in the script.
2. The volkszaehler response is a JSON string containing the tuples, data.average, data.consumption.
3. Parse the JSON string to get the power usage from key “data.average”.  
(which calculates the average electric usage over the last minute- the polling interval)
4. Update the Domoticz device Electricity House  
The device property “Energy read” is set to Computed, which means Domoticz calculated the Energy (kWh) from the given Power (Watt).

### Cross Check Domoticz Calculation vs Smart Meter Display

Compared, with data for several hours, the Domoticz calculation (Wh from the Day chart export data table) against the Smart Meter (Type EMH ED300L) display (kWh without digits, difference between start and end time in hours).

The result:

Domoticz 3.2 kWh vs Smart Meter 3 kWh (display delta 8077 – 8074).

As the Smart Meter does not display digits, the result is getting close.

<TODO>Performsome more cross checks.

## HTTP API GET Request

The HTTP GET request uses Channel 1 (1.8.0 Zählerstand, which is also displayed on the Meter) with UUID: 5b8ea2a0-b342-11e8-bec6-15c040e6d041) and timestamp parameter:

- **from=1+minutes+ago**
- **to=now**

### Request Examples with JSON Response

(see also the volkszaehler reference) (ensure to replace, in the URL request, the space by a +)

Obtain the Power Usage from key: data.average

#### Data for the last minute

```
http://domoticz-ip /middleware.php/data/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json?from=1+minutes+ago&to=now
```

```
{"version": "0.3", "data": {"tuples": [[1559376630383, 296.907, 1], [1559376632787, 299.501, 1], [1559376635056, 317.32, 1], [1559376637366, 311.688, 1], [1559376639582, 324.91, 1], [1559376641817, 322.148, 1], [1559376643928, 341.071, 1], [1559376646081, 334.417, 1], [1559376648138, 350.024, 1], [1559376650187, 351.391, 1], [1559376652233, 351.906, 1], [1559376654260, 355.205, 1], [1559376656255, 360.902, 1], [1559376658459, 326.679, 1], [1559376660686, 323.305, 1], [1559376662344, 651.387, 1], [1559376664129, 806.723, 1], [1559376665925, 400.891, 1], [1559376667657, 415.704, 1], [1559376669441, 403.587, 1], [1559376671175, 415.225, 1], [1559376672970, 401.114, 1], [155937667475, 6, 403.135, 1], [1559376676530, 405.862, 1], [1559376678314, 403.587, 1], [1559376680046, 415.704, 1], [1559376681844, 400.445, 1], [1559376683575, 415.945, 1], [1559376685339, 408.163, 1], [1559376687028, 426.288, 1], [1559376688804, 405.405, 1], [1559376690535, 415.945, 1]], "uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "from": "1559376627958", "to": "1559376690535", "min": [1559376630383, 296.90721677136], "max": [1559376664129, 806.72268794865], "average": 385.445, "consumption": 6.7, "rows": 33}}
```

The average power is 385.445 Watt calculated from 33 tuples.

#### Data now with one tuple

```
http://NNN.NNN.1.139/middleware.php/data/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json?from=now&tuples=1
```

```
{"version": "0.3", "data": {"tuples": [[1559377278643, 334.728, 1]]}, "uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "from": "1559377276492", "to": "1559377278643", "min": [1559377278643, 334.72803378454], "max": [1559377278643, 334.72803378454], "average": 334.728, "consumption": 0.2, "rows": 1}}
```

The average power is 334.728 Watt calculated from 33 tuples.

## Automation Script

The dzEvents Lua script event “electricity\_usage\_update” runs every minute and has a callback “ElectricityUsageCallback“ handling the volkszaehler server response.

Script source code with minimal comments (the full latest version can be found on GitHub).  
Event name: electricity\_usage\_update

```

local URL_DOMOTICZ = 'http://DOMOTICZ-IP:8080'
local URL_VZREQUEST = 'http://VZ-IP/middleware.php/data/5b8ea2a0-b342-11e8-bec6-
15c040e6d041.json?from=1+minutes+ago&to=now'
local RES_VZREQUEST = 'electricusagehouse'
local IDX_ELECTRICUSAGEHOUSE = 171 -- Type=General, SubType=kWh

return {
    -- active = true,
    on = {
        timer = {
            'every minute'
        },
        httpResponses = {
            RES_VZREQUEST
        }
    },
    execute = function(domoticz, item)
        if (item.isTimer) then
            domoticz.openURL({ url = URL_VZREQUEST, method = 'GET', callback = RES_VZREQUEST })
        end

        if (item.isHTTPResponse) and (item.trigger == RES_VZREQUEST) then
            if (item.ok) then -- statusCode == 2xx
                local power = math.floor(tonumber(item.json.data.average))
                domoticz.openURL({
                    url = '' .. URL_DOMOTICZ .. '/json.htm?type=command&param=udevice&idx=' ..
IDX_ELECTRICUSAGEHOUSE .. '&nvalue=0&svalue=' .. tostring(power) .. ';0',
                    method = 'GET'})
                local message = ('Electric Usage vz: %02d W, Status: %02d'):format(power, item.statusCode)
                domoticz.log(message, domoticz.LOG_INFO)
                message = ('Wh = Actual: %02d, Today: %02d, Total:
%02d'):format(domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhActual,
domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhToday,domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhTotal)
                domoticz.log(message, domoticz.LOG_INFO)
            end

            if not (item.ok) then -- statusCode != 2xx
                local message = '[ERROR] ' .. HTTPCALLBACKNAME .. ':' .. tostring(item.statusCode) .. ' ' ..
msgbox.isnowdatetime(domoticz)
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, message)
                domoticz.log(message, domoticz.LOG_INFO)
            end
        end
    end
}
}

```

## Domoticz Log

```
#Script Start Finished
2019-06-01 10:33:00.619 Status: dzVents: Info: ----- Start internal script: electricity_usage_update:, trigger: every minute
2019-06-01 10:33:00.620 Status: dzVents: Info: ----- Finished electricity_usage_update
2019-07-29 10:03:00.259 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua

#Script HTTP Callback after previous finishing
2019-07-29 10:03:00.396 Status: dzVents: Info: Handling httpResponse-events for: "electricusagehouse"
2019-07-29 10:03:00.396 Status: dzVents: Info: ----- Start internal script: electric_usage_house: HTTPResponse: "electricusagehouse"
2019-07-29 10:03:00.403 Status: dzVents: Info: Electric Usage vz: 2528 W, Status: 200
2019-07-29 10:03:00.420 Status: dzVents: Info: Wh = Actual: 2408, Today: 2579, Total: 334381
2019-07-29 10:03:00.420 Status: dzVents: Info: ----- Finished electric_usage_house
```

## MQTT

Example MQTT payload (topic "domoticz/out") for Device “Electric UsageHouse” with IDX=171.

```
{"Battery" : 255, "EnergyMeterMode" : "1", "RSSI" : 12, "description" : "Energy (Wh) & Power (Watt) provided, every minute, by the volkszaehler Raspberry Pi.\nThe Energy is computed from the Power.", "dtype" : "General", "id" : "00082171", "idx" : 171, "name" : "Electri Usage House", "nvalue" : 0, "stype" : "kWh", "svalue1" : "824", "svalue2" : "4119.1", "unit" : 1}
```

## Properties

- svalue1 = current power usage (Watt)
- svalue2 = total energy cumulated over all days (kWh).

## HTTP API Request

### Request

```
http://domoticz-ip:8080/json.htm?type=devices&rid=171
```

### JSON Response (extract)

```
{
  ...
  "app_version" : "4.10717",
  "result" : [
    {
      "CounterToday" : "2.484 kWh",
      "Data" : "4.146 kWh",
      "EnergyMeterMode" : "1",
      "HardwareName" : "VirtualSensors",
      "LastUpdate" : "2019-06-01 10:49:00",
      "Name" : "Electricity House",
      "SubType" : "kWh",
      "Type" : "General",
      "TypeImg" : "current",
      "Unit" : 1,
      "Usage" : "510 Watt",
      "Used" : 1,
      "idx" : "171"
    }
  ],
  "status" : "OK",
  "title" : "Devices"
}
```

## Properties

- Usage = current power usage (Watt)
- Data = total energy cumulated over all days (kWh).

### *HTTP API Information*

```
Electricity (instant and counter)
/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=POWER;ENERGY
IDX = id device
POWER = current power
ENERGY = cumulative energy in Watt-hours (Wh)
Incrementing counter because type "Energy read : Computed"
```

## Log Examples

The log for the device “Electricity House” shows the

- Day = Power (Watt, every 5 minutes) + Energy (Wh, every hour)
- Week = total daily energy (kWh).

In the MQTT and HTTP API Request, the total energy is returned (~4.1 kWh).



### Note

This information can also be obtained by amending the previous dzVents Lua script “electricity\_usage\_update”:

```
domoticz.log('[INFO] WhToday: ' .. tostring(domoticz.devices(IDX_ELECTRICITYHOUSE).WhToday),
domoticz.LOG_INFO)
domoticz.log('[INFO] WhTotal: ' .. tostring(domoticz.devices(IDX_ELECTRICITYHOUSE).WhTotal),
domoticz.LOG_INFO)
domoticz.log('[INFO] WhActual: ' .. tostring(domoticz.devices(IDX_ELECTRICITYHOUSE).WhActual),
domoticz.LOG_INFO)
```

```
2019-06-01 11:08:00.984 Status: dzVents: Info: [INFO] Electric usage updated: 106 W, status:200
2019-06-01 11:08:00.985 Status: dzVents: Info: [INFO] WhToday: 2671
2019-06-01 11:08:00.985 Status: dzVents: Info: [INFO] WhTotal: 4333.2
2019-06-01 11:08:00.985 Status: dzVents: Info: [INFO] WhActual: 162
```

## Database Table

The device “Electricity House” stores its data in database “domoticz.db”, table Meter with structure

```
CREATE TABLE [Meter] (
[DeviceRowID] BIGINT NOT NULL,
[Value] BIGINT NOT NULL,
[Usage] INTEGER DEFAULT 0,
[Date] DATETIME DEFAULT (datetime('now','localtime'))
);
```

- Value is current Power in Watt
- Usage is cumulated Energy in Wh

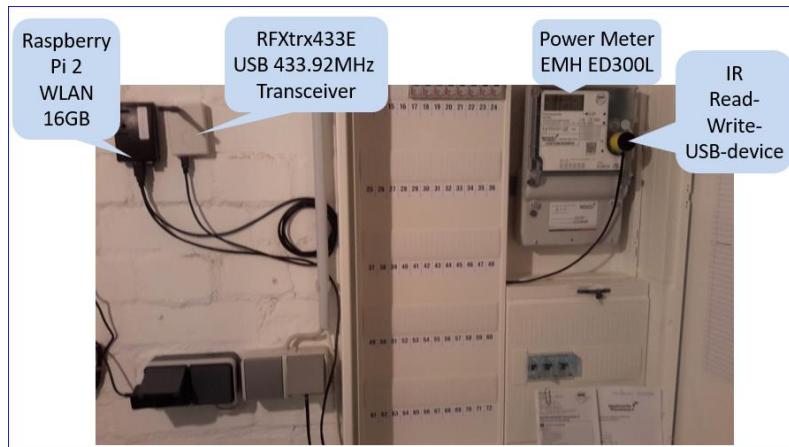
**Example SQL SELECT on table Meter**

```
cd domoticz
~/domoticz $ sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
sqlite> select * from Meter where DeviceRowID = 171 and Date > "2019-06-02 09:50:00";
171|10115|1690|2019-06-02 09:55:00
171|10132|1690|2019-06-02 10:00:00
```

# volkszaehler

## Overview

The volkszaehler (vz) Raspberry Pi (Model 2 with stretch) is connected to the electric meter reader [EMH ED300L](#) using the IR-Read-Write-USB-Device (provided by volkszaehler). The data is polled on power consumption change and stored in a vz MySQL database. Every 5 minutes Domoticz requests the average power consumption over the last 5 minutes. The vz data is kept in the MySQL database for 24 hours (older data is deleted at midnight by a Python script – see Database Management).



### Note

See [Appendix volkszaehler Setup](#) for actual update and more details.

References: [Raspberry-Pi](#), [API](#), [Debug](#), [EMH ED300L](#), [Node-RED](#).

## Setup Raspberry Pi

The Raspberry Pi is installed following the installation procedure described [here](#).

### Fix IP Address

The communication between the Domoticz system and the vz Server is via WLAN. The vz Server has a fixed IP address NNN.NNN.1.NNN, which is defined in the configuration files:

#### /etc/dhcpcd.conf

```
# wlan interface
interface wlan0
static ip_address=vz-ip/24
static routers=NNN.NNN.1.1
static domain_name_servers=NNN.NNN.1.1
```

#### /etc/wpa\_supplicant/wpa\_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=DE
network={
    ssid="o2-WLAN78_EXT"
    psk="*****"
}
```

In the documentation, the volkzaehler IP is referred as **vz-ip**

## USB Port Symlink

To ensure the right USB port is used, the Symlink `ttyUSB-VZ` is defined.

```
sudo nano /etc/udev/rules.d/99-usb-serial.rules
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="00F2E620",
SYMLINK+="ttyUSB-VZ", MODE="0666"
```

## Check

```
ls /dev/ttyU*
/dev/ttyUSB0  /dev/ttyUSB-VZ
```

## Serial Link

The serial link must be setup with 9600 baud, 8n1. This is done by terminal command:

which must be added to rc.local to ensure correct settings for the USB port during setup.

Check if data is received from the IR-Read-Write-USB-device:

```
xxd </dev/ttyUSB-VZ
00000000: 1b1b 1b1b 0101 0101 7607 000f 0016 bc87 .....v.....
00000010: 6200 6200 7263 0101 7601 0107 000f 0553 b.b.rc..v....S
00000020: 942d 0b09 0145 4d48 0000 532f 3801 0163 .-...EMH..S/8..c
00000030: 6a00 7607 000f 0016 bc88 6200 6200 7263 j.v.....b.b.rc
00000040: 0177 010b 0901 454d 4800 0053 2f38 0701 .w....EMH..S/8..
00000050: 0062 0aaf ff72 6201 6505 530d fc7a 7707 .b...rb.e.S..zw.
00000060: 8181 c782 03ff 0101 0101 0445 4d48 0177 .....EMH.w
00000070: 0701 0000 0009 ff01 0101 010b 0901 454d .....EMH.w
```

### *Note*

The tool minicom can also be used to setup the serial port.

```
sudo apt-get install minicom  
sudo minicom -s
```

## vz Setup Channels

Setting up the channels using the vz frontend (url=http://vz-ip) is crucial.

Must read: [https://wiki.volkszaehler.org/howto/emh\\_pv-anlage?redirect=1](https://wiki.volkszaehler.org/howto/emh_pv-anlage?redirect=1)

After the channels have been defined, the `vzlogger.conf` must be changed to add the channels.

## Important

The EMH ED300L is sending in SML (Obis-Codes), which are 1.8.0 etc.

The ENR-EB36E is sending in SML (Obis-Codes), which are 1.0.0 etc.  
These are NOT S0-Impulses, but counters, i.e. „Zählerstände, Leistungswerte“ depending Obis-Code.

The meter displays the OBIS Code 1.8.0, meter reading (“Zählerstand”, kWh) and current power usage (“Momentane Leistung”, Watt).

There are two channels defined:

**Channel 1:** Haus 1.8.0 (Bezug +A), Typ El. Energie (Zählerstände), Auflösung 1000

UUID:5b8ea2a0-b342-11e8-bec6-15c040e6d041

**This channel is used for Domoticz.**

**Channel 2:** Haus 16.7.0 (Leistung), Typ El. Energie (Leistungswerte)

UUID: 958bce60-b342-11e8-b54f-bbec0573e1f4

Channel: Zählerstand	Channel: Leistung																																																
Haus 1.8.0 (Bezug +A), Typ El. Energie (Zählerstände), Auflösung 1000, kWh	Haus 16.7.0 (Leistung), Typ El. Energie (Leistungswerte), Watt																																																
<p>Kanal hinzufügen</p> <p>Öffentliche Kanäle Private Kanäle <b>Kanal erstellen</b></p> <table> <tr> <td>Eigenschaft</td> <td>Wert</td> </tr> <tr> <td>Middleware:</td> <td>Local (default)</td> </tr> <tr> <td>Typ:</td> <td>El. Energie (Zählerstände)</td> </tr> <tr> <td>Titel</td> <td>Energie Zählerstand</td> </tr> <tr> <td>Auflösung</td> <td>1000</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td><span style="background-color: cyan; width: 10px; height: 10px;"></span></td> </tr> <tr> <td>Stil</td> <td>steps</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </table> <p><b>Erstellen</b> <b>Cookie:</b> <input checked="" type="checkbox"/></p>	Eigenschaft	Wert	Middleware:	Local (default)	Typ:	El. Energie (Zählerstände)	Titel	Energie Zählerstand	Auflösung	1000	Öffentlich	<input checked="" type="checkbox"/>	Farbe	<span style="background-color: cyan; width: 10px; height: 10px;"></span>	Stil	steps	Füllgrad		Achse	auto	Kosten		Initialverbrauch		<p>Kanal hinzufügen</p> <p>Öffentliche Kanäle Private Kanäle <b>Kanal erstellen</b></p> <table> <tr> <td>Eigenschaft</td> <td>Wert</td> </tr> <tr> <td>Middleware:</td> <td>Local (default)</td> </tr> <tr> <td>Typ:</td> <td>El. Energie (Leistungswerte)</td> </tr> <tr> <td>Titel</td> <td>Energie Leistung</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td><span style="background-color: cyan; width: 10px; height: 10px;"></span></td> </tr> <tr> <td>Stil</td> <td>lines</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Auflösung</td> <td></td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </table> <p><b>Erstellen</b> <b>Cookie:</b> <input checked="" type="checkbox"/></p>	Eigenschaft	Wert	Middleware:	Local (default)	Typ:	El. Energie (Leistungswerte)	Titel	Energie Leistung	Öffentlich	<input checked="" type="checkbox"/>	Farbe	<span style="background-color: cyan; width: 10px; height: 10px;"></span>	Stil	lines	Füllgrad		Achse	auto	Auflösung		Kosten		Initialverbrauch	
Eigenschaft	Wert																																																
Middleware:	Local (default)																																																
Typ:	El. Energie (Zählerstände)																																																
Titel	Energie Zählerstand																																																
Auflösung	1000																																																
Öffentlich	<input checked="" type="checkbox"/>																																																
Farbe	<span style="background-color: cyan; width: 10px; height: 10px;"></span>																																																
Stil	steps																																																
Füllgrad																																																	
Achse	auto																																																
Kosten																																																	
Initialverbrauch																																																	
Eigenschaft	Wert																																																
Middleware:	Local (default)																																																
Typ:	El. Energie (Leistungswerte)																																																
Titel	Energie Leistung																																																
Öffentlich	<input checked="" type="checkbox"/>																																																
Farbe	<span style="background-color: cyan; width: 10px; height: 10px;"></span>																																																
Stil	lines																																																
Füllgrad																																																	
Achse	auto																																																
Auflösung																																																	
Kosten																																																	
Initialverbrauch																																																	
<p><b>UUID</b></p> <p>5b8ea2a0-b342-11e8-bec6-15c040e6d041</p> <pre>// /etc/vzlogger.conf { // Zählerstand "uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "middleware" : "http://localhost/middleware.php", "identifier" : "1-0:1.8.0" }</pre>	<p><b>UUID</b></p> <p>958bce60-b342-11e8-b54f-bbec0573e1f4</p> <pre>/etc/vzlogger.conf { // Leistungswert "uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4", "middleware" : "http://localhost/middleware.php", "identifier" : "1-0:16.7.0" }</pre>																																																

Web Frontend Settings	Web Frontend Settings
<pre>{   "version": "0.3",   "entity": {     "uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041",     "type": "electric meter",     "color": "aqua",     "fillstyle": 0,     "public": true,     "resolution": 1000,     "style": "steps",     "title": "Energie Z\u00fchlerstand",     "yaxis": "auto"   } }</pre>	<pre>{   "version": "0.3",   "entity": {     "uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4",     "type": "powersensor",     "color": "aqua",     "fillstyle": 0,     "public": true,     "style": "lines",     "title": "Energie Leistung",     "yaxis": "auto"   } }</pre>

### Example requesting Data for Channel 1 for the timeframe 1minute ago till now

```
http://rpi-volkszaehler-ip/middleware.php/data/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json?from=1+minutes+ago&to=now
```

### HTTP Response

```
{"version": "0.3", "data": {"tuples": [[1559311263427, 190.476, 1], [1559311265329, 189.274, 1], [1559311267248, 187.598, 1], [1559311269139, 190.375, 1], [1559311271039, 189.474, 1], [1559311272928, 190.577, 1], [1559311274860, 186.335, 1], [1559311276762, 189.274, 1], [1559311278640, 191.693, 1], [1559311280542, 189.274, 1], [1559311282443, 189.374, 1], [1559311284374, 186.432, 1], [1559311286253, 191.591, 1], [1559311288048, 200.557, 1], [1559311289875, 197.044, 1], [1559311291775, 189.474, 1], [1559311293613, 195.865, 1], [1559311295568, 184.143, 1], [1559311297457, 190.577, 1], [1559311299399, 185.376, 1], [1559311301330, 186.432, 1], [1559311303284, 184.237, 1], [1559311305205, 187.402, 1], [1559311307116, 188.383, 1], [1559311309048, 186.335, 1], [1559311310979, 186.432, 1], [1559311312891, 188.285, 1], [1559311314865, 182.371, 1], [1559311316619, 205.245, 1], [1559311318351, 207.852, 1], [1559311320072, 209.181, 1], [1559311321804, 207.852, 1], [1559311323589, 201.681, 1]], "uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "from": 1559311261537, "to": 1559311323589, "min": 1559311314865, "max": 182.37082168777, "average": 191.452, "consumption": 3.3, "rows": 34}}
```

The properties **data.average** and **data.consumption** are used to update the Domoticz devices.

## vzlogger

The [vzlogger](#) is used to read the serial data and transfer the data to the middleware.

### Configuration File

The configuration file is /etc/vzlogger.conf. Must [read](#) to understand the settings.

Ensure no empty lines are spaces after the last }.

#### Important

After making changes (sudo nano /etc/vzlogger.conf) restart the vzlogger and check status.

```
sudo systemctl restart vzlogger
sudo systemctl status vzlogger
```

#### /etc/vzlogger.conf

```
{
  "daemon": true,
  "verbosity": 0,
  "log": "/var/log/vzlogger.log",
  "retry": 30,
  "local": {
    "enabled": false,
    "port": 8080,
    "index": true,
    "timeout": 30,
    "buffer": 600
  },
  "push": [],
  "meters" : [
    {
      "enabled" : true,
      "protocol" : "sml",
      "device" : "/dev/ttyUSB-VZ",
      "baudrate": 9600,
      "parity": "8n1",
      "channels": [
        {
          // Leistungswert
          "uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4",
          "middleware" : "http://localhost/middleware.php",
          "identifier" : "1-0:16.7.0"
        },
        {
          // Zählerstand
          "uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041",
          "middleware" : "http://localhost/middleware.php",
          "identifier" : "1-0:1.8.0"
        }
      ]
    }
  ]
}
```

## Start Stop Status

### Terminal commands

```
Stop:      sudo systemctl stop vzlogger
Start:     sudo systemctl start vzlogger
Restart:   sudo systemctl restart vzlogger
Status:    sudo systemctl status vzlogger
```

## LogFile

### Check

```
sudo cat /var/log/vzlogger.log
```

### Delete

```
sudo systemctl stop vzlogger
sudo rm /var/log/vzlogger.log
sudo systemctl start vzlogger
sudo cat /var/log/vzlogger.log
```

## vz Web Frontend

URL=<http://vz-ip/?uuid=958bce60-b342-11e8-b54f-bbec0573e1f4>  
(Leistungswerte)

Read more [here](#).



## Database Management

To manage the data stored in a MySQL database, the tool phpMyAdmin is used.  
Enter in a browser, the URL <http://vz-ip/phpmyadmin/>

### User vz-admin

Prior using, create a user vz-admin and grant access.

```
sudo mysql -uroot -praspberry
CREATE USER 'vz-admin'@'%' IDENTIFIED BY 'secure';
GRANT USAGE ON *.* TO 'vz-admin'@'%';
GRANT ALL PRIVILEGES ON `volkszaehler`.* TO 'vz-admin'@'%' WITH GRANT OPTION;
exit
```

### Login phpMyAdmin

Username: vz-admin, Password: \*\*\*\*\*, Database: volkszaehler

Table	Action	Rows	Type	Collation	Size	Overhead
aggregate	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	2,612,489	InnoDB	latin1_swedish_ci	190.3 MiB	0 B
data	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8_unicode_ci	48 KiB	-
entities	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	16	InnoDB	utf8_unicode_ci	32 KiB	-
entities_in_aggregator	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	0	InnoDB	utf8_unicode_ci	48 KiB	-
properties	<a href="#">Browse</a> <a href="#">Structure</a> <a href="#">Search</a> <a href="#">Insert</a> <a href="#">Empty</a> <a href="#">Drop</a>	127	InnoDB	utf8_unicode_ci	48 KiB	-
5 tables	Sum	2,612,632	InnoDB	latin1_swedish_ci	190.5 MiB	0 B

### Check Database Size

#### URL

```
http://vz-ip/middleware.php/capabilities/database.json?
```

#### Result

```
{"version":"0.3", "capabilities": {"database": {"data_rows":67235725,"data_size":5935382528,"aggregation_enabled":1,"aggregation_rows":2611868,"aggregation_ratio":25.742}}}
```

### Size of database and access\_log

#### Limit logging

```
sudo nano /etc/mysql/my.cnf
[mysqld]
expire_log_days = 2
max_binlog_size = 5M
```

## Delete Data

There are several ways to delete data from the vz MySQL database.

### Option 1

This is the hard way of deleting data via MySQL statements.

Example deleting all data older than 24 hours from database volkszaehler, table data.

```
mysql --user=root -praspberry

mysql> grant select, update, insert, delete on volkszaehler.* to vz@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> use volkszaehler

mysql> delete FROM data where timestamp< (unix_timestamp(current_timestamp)-24*60*60)*1000 ;
```

*Note*

This operation might take long...

Check the database size after completion

### Option 2

Use Terminal Command

```
wget -O - -q "http://vz-ip/middleware.php/data/27e28d00-b2dc-11e8-903d-
c3b75ae4ef20.json?operation=delete&from=now"
```

### Option 3

Use URL

#### Prepare mysql

```
mysql --user=vz-admin -psecure

mysql> grant select, update, insert, delete on volkszaehler.* to vz@localhost;
Query OK, 0 rows affected (0.01 sec)
```

#### URL deleting data

```
http://vz-ip/middleware.php/data/958bce60-b342-11e8-b54f-
bbec0573e1f4.json?operation=delete&from=1+day+ago
```

#### Result JSON

```
{"version":"0.3","rows":37466}
```

#### URL Check database size

```
http://vz-ip/middleware.php/capabilities/database.json?
```

#### Result JSON

```
{"version":"0.3","capabilities":{"database":{"data":{"rows":72047,"size":12632064}, "aggregation":{"rows":0,"size":49152,"ratio":0}}}}
```

## vzclean

### Purpose

To delete, every day at midnight, data which is older than 24 hours from the database volkszaehler, table data.

### Solution

A Python script (vzclean.py) logs into the database as user vz-admin and deletes the data from table data. The script is executed, via bash script (vzclean.sh) once a day at 00:05.

Script location:

```
/home/pi/scripts
```

The script uses the config file vzclean.ini to obtain MySQL database and user login information.

### Hint

After running vzclean or to check adhoc, use URL

```
http://vz-ip/middleware.php/capabilities/database.json?
```

## Python

### vzcleanconfig.py

```
## See https://pypi.org/project/configparser/
import configparser

def read_db_config(filename='vzclean.ini', section='mysql'):
    """ Read database configuration file and return a dictionary object
    :param filename: name of the configuration file
    :param section: section of database configuration
    :return: a dictionary of database parameters
    """
    # create parser and read ini configuration file
    parser = configparser.ConfigParser()
    parser.read(filename)

    # get section, default to mysql
    db = {}
    if parser.has_section(section):
        items = parser.items(section)
        for item in items:
            db[item[0]] = item[1]
            # print(db[item[0]])
    else:
        raise Exception('{0} not found in the {1} file'.format(section, filename))

    # print(db)
    return db
```

### vzclean.ini

```
[mysql]
host = localhost
database = volkszaehler
user = vz-admin
password = *****
```

### vzclean.py

```
#!/usr/bin/env python
"""

File: vzclean.py
Author: Robert W.B. Linn, 20180909
Project: AtHome
Purpose: Delete all channel data prior last 24 h from database volkszaehler, table data. Run once per day via crontab.
The script logs to a file. If the script is executed with parameter 1, then the log is printed to the console.
Run the script: python3 vzclean.py or python3 vzclean.py 1
The script is executed via crontab "5 0 * * * /home/pi/scripts/vzclean.sh 1" using the bash script vzclean.sh.
    cd /home/pi/scripts
    python3 vzclean.py $1
Example Log
    vzclean 20180909
    Connection: OK
    Rows before delete: 31
    Rows after delete: 1
    Rows last 24h: 25067
    Connection closed

Install mysql-connector
sudo apt-get update
sudo apt-get upgrade
sudo apt-get -y install python-mysql.connector
sudo apt-get -y install python3-mysql.connector

References
http://www.mysqltutorial.org/python-mysql-query/
```

```

volkszaehler database structure
https://wiki.volkszaehler.org/development/schema
volkszaehler API ref:
https://wiki.volkszaehler.org/development/api/reference
Table data
    id, int(11); channel_id, int(11); timestamp, bigint(20); value, double
"""

# Import
from mysql.connector import MySQLConnection, Error
from vzcleanconfig import read_db_config
import sys

vzclean = 'vzclean 20180909'
logfilename = 'vzclean.log'

# Define the queries including test queries
query_select_entities = "SELECT * FROM entities ;"
query_select_data = "SELECT * FROM data ;"
query_select_data_by_channel_ts = "SELECT * FROM data WHERE timestamp > (unix_timestamp(current_timestamp)-24*60*60)*1000 ;"
query_select_data_prior_last24h = "SELECT * FROM data WHERE timestamp < (unix_timestamp(current_timestamp)-24*60*60)*1000 ;"
query_select_data_prior_last24h_count = "SELECT COUNT(channel_id) FROM data WHERE timestamp < (unix_timestamp(current_timestamp)-24*60*60)*1000 ;"
# Ref: http://www.mysqltutorial.org/python-mysql-delete-data/
query_delete_data_prior_last24h = "delete FROM data where timestamp< (unix_timestamp(current_timestamp)-24*60*60)*1000 ;"

def log(logfile, text):
    """Write a line to the logfile
    :logfile: Logfile opened for writing (w+) or append (a+)
    :text: Line to write
    """
    logfile.write('{}\r\n'.format(text))

def logprint(logfilename):
    """Print the content of the logfile
    :logfile: Logfile opened for reading (r)
    """
    f = open(logfilename, "r")
    content = f.read()
    print(content)

def show_tables(conn):
    """Show the table for database volkszaehler
    :conn: Database connection
    """
    cursor = conn.cursor()
    cursor.execute("SHOW TABLES")
    for x in cursor:
        print(x)

def select_fetchone(conn, qry):
    """Query a table
    :conn: Database connection
    :qry: Query to execute
    """
    cursor = conn.cursor()
    cursor.execute(qry)

    row = cursor.fetchone()

    while row is not None:
        print(row)
        row = cursor.fetchone()

def select_count(conn, qry):
    """Get the number of records
    :conn: Database connection
    :qry: Select query with COUNT()
    :return: Number of records
    """
    cursor = conn.cursor()
    cursor.execute(qry)

```

```

        result = cursor.fetchone()
        return result[0]

def delete_data(conn, qry):
    """Delete data from a
    :conn: Database connection
    :qry: Delete statement
    """
    cursor = conn.cursor()
    cursor.execute(qry)
    # accept the change
    conn.commit()

# Connect and run queries
def cleandata(showlog):
    logfile = open(logfilename,"w+")
    log(logfile, vzclean)

    """ Connect to MySQL database """
    db_config = read_db_config()

    try:
        conn = MySQLConnection(**db_config)

        if conn.is_connected():
            log(logfile, 'Connection: OK')
            #show_tables(conn)
            #select_fetchone(conn, query_select_entities)
            #select_fetchone(conn, query_select_data_by_channel_ts)

            # delete data prior last 24h
            log(logfile, 'Rows before delete: {}'.format(select_count(conn,
query_select_data_prior_last24h_count)))
            delete_data(conn, query_delete_data_prior_last24h)
            log(logfile, 'Rows after delete: {}'.format(select_count(conn,
query_select_data_prior_last24h_count)))
            log(logfile, 'Rows last 24h: {}'.format(select_count(conn,
query_select_data_by_channel_ts)))
        else:
            log(logfile, '[ERROR] Database connection: failed')

    except Error as error:
        log(logfile, '[ERROR] Database connection:{}'.format(error))

    finally:
        conn.close()
        log(logfile, 'Connection closed')
        logfile.close()

    # If showlog argument is given, print the log to the console
    if showlog == '1':
        logprint(logfilename)

if __name__ == '__main__':
    if (len(sys.argv) == 2):
        cleandata(sys.argv[1])
    else:
        cleandata('')

```

## Bash

### vzclean.sh

```

#!/bin/bash
# Run vzclean Python script
# Optional Parameter: l to show the log
# Ensure to make the script executable: sudo chmod +x vzclean.sh
# Robert W.B. Linn, 20180909

# Set the scripts folder
cd /home/pi/scripts

```

```
# Run the script
python3 vzclean.py $1
```

## Crontab

Add to crontab the following lines to execute the script daily at 5 minutes past midnight.

```
crontab -e
# Run vzclean daily at 00:05
5 0 * * * /home/pi/scripts/vzclean.sh 1
```

## Ethernet Link volkszaehler Raspberry Pi

This is an example incase the vz Raspberry Pi and the Domoticz Raspberry Pi are direct connected via an Ethernet cable.

For the wired connection, a standard cable is used, no twisted wires required.  
The communication uses ETH0 with a static network addresses on both sides.

### Important

The ETH0 network address must be different then the WLAN network address.

### Configuration

vz Raspberry Pi [Raspbian GNU/Linux 8 (stretch)]	Domoticz Raspberry Pi [Raspbian GNU/Linux 8 (stretch)]
eth0 inet addr:169.254.87.85 Bcast:169.254.255.255 Mask:255.255.0.0	eth0 inet addr:169.254.87.84 Bcast:169.254.255.255 Mask:255.255.0.0
Set the static Ethernet network Address sudo nano /etc/dhcpcd.conf	sudo nano /etc/dhcpcd.conf
Add the lines	
interface eth0 static ip_address=169.254.87.85 static routers=169.254.0.1	interface eth0 static ip_address=169.254.87.84 static routers=169.254.0.1

### Note

\$ uname –a to determine the Linux version

\$ cat /etc/os-release to determine the Raspberry Pi release

Steps to login from vz Raspberry Pi to the Domoticz Raspberry Pi:

- Login to vz Raspberry Pi as user pi.
- Check if the Raspberry Pi Domoticz is reachable: \$ping 169.254.87.85
- Check if the sshd service is running on the vz Raspberry Pi: \$ps ax | grep sshd
- Connect to the Domoticz Raspberry Pi via ssh: \$ssh [pi@169.254.87.85](http://pi@169.254.87.85)  
pi@169.254.87.85's password: \*\*\*\*\*
- The pi@DoPro prompt is shown

# Electric Usage Rooms/Devices

## Purpose

To control the power and measure the energy consumption, voltage, current, power of connected devices to an power outlet.

The power outlet could be used for selected devices (coffee machine, washer, dryer etc.) or dedicated rooms (MakeLab, Library).

## Solution Homematic IP

The Homematic IP Pluggable Switch and Meter is used, as this device enables to switch on and off connected loads and meter the corresponding energy consumption as well as voltage, current and power of the connected devices.

The Domoticz Python plugin “HomematicIP Pluggable Switch and Meter (HmIP-PSM)” has been developed - described [here](#).

**For testing, the power & energy for the room MakeLab is controlled & measured.**

### Homematic IP HmIP-PSM (Plugin)

New hardware added using the plugin.

Idx	Name	Enabled	Type	Address	Port	Data Timeout
25	Schalt-Mess-Steckdose MakeLab	Yes	homematicIP Pluggable Switch and Meter (HmIP-PSM)	CCU-IP		Disabled

Showing 1 to 1 of 1 entries (filtered from 16 total entries)

Update Delete

Enabled:

Name: Schalt-Mess-Steckdose MakeLab

Type: homematicIP Pluggable Switch and Meter (HmIP-PSM)

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
Do not enable this option for devices that do not receive data!

homematicIP Pluggable Switch and Meter (HmIP-PSM) v1.0.3

- Switch the device On or Off.
- Measure, in regular intervals, the power (W), energy (Wh), voltage (V), current (mA).

Domoticz Devices Name (TypeName)

- Energy (kWh)
- Voltage (Voltage)
- Current (Current)
- Switch (Switch)

Configuration

- CCU IP address (default: CCU-IP)
- IDs (obtained via XML-API script <http://ccu-ip-address/addons/xmlapi/statelist.cgi> using the HomeMatic WebUI Device Channel, i.e. HmIP-PSM 0001D3C99C6AB3:3 (switch) or .6 (meter):
  - Device ID HmIP-PSM (default: 1418)
  - Datapoint ID SWITCH: STATE (default: 1451)
  - Datapoint IDs ENERGY(#4): ENERGY\_COUNTER, POWER, VOLTAGE, CURRENT as comma separated list in this order (default: 1467,1471,1473,1465)

CCU IP: CCU-IP

Device: 1418

Datapoint ID STATE: 1451

Datapoint IDs ENERGY: 1467,1471,1473,1465

Check Interval (sec): 60

Debug: False

## Domoticz Devices

The Domoticz devices are created by the plugin.

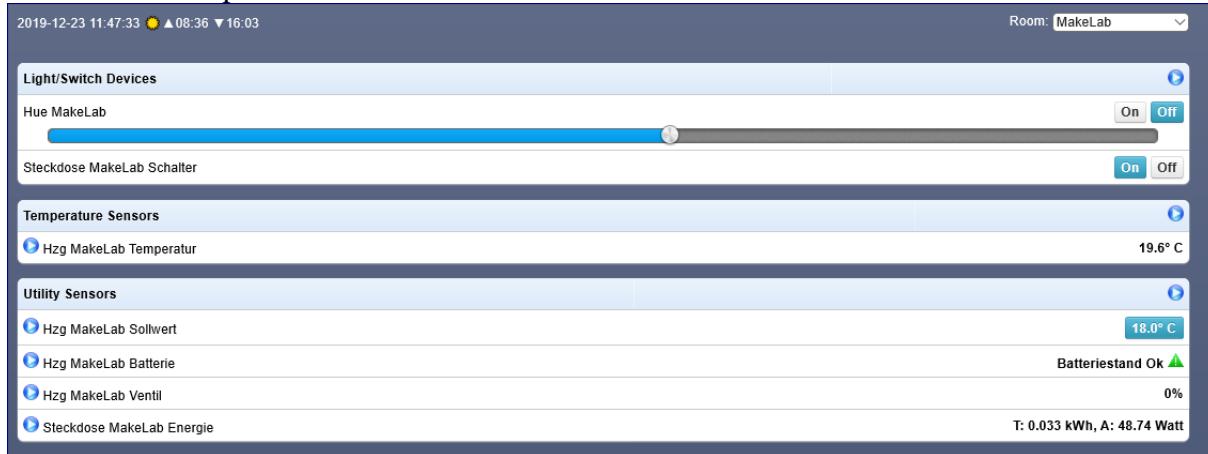
	226	Schalt-Mess-Steckdose MakeLab	00190004	4	Steckdose MakeLab Schalter	Light/Switch	Switch	On
	225	Schalt-Mess-Steckdose MakeLab	00190003	3	Steckdose MakeLab Ampere	General	Current	0.29 A
	224	Schalt-Mess-Steckdose MakeLab	00190002	2	Steckdose MakeLab Volt	General	Voltage	227.8 V
	223	Schalt-Mess-Steckdose MakeLab	00190001	1	Steckdose MakeLab Energie	General	kWh	6.595 kWh

*Note*

The devices with the IDX=226 & 223 are used.

## Roomplan MakeLab

For tests a room plan is defined for the room called “MakeLab”.



*Note*

The dashboard for the room makeLab shows the two devices Steckdose MakeLab Schalter & Steckdose MakeLab Energie.

## Solution Revolt SF-436.m

NOT USED – see issues.

For testing this solution, the power & energy for the room MakeLab is measured.

The Electric Usage is measured by a **433.92 MHz wireless plug Revolt SF-436.m**.

This device communicates with the [RFXCOM RFXtrx433E](#) with following requirements:

- RFXtrx type2 firmware,
- ELEC5 protocol,
- RSL option (Hardware > RFXTX433E > set mode).

### RFXCOM Test

#### Step 1

Plug the Revolt SF-436.m device in the wall plug (without pressing the green button!).

#### Step 2

Test if the Revolt device is recognized by the RFXtrx433E (with firmware version RFXtrx433E\_Ext2\_1025) using the RFXCOM RFXmngr.

If the device is recognized, the RFXmngr logs an entry with packettype POWER and subtype ELEC5Revolt.

```
Packettype=POWER,  
subtype=ELEC5Revolt,  
Sequencenbr=27, ID=32321,  
Voltage=233Volt, Current=0,07Ampere, Instantpower=10,9Watt, totalusage=0,06kWh,  
powerfactor=0,62, Frequency=50, Signallevel=8
```

#### Notes

- The Revolt SF-436.m is sending data almost every second.
- The ID is unique for this device and used by Domoticz (see next step).

As the device is recognized by RFXtrx433E, checked the Domoticz source for Revolt / ELEC5 support and found = pTypePOWER, sTypeELEC5, "Revolt".  
Moved on with the Domoticz Configuration.

## Domoticz Configuration

### Step 1

Allow Domoticz to receive new hardware for 5 minutes (Domoticz GUI > Setup > Settings > Hardware/Devices).

The Domoticz log lists initial entries for the new Power Device:

```
2019-05-30 13:23:01.301 (RFXtrx433e) Power (Unknown)
2019-05-30 13:23:04.359 (RFXtrx433e) Power (Unknown)
2019-05-30 13:23:07.869 (RFXtrx433e) Power (Unknown)
2019-05-30 13:23:13.989 (RFXtrx433e) Power (Unknown)
```

This means Domoticz recognizes the new Power device.

### Step 2

Switch to the Domoticz Devices List to check out the new devices

Recognized by Domoticz (Type, SubType, Data, Idx) are these four devices:

1. Type=Power, SubType=Revolt, Data=0.070kWh, IDX=167
2. Type=General, SubType=Voltage, Data=228 V, IDX=168
3. Type=General, SubType=Percentage, Data=0.73%, IDX=169  
*Note:* This value is the Power Factor.
4. Type=General, SubType=Percentage, Data=50%, IDX=170  
*Note:* This value is the frequency in Hz.

## Initial Devices List

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
⚡	167	RFXtrx433e	32321	0	Revolt Power	Power	Revolt	0.090 kWh
⚡	168	RFXtrx433e	32321	1	Revolt Voltage	General	Voltage	228 V
⚡	169	RFXtrx433e	32321	2	Revolt Powerfactor	General	Percentage	0.59%
⚡	170	RFXtrx433e	32321	3	Revolt Frequency	General	Percentage	50%

The **ID=32321** is the same as logged by the RFXmngr, which ensures this is the right device. The device **Type=Power, SubType=Revolt** is added as " Electricity MakeLab" (renamed from "Revolt Power"), the other devices are not used.

## Utility Tab Widget & Log with a few days





## MQTT

Example MQTT payload (topic "domoticz/out") for Device “Electricity MakeLab” with IDX=167.

```
{"Battery" : 255, "RSSI" : 8, "description" : "Energy (Wh) & Power (Watt) provided, every few seconds, by a Revolt SF-436 device. The Energy is computed from the Power.", "dtype" : "Power", "id" : "32321", "idx" : 167, "name" : "Electricity MakeLab", "nvalue" : 0, "stype" : "Revolt", "svalue1" : "7", "svalue2" : "720.00", "unit" : 0}
```

## Properties

- svalue1 = current Power usage (Watt)
- svalue2 = total Energy cumulated over all days (kWh).

## HTTP API Request

### Request

```
http://domoticz-ip:8080/json.htm?type=devices&rid=167
```

### JSON Response (extract)

```
{
  "app_version" : "4.10717",
  "result" : [
    {
      "CounterToday" : "0.230 kWh",
      "Data" : "0.720 kWh",
      "EnergyMeterMode" : "",
      "HardwareName" : "RFXtrx433e",
      "HardwareType" : "RFXCOM - RFXtrx433 USB 433.92MHz Transceiver",
      "ID" : "32321",
      "Name" : "Electricity MakeLab",
      "SubType" : "Revolt",
      "SwitchTypeVal" : 0,
      "Type" : "Power",
      "Unit" : 0,
      "Usage" : "7 Watt",
      "idx" : "167"
    }
  ],
  "status" : "OK",
  "title" : "Devices"
}
```

## Properties

- Usage = current power usage (Watt)
- Data = total energy cumulated over all days (kWh).

## HTTP API Information

```
Electricity (instant and counter)
/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=POWER;ENERGY
IDX = id device
POWER = current power
ENERGY = cumulative energy in Watt-hours (Wh)
Incrementing counter because type "Energy read : Computed"
```

## Issues

Whilst using this device, encountered several issues.

- 1) If the RFXtrx433E is more than about 10 meters range away from the device, then no data is received else working ok. Probably rather weak antenna.  
Explore how to improve the range ELSE consider other solution.
- 2) The device is causing high traffic on the 433.92MHz frequency. Sending almost every few seconds data.  
The Domoticz log is getting spoiled with data and of course the RFXtrx433E is rather busy.
- 3) Had been thinking about additional devices - but regarding issues 1 and 2 will not continue.

Put this solution On Hold for now.

# Email Control

## Purpose

To control Domoticz remote via Email, i.e. sent Domoticz commands to trigger actions.

Examples: Switch Outdoor light on, set thermostat setpoint, request status devices etc.

The idea behind this function is control Domoticz without the need to have access to the Domoticz system from the public internet.

*This solution is basically an experiment out of curiosity if it would be possible to remote access the Domoticz system.*

### Notes

1. The solution is rather technical as it requires knowledge of the Domoticz HTTP API or MQTT commands as well as the idx of devices and the device properties from JSON result. A thought on simplification could be
  - create email templates with subject & body.
  - an App (i.e. Android) acting as an email client submitting predefined commands.
2. **Security** is always an issue – checkout the hints for some thoughts.
3. There might be other solutions, but the one shared works fine so far.

... must say it is fun submitting an email with a Domoticz command and watching the actions.

# Solution

## Overview

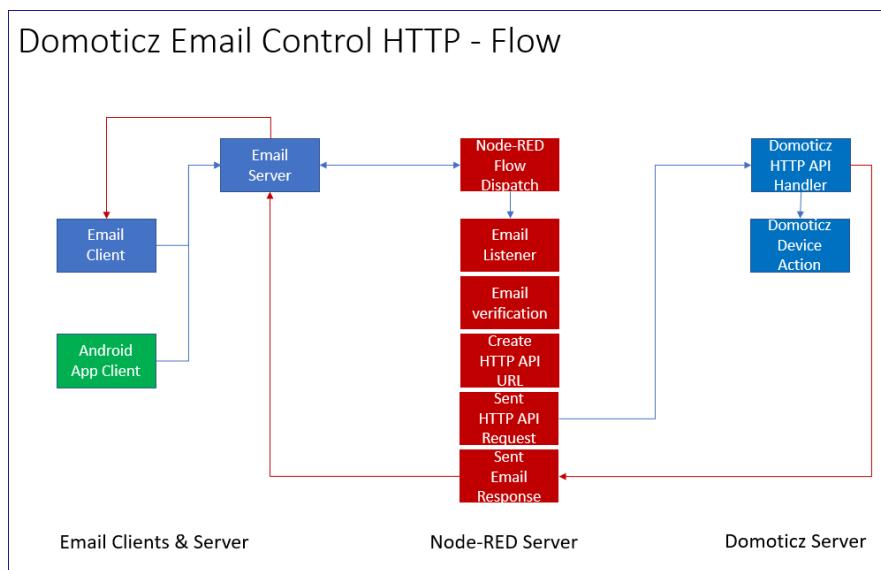
The solution uses a Node-RED flow (named Email Control), acting as kind of middleware, to handle Domoticz commands being sent via email (i.e. email client or App).

Required are the Node-RED nodes: [node-red-node-email](#) (thanks to the author).

There are two solutions developed based on MQTT or HTTP

The preferred solution is HTTP, because it offers more commands, but it is also the underlying Domoticz API.

The MQTT solution is not described further.



## Concept

### Email Client

An email (plain text) is created with a subject using the *prefix:description* line.

The prefix used is “DOMOTICZ:” (in uppercase).

The description can be any short text, i.e. Switch Light On.

The full subject would be “DOMOTICZ:Switch Light On”.

### Node-RED Flow

1. listens, every 60 seconds (value can be changed) to new email messages
2. checks first the fixed “from email address” (hardcoded in the flow, i.e your email address).
 

**For Security:** Only a single hardcoded email address is accepted. Use a dedicated email address.
3. checks next the email subject (start with a prefix, i.e. “DOMOTICZ:”)
 

**For Security:** Use a prefix which is only known to the sender.
4. parses the email body containing the HTTP API command and creates the HTTP API Request URL.
 

**For Security:** Add a key to the email body, which is checked and stripped for the submitted HTTP API Request URL.
5. submits the Domoticz HTTP API Request URL.

6. Handle HTTP response message, which is sent as email to a fixed “*to email address*” (i.e the same as the sender address).
7. The response email
  - a. subject starts with “RESPONSE:” and the description set as created
  - b. body contains the HTTP response in JSON format

#### Notes

- When using Domoticz Authorization over HTTP, then check if the "Authorization" HTTP request header" needs to be set (see documentation “Domoticz API/JSON URL's”).
- The email addresses “from” and “to” are hardcoded in the respective Node-RED nodes.
- **Security:** If the flow is not used, disable the Node-RED flow.

## Email Subject & Body

The email subject must start with prefix in uppercase: DOMOTICZ:

The email body, as plain text, contains a string with commands.

Any command as described in the [Domoticz API/JSON URL's manual](#) can be defined in the email body.

The HTTP API URL is created with the prefix and message payload added (in Node-RED function node *Create HTTP Request*)

```
"http://DOMOTICZ-IP:8080/json.htm?" + msg.payload;
```

### Examples

The first line is the email subject (starting with the prefix DOMOTICZ.) and the second line is the email body as plain text holding the API command. NNN=Idx of the device.

```
DOMOTICZ:Switch HUE Light ON  
type=command&param=switchlight&idx=NNN&switchcmd=On
```

```
DOMOTICZ:Switch HUE Light OFF  
type=command&param=switchlight&idx=NNN&switchcmd=Off
```

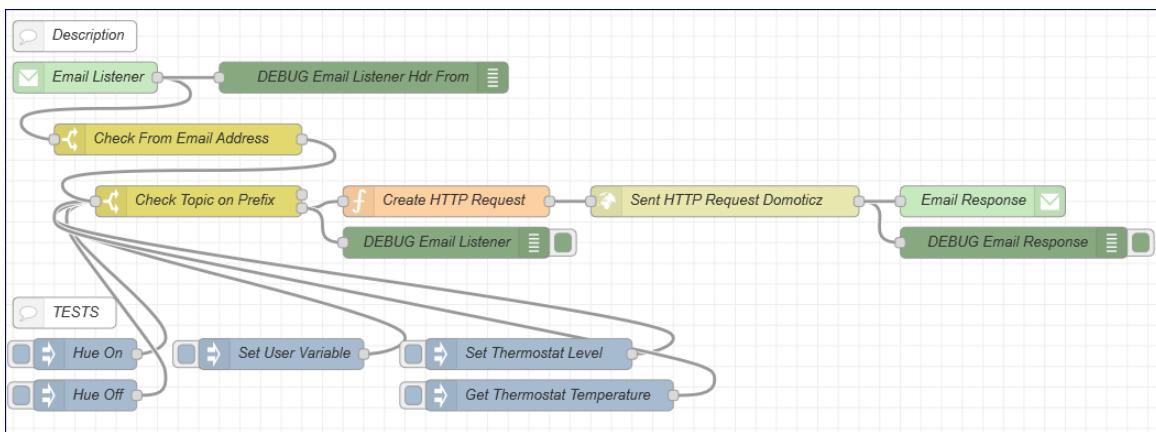
```
DOMOTICZ:Set User Variable AlertToEmail to 1  
type=command&param=updateuservariable&vname=TH_ALERTTOEMAIL&vtype=0&vvalue=1
```

```
DOMOTICZ:Get Status Uservariable TH_ALERTTOEMAIL  
type=command&param=getuservariable&idx=NNN
```

```
DOMOTICZ:Thermostat MakeLab Off  
type=command&param=switchlight&idx=NNN&switchcmd=Set%20Level&level=0
```

```
DOMOTICZ:Thermostat MakeLab 20 Degrees  
type=command&param=switchlight&idx=NNN&switchcmd=Set%20Level&level=30
```

## Node-RED Flow



### Node Email Listener (Type email in)

Listens to incoming emails automatically every 60 seconds (polling time) from the email server using the IMAP protocol, SSL, Port 993, Folder INBOX, Disposition Mark Read, Criteria Unseen.

#### Important

As given by the Node Description - this node only gets the most recent single email from the inbox, so set the repeat (polling) time appropriately.

Workaround:

- if getting no response within the polling timeframe after sending the email, then either submit again or decrease the polling time.
- Use a dedicated email address for this solution which does not receive any other emails than from the “from email address”.

### Node Check From Email Address (Type switch)

Check the header from email address. This depends on the email header as send by the provider email server. The message header is JSON parsed.

Example:

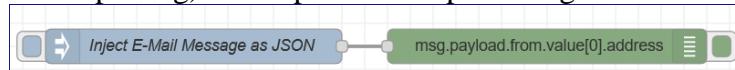
```
msg.header.from.value[0].address
```

to compare against a dedicated (=hardcoded) email address.

This ensures only emails are handled which are sent by an accepted email address.

#### Note

The node “DEBUG Email Listener Hdr From” can be used to determine the from address property (requires JSON parsing). A simple flow helps finding out:



Copy the output of debug note and insert a JSON string in the Inject node. Analyze the string on the from address. Set the JSON property in the debug node to determine the from address. If too complex then consider checking if the msg.payload contains the from email address string.

### Node Check Topic on Prefix (Type switch)

Check if the message topic starts with the defined prefix “DOMOTICZ:”. A JSONata exp is used = check substring first 9 characters match the prefix.

```
(  
$substring(topic, 0, 9) = "DOMOTICZ:"  
)
```

### Node Create HTTP Request (Type function)

Function node to create the message URL to be submitted to Domoticz.

The prefix DOMOTICZ: is replaced by RESPONSE: to indicate the HTTP API response is sent.

```
var t = msg.topic;  
msg.topic = t.replace("DOMOTICZ:", "RESPONSE:");  
// Set the URL  
msg.url = "http://domoticz-ip:8080/json.htm?" + msg.payload;  
// Return the message to be published by the mqtt out node  
return msg;
```

*Note*

If Node-RED is running on the same system as Domoticz, then “localhost” can be used as IP address.

### Node Sent HTTP Request Domoticz (Type http request)

Sent the message URL using GET method and return a UTF-8 string.

### Node Email Response (Type email)

Sent the HTTP API Request response as email.

The SMTP server uses port 587 without option “Use secure connection”, but with option “Use TLS?”. This is required for the email server used by the author.

The response from Domoticz is a JSON formatted string as email body.

If the result of the command is OK, then the email body is like:

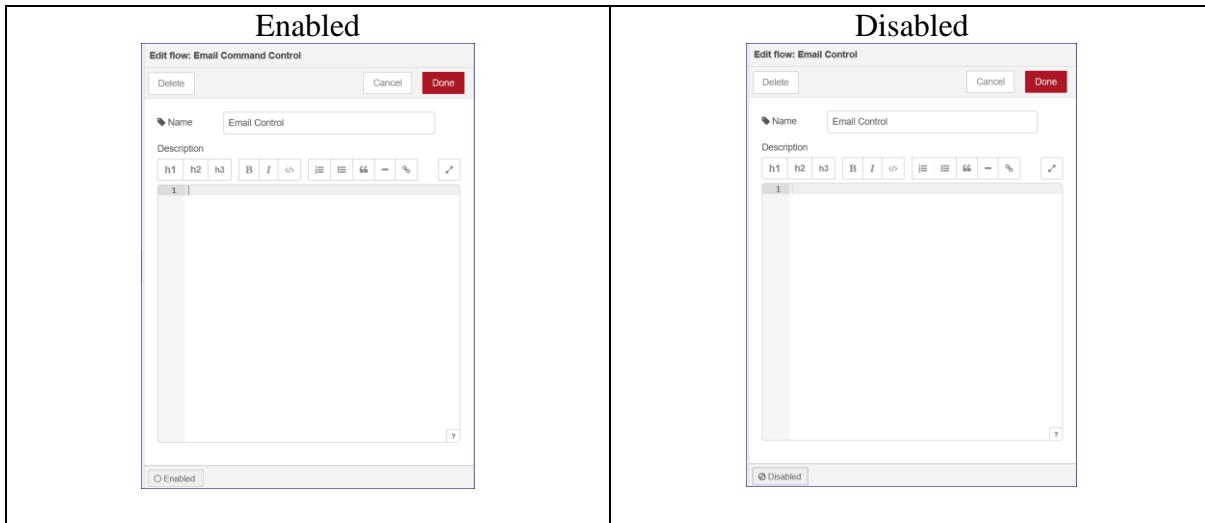
```
{"status" : "OK", "title" : "SwitchLight"}
```

If the result of the command is an ERROR, then the email body is like:

```
{"status" : "ERR"}
```

## Node-RED Flow Enabled or Disabled

If not using the flow, the flow can be disabled in Node-RED.  
Edit the flow and tick at the bottom of the dialog, the option Enabled or Disabled.



## Control Examples

### Domoticz E-Mail Control HTTP - Example Thermostat

#### Control Action

Set homematic thermostat level in room MakeLab to Off (Level 0).

#### E-Mail Subject & Body

DOMOTICZ:Thermostat MakeLab Off  
type=command&param=switchlight&idx=183&switchcmd=Set%20Level&level=0

*Note:* Thermostat device properties Level=Temperature; 0=Off,10=18°,20=19°,30=20°

#### Domoticz Log Entries

2019-11-07 09:59:01.798 Status: User: Admin initiated a switch command (183/MakeLab Thermostat - Setpoint/Set Level)

2019-11-07 10:00:34.918 (MakeLab Thermostat) T=21.7,S=4.5  
*Note:* The second entry confirms that the thermostat setpoint has been set to Off which is stated by S=4.5 (homematic specific value)

#### Node-RED Debug Log after the e-mail is sent

```
07/11/2019, 09:59:01 node: Create HTTP Request
function: (wearn)
"DOMOTICZ:Thermostat_MakeLab_Off"
07/11/2019, 09:59:01 node: Create HTTP Request
function: (wearn)
"RESPONSE:Thermostat_MakeLab_Off"
07/11/2019, 09:59:01 node: Create HTTP Request
function: (wearn)
"http://<ip-address>_0/json.htm?type=command&param=switchlight&idx=183&switchcmd=Set%20Level&level=0"
07/11/2019, 09:59:01 node: DEBPU Email Listener
DOMOTICZ:Thermostat_MakeLab_Off : msg.payload: string[8]
"type=command&param=switchlight&idx=183&switchcmd=Set%20Level&level=0"
07/11/2019, 09:59:01 node: DEBPU Email Response
RESPONSE:Thermostat_MakeLab_Off : msg.payload: string[9]
> "{<status> : "OK",<title> : "SwitchLight"}"
```

#### E-Mail Response Subject & Body

RESPONSE:Thermostat\_MakeLab Off
{
 "status": "OK",
 "title": "SwitchLight"
}

*Note:* This is a second e-mail as the first e-mail contains the request with subject DOMOTICZ:Thermostat MakeLab Off and the command string as plain text in the body.

## Domoticz E-Mail Control HTTP - Example Get User Variable

### Control Action

Get the status of the user variable TH\_ALERTTOEMAIL with IDX=14

### E-Mail Subject & Body

DOMOTICZ:Get Status Uservariable TH\_ALERTTOEMAIL  
type=command&param=getuservariable&idx=14

### Domoticz Log Entries

This request is not logged.

The e-mail response should be sufficient.

### Node-RED Debug Log after the e-mail is sent

```
07/11/2019, 10:59:41 node: Create HTTP Request
function : (wam)
"DOMOTICZ:Get Status Uservariable TH_ALERTTOEMAIL"
07/11/2019, 10:59:41 node: Create HTTP Request
function : (wam)
"RESPONSE:Get Status Uservariable TH_ALERTTOEMAIL"
07/11/2019, 10:59:42 node: Create HTTP Request
function : (wam)
"HTTP:// [REDACTED] /json.htm?type=command&param=getuservariable&idx=14"
07/11/2019, 10:59:43 node: DEBUG E-Mail Listener
DOMOTICZ:Get Status Uservariable TH_ALERTTOEMAIL: msg.payload : string[4]
"Type": "command&param=getuservariable&idx=14"
07/11/2019, 10:59:44 node: DEBUG E-Mail Response
RESPONSE:Get Status Uservariable TH_ALERTTOEMAIL: msg.payload : string[246]
"result": [{"LastUpdate": "2019-11-07 10:49:35", "Name": "TH_ALERTTOEMAIL", "Type": "0", "Value": "4", "Idx": "14"}], "status": "OK", "title": " GetUserVariable"}
```

### E-Mail Response Subject & Body

RESPONSE:Get Status Uservariable TH\_ALERTTOEMAIL
{  
 "result": [  
 {  
 "LastUpdate": "2019-11-07 10:49:35",  
 "Name": "TH\_ALERTTOEMAIL",  
 "Type": "0",  
 "Value": "4",  
 "Idx": "14"  
 }  
 ],  
 "status": "OK",  
 "title": " GetUserVariable"  
}

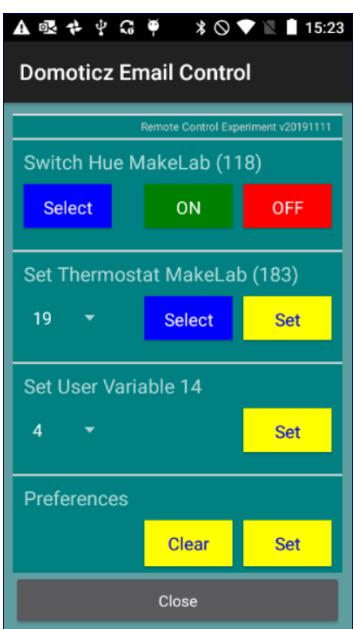
## Android Client App

An example of a simple app to test sending emails with commands (subject & body) to the Domoticz system via Node-Red as the middleware.

The Android app is created with the development tool [B4A](#) from Anywhere Software. B4A offers a simple way to develop native Android apps with the B4X language.

The app has a few controls (views) which are used to create the email with the subject and the body which is then sent using SMTP (Simple Mail Transfer Protocol). The SMTP parameter are hardcoded in the app except username & password which are requested prior sending the email.

The email response from the Node-RED flow, is not handled by the app but in the email client inbox. The email sent is also kept on the email server.

	
<p>The app uses a B4X customlistview with items related to a function.</p> <ol style="list-style-type: none"> <li>1. Switch Light / Switch Select a device and switch ON or OFF Example Hue MakeLab</li> <li>2. Set Thermostat Select a device and set the temperature from dropdown list between 0 (Off), 18, 19, 20</li> <li>3. Set User Variable with idx 14 This is a hardcoded example to set a value for the user var TH-ALERTTOEMAIL holding the level for sending alerts (email)</li> </ol>	<p>The preferences dialog uses the B4XPreferences library with a json configuration file (Files Folder &gt; prefdialog.json).</p>

# Event Monitor

## Purpose

To monitor the state change of selected devices to a text device with logging option.  
This function is used as to control or watch devices, for example security or time triggers.

Date	Data
2019-05-23 10:05:00	Security Main Door changed to Normal
2019-05-23 10:03:19	Security Main Door changed to Alarm + Tamper
2019-05-23 10:03:13	Security Main Door changed to Alarm + Tamper
2019-05-23 10:03:09	Security Main Door changed to Alarm + Tamper

*Note*

Just a few entries as an example.

## Domoticz Configuration

Create a text device to display the events.

Domoticz GUI > Setup > Hardware > Hardware VirtualSensors > Press Create Virtual Sensors

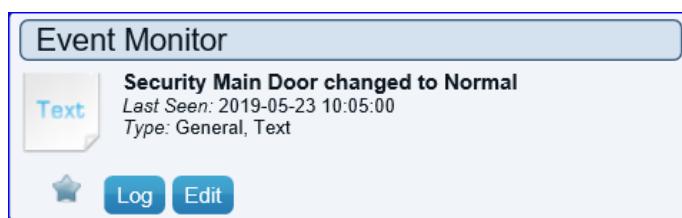
Name: Event Monitor

Sensor Type: Text

Domoticz GUI > Setup > Devices > Scroll down to the last list entry.

A new device is listed:

- Idx: 161
- Hardware: VirtualSensors
- Name: Event Monitor
- Type: General
- SubType: Text
- Data: Hello World



## Automation Script

The event, defined as a dzVents Lua script, monitors the state change of selected devices (by their idx), and updates the text of the device Event Monitor (IDX=161).

Create a new Event type dzVents Lua: GUI > Setup > More Options > Events

### Shared helper function: global\_data

Add a function to update the text of the Event Monitor device.

```
(in chapter Helpers)
IDX_EVENTMONITORMSG = 161,

(in chapter Messages)
-- Update the event monitor message with text
eventmonitormsg = function(domoticz, msg)
    domoticz.devices(domoticz.helpers.IDX_EVENTMONITORMSG).updateText(msg)
end
```

### Event name: event\_monitor

```
return {
  on = {
    devices = {
      -- list of the idx of the devices monitored
      -- update accordingly
      110,111,112,113,118,160
    }
  },
  execute = function(domoticz, device)
    local message = '' .. device.name .. ' changed to ' .. device.state
    domoticz.helpers.eventmonitormsg(domoticz, message)
  end
}
```

<TODO>

Check if possible, to define a user variable (type String) with the idx of the devices to monitor. Example:

Name: DEF\_EVENTMONITOR\_DEVICES

String: 110,111,112,113,118,160

Update the dzVents Lua script to loop over the devices.

```
on = {
  devices = {
    -- list of the idx of the devices monitored
    -- update accordingly
    Loop over user variable DEF_EVENTMONITOR_DEVICES
  }
},
```

# Maintenance

Clear the device log from time-to-time.

## Option Domoticz GUI

Tab Utility > Widget Event Monitor > Log > Clear

## Option SQLite3 Command

*Note*

Requires [SQLite3 Package](#) to be installed.

*Example*

1. Open a terminal.
2. Login as user Pi
3. Change to the domoticz database folder  
cd /home/pi/domoticz
4. Start sqlite3 command  
sqlite3
5. Select device log entries to check if there are entries  
select count(\*) from lightinglog where devicerowid=161;
6. Delete device log entries  
delete from lightinglog where devicerowid=161;
7. Select device log entries to check if entries are deleted  
select count(\*) from lightinglog where devicerowid=161;

```
Login Using username "pi".
cd domoticz
sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
sqlite> select count(*) from lightinglog where devicerowid=161;
46
sqlite> delete from lightinglog where devicerowid=161;
sqlite> select count(*) from lightinglog where devicerowid=161;
0
```

# Garage Door Monitor (HMIP-SWDM)

## Purpose

- To monitor open & close state of the garage door
- To check daily at 20:00 if the garage door state is open and notify via email

## Solution

### RaspberryMatic

The solution makes use of the Homematic IP Window/Door Contact with magnet device (HMIP-SWDM).

The HMIP-SWDM is integrated in the RaspberryMatic system running a Homematic CCU3 (see chapter Explore [RaspberryMatic](#) – must read to learn adding & changing devices).

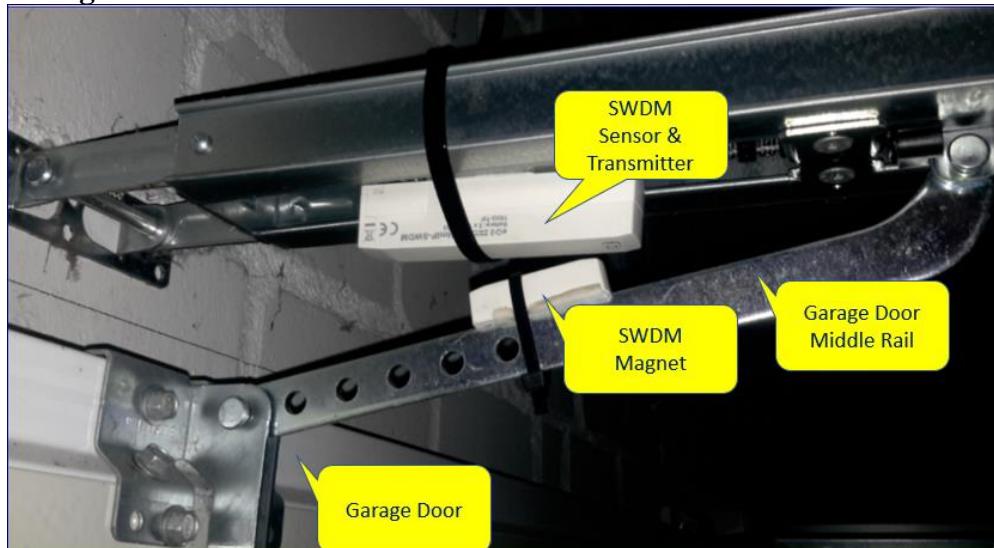
The HMIP-SWDM is mounted inside the garage door using the middle rail. The max distance between magnet and the sensor is ~5mm to ensure a signal is generated when moving the garage door = detecting the magnet. If the magnet is detecting, the LED of the sensor flashes short.

### Domoticz

A Domoticz virtual sensor type Alert, indicates the state of the HMIP-SWDM device:  
Level 1 = Green = Garage Door state closed, Level 4 = Red = Garage Door state open.

## Screenshots

### Garage Door with SWDM Device mounted inside



## RaspberryMatic SWDM Device

## Domoticz Alert Device

## Process Flow Steps

1. RaspberryMatic listens to HMIP-SWDM device channel state changes (Homematic Script).  
Channel status: HmIP-SWDM 001558A99D5A78:1 when open trigger when changed  
...
2. On channel state change, update the Domoticz Alert Sensor via HTTP API Request (Homematic Script)
3. Domoticz Alert Sensor Level indicating garage door state:  
Green = Closed, Red = Open
4. Domoticz Automation Event (dzVents) checks daily at 20:00 garage door state and notifies via email if open.

## Homematic Script

Source: garage\_door\_monitor.script

```
! Function: Garage Door Monitor
! Ensure to escape special characters
! 20200821 rwbl

! Get the actual date and time DD.MM HH:MM
string actDate = system.Date("%d.%m"); ! sDate = "09.08"; "%d.%m.%Y" = "09.08.2020";
string actTime = system.Date("%H:%M"); ! sTime = "07:32"; "%H:%M:%S" = "07:32:00";

! RaspMatic Device
! Get the state & object of the SWDM device via XMLAPI request: http://ccu-
ip/config/xmlapi/statelist.cgi
! <datapoint name="HmIP-RF.001558A99D5A78:1.STATE" type="STATE" ise_id="3622" value="0" valuetype="16"
valueunit="" timestamp="0" operations="5"/>
string swdmState = dom.GetObject("HmIP-RF.001558A99D5A78:1.STATE").State(); ! 0=Closed, 1=Open

! Domoticz Device
! A Domoticz virtual sensor type Alert is used to how the SWDM state
! 324,VirtualSensors,Garagentor,General,Alert,No Alert!
```

```

string alertIdx = 324; ! Type: General; SubType: Alert
string alertLevel = "0";
string alertState = "Unknown";

! Check SWDM state 0=Closed,1=Open
if (swdmState == 0) {
    alertLevel = "1"; ! Green
    alertState = "Geschlossen";
}
if (swdmState == 1) {
    alertLevel = "4"; ! Red
    alertState = "Ge%C3%BCffnet";
}
string alertText = alertState#"%20(" # actDate # "%20" # actTime # ")";
! string alertText = "Garagentor%20#"alertState#"%20(" # actDate # "%20" # actTime # ")";
WriteLine(alertText);

! Build the Domoticz http rest request url to update the device
! /json.htm?type=command&m=udevice&idx=IDX&nvalue=LEVEL&svalue=TEXT
! Level: 0=gray, 1=green, 2=yellow, 3=orange, 4=red
string cAmp = "&";
string urlBase = "'http://domoticz-ip:8080/json.htm'; ! Production
string urlRequest =
urlBase#"?type=command#cAmp#param=udevice#cAmp#"idx="#alertIdx#cAmp#"nvalue="#alertLevel#cAmp#"svalu
e="#alertText#"";;
WriteLine(urlRequest);

! Run the command without a return result
cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);

```

## Test Execute Script

```

Geschlossen%20(21.08%2009:07)
'http://domoticz-
ip:port/json.htm?type=command&param=udevice&idx=324&nvalue=1&svalue=Geschlossen%20(21.08%2009:07)'

```

## Automation Script

```

--[[[
garage_door_check.dzvents
Check daily at 2000, if the garagedoor is closed. Send email notification if open.
2000820 rwbl
--]]

local IDX_GARAGEDOOR_ALERT = 324
local DOOROPEN_STATE = 4      -- alert level 4 red

return {
  on = {
    timer = {'at 20:00',}
  },
  execute = function(domoticz, timer)
    -- Get state garagedoor alert device
    local state = domoticz.devices(IDX_GARAGEDOOR_ALERT).nValue
    if (state == DOOROPEN_STATE) then
      domoticz.log("Garagedoor is open. Please Close...")
      -- subject,message,mailto with cc
      domoticz.email(
        'Garagentor',
        'Das Garagentor ist noch offen. Bitte umgehend schließen.',
        'name1@email; name2@email')
    end
  end
}

```

# Hardware Monitor Raspberry Pi

## Purpose

To monitor hardware data for the Raspberry Pi (RPi) devices used in this Home Automation system solution.

### RPi Data Monitored

CPU Usage %, CPU Temperature °C, Discspace Used %, RAM Used %.

### RPi Devices (Name)

Domoticz Production (pidopro), Domoticz Development (pidodev), volkszaehler (pivz), Homematic (pihm - planned).

The hardware data for all the RPi's is displayed in the Domoticz development system.

Screenshots of the Hardware Monitor Dashboard & Floorplan.

**Temperature Sensors**

pidopro CPU Temp	49.4°C
pidodev CPU Temp	48.9°C
pivz CPU Temp	37.932°C

**Utility Sensors**

pidopro CPU Usage	2.23%
pidopro RAM Usage	24.5%
pidopro Disc Usage	32.61%
pidodev CPU Usage	2.17%
pidodev RAM Usage	47.84%
pidodev Disc Usage	82.25%
pivz CPU Usage	2%
pivz RAM Usage	46.493%
pivz Disc Usage	16%

**Domoticz Production**

CPU Temp	CPU Usage	RAM	Disc
49.4°C	2.14%	24.2%	32.61%

**Domoticz Development**

CPU Temp	CPU Usage	RAM	Disc
48.3°C	2.56%	43.35%	62.24%

**volkszaehler**

CPU Temp	CPU Usage	RAM	Disc
37.532°C	1%	46.6135%	16%

**homematic**

CPU Temp	CPU Usage	RAM	Disc

## Solution

The data provided by each RPi: CPU Temp, CPU Usage, RAM Usage, Disc Usage.

For the RPis running a Domoticz instance, data from the hardware "Motherboard Sensors" is used.

The devices are named as mentioned, added to the Domoticz instance roomplan "Hardware Monitor" which is used as a dashboard on that instance including a floorplan.

For the other RPis, a webserver solution "hwmon" is used to gather the hardware data.

The Domoticz Development system:

- has 4 virtual sensor devices (device name data) for each of the RPis, i.e. pivz CPU Temp, pivz CPU Usage etc.
- has a roomplan "Hardware Monitor" with all the hardware monitor devices
- has a flooplans "Hardware Monitor" using the roomplan "Hardware Monitor"
- gathers the data in regular intervals via dzVents Lua script events (one for each RPi), i.e. event name: hwmon-pivz, hwmon-pidopro.
- displays the data in the dashboard and floorplan

### Node-RED

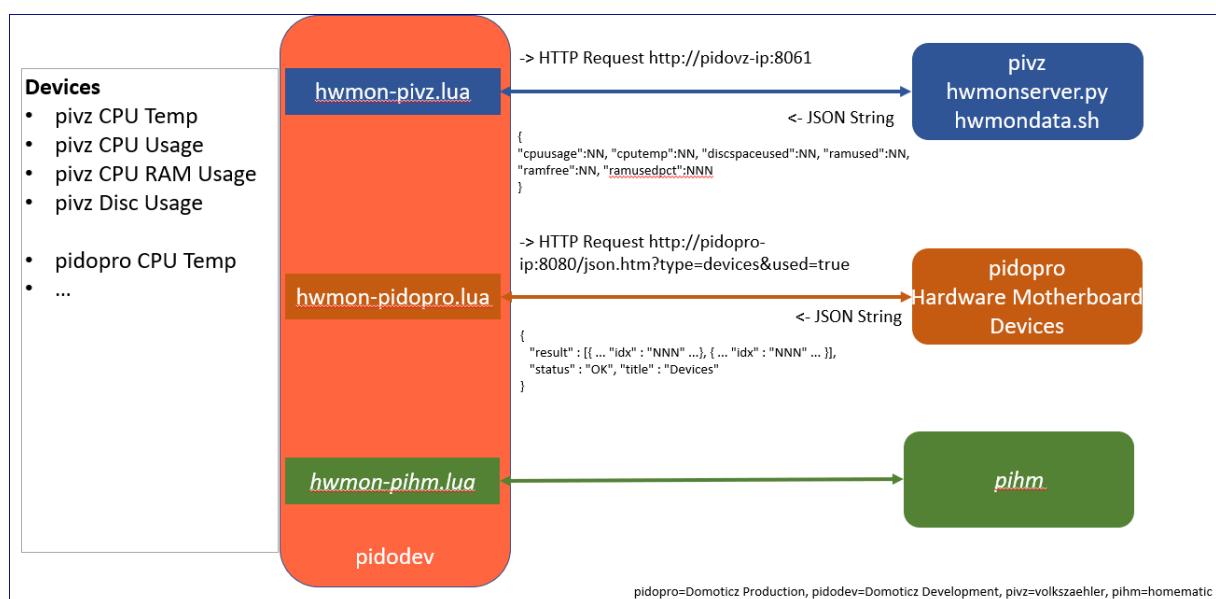
Tested various Node-RED flows for pivz data.

- Simple flow with a HTTP request node, triggered every 5 minutes by an Inject node, gathers the data and logs.
- Dashboard UI more complex flow with gauges and charts grouped by data property.

#### Note

For the latest version of the source code scripts & flows check the source code files.  
Only the pseudo code is shared.

### Flow



## RPi volkszaehler

Device name: pivz

This solution, called hwmon, has various scripts:

1. Bash script, *hwmondata.sh*, gets the data and outputs a JSON string. This script can be run from the CLI or via webserver request.
2. Python3 webserver, *hwmonserver.py*, running as a service, *hwmon.service* returns the JSON string on the HTTP API Request by running the as script.
3. Domoticz: dzVents Lua script opens the HTTP API Request URL and parses the HTTP response JSON string into Domoticz Virtual Sensors devices.

The files *hwmondata.sh*, *hwmonserver.py*, *hwmon.service* are in folder */home/pi/scripts*. Included are also tmp files generated.

Example of the JSON string returned by the bash script, which is called by the webserver and parsed by the dzVents Lua script event “hwmon-pivz” to update the Domoticz devices:

```
{"cpuusage":2,"cputemp":37.394,"discspaceused":16,"ramused":234,"ramfree":274,"ramusedpct":46.063}
```

### Bash Script

File: *hwmondata.sh*

Folder: */home/pi/scripts*

#### Pseudo Code

Run rpi commands to obtain system data in single var:

1. CPU Usage %
2. CPU Temp C
3. RAM Usage %
4. Disc Usage %

Build and output the JSON string.

#### Note

This Bash script is handy to use for a quick check by running from the CLI:

```
pi@vz:~ $ cd scripts  
pi@vz:~/scripts $ ./hwmondata.sh  
{"cpuusage":3,"cputemp":37.932,"discspaceused":16,"ramused":237,"ramfree":262,"ramusedpct":47.495}
```

## Python3 Webserver

File: hwmonserver.py

Folder: /home/pi/scripts

Python version: 3.7

### Pseudo Code

Run and wait for incoming HTTP API Requests.

Handle request by executing Bash script *hwmondata.sh*.

Create and return the HTML response

- code 200
- header content-type application/json
- content JSON string from output *hwmondata.sh*.

### Service hwmon.service

The webserver runs as a service from folder /home/pi/scripts.

Create the service file

```
nano hwmon.service
[Unit]
Description=Hardware Monitor Service
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u hwmonserver.py
WorkingDirectory=/home/pi/scripts
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

### Copy the service file to the systemd folder

```
sudo cp hwmon.service /etc/systemd/system/hwmon.service
```

### Start service

```
sudo systemctl start hwmon.service
```

### Check service is running

```
ps ax | grep hwmonserver
2299 ? Ss 0:00 /usr/bin/python3 -u hwmonserver.py
```

### Stop service

```
sudo systemctl stop hwmon.service
```

### Start Service at Boot

```
sudo systemctl enable hwmon.service
```

The `systemctl` command can also be used to restart the service or to disable. Example output:

```
sudo systemctl enable hwmon.service
Created symlink /etc/systemd/system/multi-user.target.wants/hwmon.service →
/etc/systemd/system/hwmon.service.
```

### Issues

If there are issues to copy the service file after a change, ensure

1. to stop the server: `sudo systemctl stop hwmon.service`
2. reload the deamon: `sudo systemctl daemon-reload`

## Automation Script

Event name: hwmon-pivz (embedded in the domoticz database).

### Pseudo Code

Place a HTTP API Request to the *pivz webserver* every minute triggered by a timer.

The HTTP request has a unique callback.

Handle the HTTP JSON response by updating the Domoticz Devices.

### Domoticz Log

```
2019-06-19 09:30:00.542 Status: dzVents: Info: ----- Start internal script: hwmon-rpi-vz:, trigger:  
every minute  
2019-06-19 09:30:00.542 Status: dzVents: Info: ----- Finished hwmon-rpi-vz  
2019-06-19 09:30:01.138 Status: dzVents: Info: Handling httpResponse-events for: "hwmonrpivz"  
2019-06-19 09:30:01.138 Status: dzVents: Info: ----- Start internal script: hwmon-rpi-vz:  
HTTPResponse: "hwmonrpivz"  
2019-06-19 09:30:01.144 Status: dzVents: Info: CPU Usage % = 1.5  
2019-06-19 09:30:01.144 Status: dzVents: Info: CPU Temp C = 37.932  
2019-06-19 09:30:01.144 Status: dzVents: Info: Disc Used % = 16  
2019-06-19 09:30:01.144 Status: dzVents: Info: RAM Used % = 46.063  
2019-06-19 09:30:01.163 Status: dzVents: Info: ----- Finished hwmon-rpi-vz
```

## RPi Domoticz Production

Device name: pidopro

A dzvents Lua script event is running on the Domoticz development system, gathers all used devices data from the Domoticz production system.

The hardware monitor devices (#4) are selected and used to update the devices on the Domoticz development system.

### Automation Script

Event name: hwmon-pidopro (embedded in the domoticz database).

#### Pseudo Code

Place a HTTP request to the Domoticz production system (pidopro) every minute triggered by a timer.

The request obtains all data from the used devices.

The HTTP request has a unique callback.

Handle the HTTP JSON response by

- selecting a device by its idx
- get the data and update the Domoticz device on the Domoticz development system

## RPi Status Indicator

### Purpose

Check the device “RPi CPU Usage” and indicate the state using an RGB LED connected to GPIO.

#### Note

1. The solution is open for enhancements with more checks and notifications (see Automation Script)
2. In addition to the RGB LED as indicator, an LCD display is an option to display and also indicate the RPi status (see next [RPi LCD Hardware Info](#) and [Hardware Monitor](#))

### Solution

An automation event sets the state of the RGB LED to:

- green, full brightness, no blink: if CPU Usage < threshold
- red, full brightness, blinking: if CPU Usage => threshold

The threshold is hardcoded in the Automation script. Using a User Variable provides more flexibility in setting the threshold.

This solution is based upon [RGB LED Plugin Interface](#).

#### In a nutshell

If the CPU Usage triggers a RGB LED state change, then the text of the RGB LED Plugin Text Device is set (by an Automation Event).

This change is handled by the RGB LED Plugin onModified function, which sets the color of the RGB LED via the GPIO Zero library (Python).

The device “RPi CPU Usage” (IDX=13) is just an example from the Raspberry Pi hardware devices:

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
14	RPi Hardware	0001	1	RPi Temperature	Temp	LaCrosse TX3	49.7 C
13	RPi Hardware	0000044D	1	RPi CPU Usage	General	Percentage	0.83%
17	RPi Hardware	0000044E	1	RPi HDD /boot	General	Percentage	21.16%
18	RPi Hardware	0000044F	1	RPi HDD /	General	Percentage	25.26%
15	RPi Hardware	0000044C	1	RPi Memory Usage	General	Percentage	10.95%
16	RPi Hardware	000000DC	1	RPi Process Usage	General	Custom Sensor	42.23 MB

## Automation Script

Source: gpiozero\_hardware\_monitor.dzvents

```
-- gpiozero_hardware_monitor.dzvents
-- Trigger: device change
-- Check the CPU_Usage and indicate the state using an RGB LED connected to GPIO.

-- device idx
local IDX_CPUUSAGE = 13      -- cpu usage device monitored
local IDX_RGBLCOLOR = 12      -- text device created by the rgb plugin; updated by the cpu usage state
(see below)

-- cpu high threshold % - value hardcoded but can also be set via user variable
local TH_CPUUSAGE = 9
-- rgb led states red-green-blue-brightness-blink
-- color: 0-255, brightness: 0-100%, blink 0 (OFF) | 1 (ON)
local STATE_CPUUSAGE_OK = "0-255-0-100-0"    -- green, full brightness, no blink
local STATE_CPUUSAGE_HIGH = "255-0-0-100-1" -- red, full brightness, blinking

return {
  on = {
    devices = {IDX_CPUUSAGE}
  },
  execute = function(domoticz, device)
    state_color = domoticz.devices(IDX_RGBLCOLOR).text
    if device.idx == IDX_CPUUSAGE then
      if tonumber(device.sValue) > TH_CPUUSAGE and state_color ~= STATE_CPUUSAGE_HIGH then
        newcolor = STATE_CPUUSAGE_HIGH
        domoticz.devices(IDX_RGBLCOLOR).updateText(newcolor)
        domoticz.log(string.format("Threshold HIGH: Color set:%s",newcolor), domoticz.LOG_INFO)
      end
      if tonumber(device.sValue) <= TH_CPUUSAGE and state_color ~= STATE_CPUUSAGE_OK then
        newcolor = STATE_CPUUSAGE_OK
        domoticz.devices(IDX_RGBLCOLOR).updateText(newcolor)
        domoticz.log(string.format("Threshold OK: Color set:%s",newcolor), domoticz.LOG_INFO)
      end
    end
  end
}
```

# RPi LCD Hardware Info

## Purpose

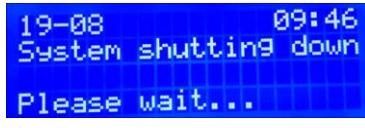
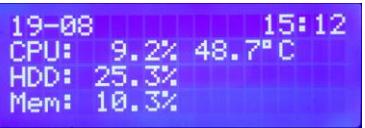
To display selected hardware information on a I2C LCD 2004 which is taken from the devices as defined by the Domoticz Hardware Type Motherboard Sensors:

Idx	Hardware	ID	Unit	Name	Type	SubType	
13	RPi Hardware	0000044D	1	RPi CPU Usage	General	Percentage	9.48%
14	RPi Hardware	0001	1	RPi CPU Temperature	Temp	LaCrosse TX3	50.2 C
15	RPi Hardware	0000044C	1	RPi Memory Usage	General	Percentage	9.76%
16	RPi Hardware	000000DC	1	RPi Process Usage	General	Custom Sensor	31.89 MB
17	RPi Hardware	0000044E	1	RPi HDD /boot	General	Percentage	21.16%
18	RPi Hardware	0000044F	1	RPi HDD /	General	Percentage	25.27%

Used are the devices with idx 13,14,15,18.

## Solution

An automation event sets the hardware information on the LCD. In addition, system start, and shutdown is displayed on the LCD.

System Start	System Shutdown	System Information
		
LCD set by dzVents trigger on.system = 'start'	LCD set by dzVents trigger on.system = 'stop'	LCD updated by dzVents trigger on.timer = 'every minute'

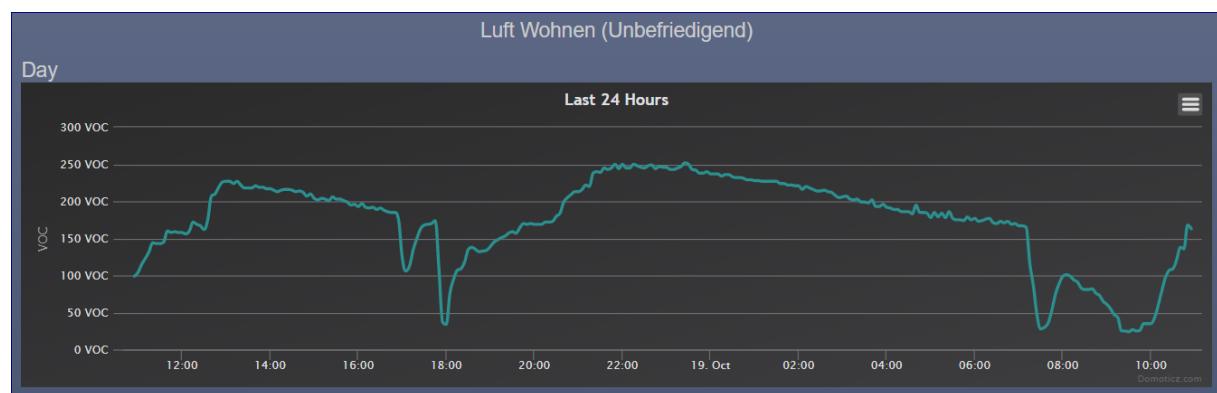
## Automation Script

See source lcdi2c\_hwmonitor\_v2.dzvents.

# Indoor Air Quality (Tinkerforge, Script)

## Purpose

- To measure, in regular intervals, the Indoor Air Quality Index & Condition of volatile organic components (VOC).
- To display the Air Quality Index and Condition in a Domoticz device.
- To indicate the Air Quality Index by setting a specific color of a Hue Bulb.
- To enable additional or alternate functionality, i.e.
  - send notification alert if Air Quality Index threshold exceeded
  - show Air Quality Index color by RGB LED instead Hue Bulb
  - show Air Quality Index color periodically, i.e. every hour for 1 minute, by a Hue Bulb which is normally used as a light



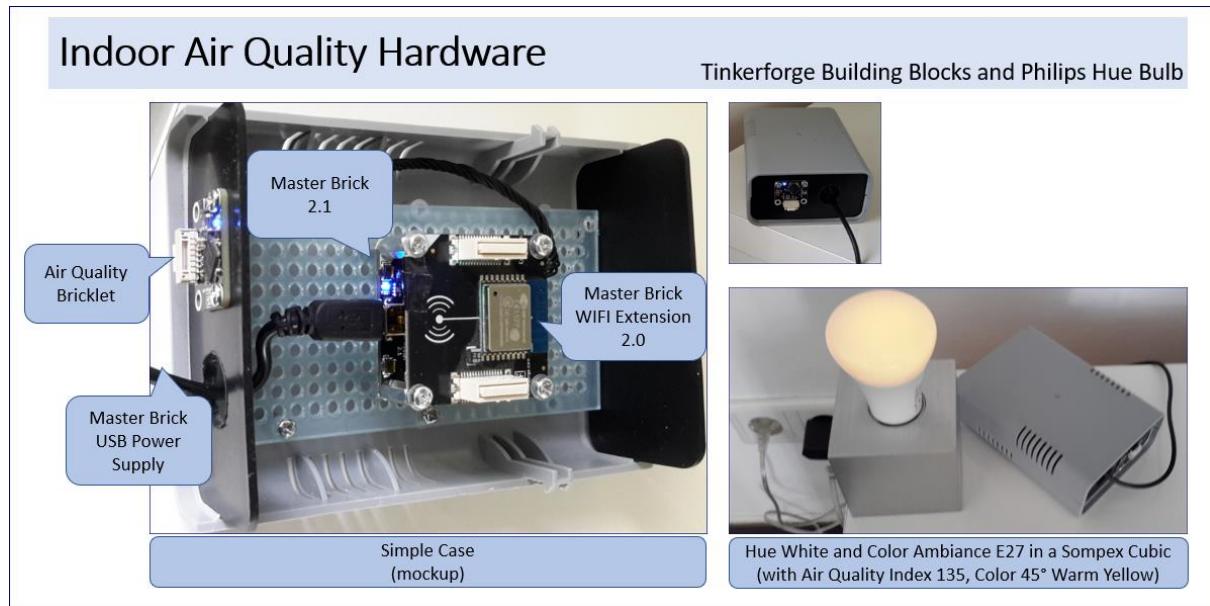
## Solution

The solution is based on hardware [Tinkerforge](#) Building Blocks with Python API bindings handled by Domoticz Automation Event dzVents Lua.  
If not familiar with Tinkerforge, recommend reading chapter Explore > [Tinkerforge](#).

### The Indoor Air Quality

- is measured using a Tinkerforge Master Brick + WiFi Extension with a Tinkerforge Air Quality Bricklet connected,
- the Indoor Air Quality Index value is assigned to a Domoticz Custom Sensor (with unit VOC),
- the Indoor Air Quality Index (0-500) specific color is set for a Philips Hue Bulb via a Domoticz Color Switch, RGBWW, Dimmer using the Hue Color Wheel.

## Screenshot of the solution placed in the living room



## Hardware

- Tinkerforge [Master Brick](#) 2.1 (Firmware 2.4.10) with [WiFi Extension](#) 2.0
- Tinkerforge [Air Quality Bricklet](#) (Firmware 2.0.5)
- Tinkerforge [USB Power Supply](#) + Mini USB Cable 90cm
- Raspberry Pi Case for the components (or use any other case)
- Tinkerforge [Air Quality Bricklet Case](#)
- Tinkerforge 7p-10p 15cm wire connect the Air Quality Bricklet with Master Brick
- Tinkerforge Mounting Plate 22x10 (12x6cm)
- Hue White and Color Ambiance Bulb

## Software

- Tinkerforge [Brick Daemon](#) (2.4.1)
- Tinkerforge [Brick Viewer](#) (2.4.14)
- Tinkerforge [Python API Bindings](#) (2.1.26)
- Python Script to get & send Air Quality Bricklet data to Domoticz  
`/home/pi/domoticz/scripts/python/indoor_air_quality.py`
- Domoticz Automation Event [dzVents](#) to read the data and update a Custom Sensor  
`indoor_air_quality.dzvents`

### Note

Hardware and Software versions may be subject to change.

Look up the [Tinkerforge Announcements](#).

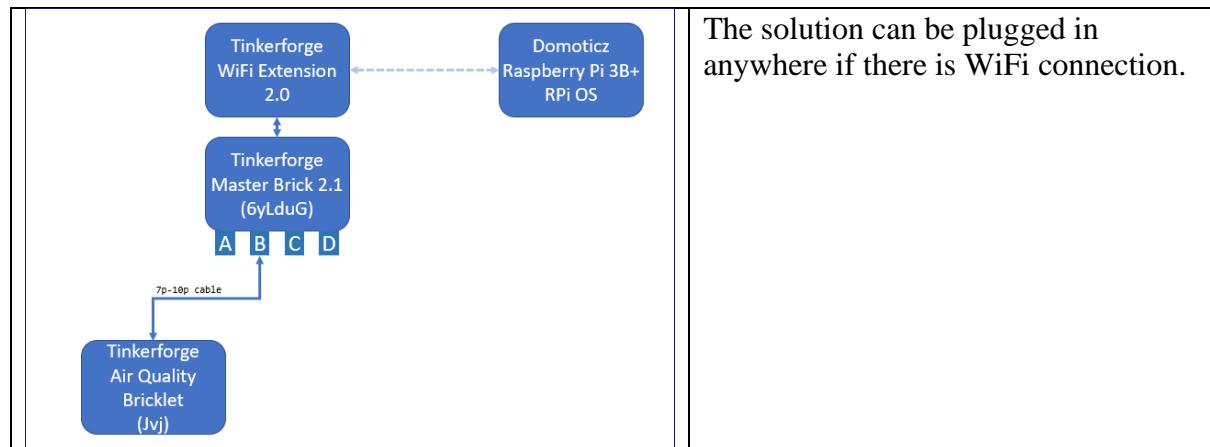
## Air Quality Bricklet

**Air Quality Bricklet** measures the IAQ Index (ppm), IAQ Condition, IAQ Accuracy, Air Pressure (mbar), Humidity (%), Temperature (C).

Referencing Tinkerforge [documentation](#): *The IAQ index is a measurement for the quality of air. To calculate the IAQ index the Bricklet detects ethane, isoprene (2-methylbuta-1,3-diene), ethanol, acetone, and carbon monoxide (often called VOC, volatile organic components) by adsorption. These gas measurements are combined with the measurements of air pressure, humidity, and temperature to calculate the final IAQ index.*

- **IAQ Condition Levels** (6) range=condition (indicative color):  
0-50=Good (green), 51-100=Moderate (yellow), 101-150=Unhealthy sensitive groups (orange), 151-200=Unhealthy (red), 201-300=Very Unhealthy (purple), 301-500=Hazardous (maroon).
- **IAQ Index Accuracy Levels** (4): Unreliable, Low, Medium, High.

## Wiring



## Tinkerforge Setup

### Software

The Tinkerforge software is installed on the Raspberry Pi by following the Tinkerforge Installation Instructions ([Raspberry Pi](#) and [Python API Bindings](#)).

For the ease of use, created a bash script, located in folder /home/pi/tinkerforge, to:

- Update Raspberry OS
- Install Tinkerforge Python API Bindings
- Install Brick Daemon (brickd)
- Install Brick Viewer (brickv)

Source: Tinkerforge-update.sh

```
# Bash script to update Tinkerforge Brick Daemon & Viewer
# Make script executable: sudo chmod +x tinkerforge-update.sh
# Run script: ./update-tinkerforge.sh
echo -----
```

```

echo Tinkerforge Update
echo -----
echo Pre Work...
mkdir /home/pi/tinkerforge
cd /home/pi/tinkerforge
sudo rm *.deb
echo Done
echo -----
echo Raspberry OS Update
echo -----
sudo apt-get update
echo Done
echo -----
echo Python Bindings
echo -----
echo Installing...
sudo pip3 install tinkerforge
echo Done
echo -----
echo Brick Daemon
echo -----
sudo apt-get -y install libusb-1.0-0 libudev0 pm-utils
sudo wget https://download.tinkerforge.com/tools/brickd/linux/brickd_linux_latest_armhf.deb
sudo dpkg -i brickd_linux_latest_armhf.deb
sleep 1
echo Checking brickd process...
ps -ef|grep brickd
echo Done
echo -----
echo Brick Viewer
echo -----
sudo apt-get install -y python3 python3-pyqt5 python3-pyqt5.qtopen gl python3-serial python3-tz python3-tzlocal
sudo wget https://download.tinkerforge.com/tools/brickv/linux/brickv_linux_latest.deb
sudo dpkg -i brickv_linux_latest.deb
echo Done
echo -----
echo Cleanup
echo -----
sudo apt-get clean
sudo apt-get -y autoremove
sudo rm /home/pi/tinkerforge/*.deb
echo Done
echo -----
echo Completed

```

## Firmware

The [Brick Viewer](#) is used to connect to the Master Brick and check or update to the latest version of the firmware for the building blocks used.

It is important to ensure the latest firmware versions are used.

The Brick Viewer lists under Tab Setup, the name, UID and Firmware version of all building blocks (Bricks, Extension, Bricklets...).

## Master Brick + WiFi Extension

The Master Brick WiFi Extension is configured using the Tinkerforge Brick Viewer (Tab Master brick 2.1) as described [here](#).

General	
Mode	Client
Port	4223
WebSocket Port	4280
WebSite Port	80
Disable Web Interface	NO = Not checked

PHY Mode	G
Use Authentication	NO = Not checked
<b>Client Mode</b>	
Hostname	Air Quality Monitor
IP Configuration	Static IP
IP	192.168.N.NNN
Subnet Mask	255.255.255.0
Gateway	192.168.N.N
SSID	***
Password	***

#### Note

The web interface is enabled, which means calling the WiFi Extension URL 192.168.N.NN in a web browser enabled to see and update the WiFi settings.

In tests, it takes up-to a minute to see the web interface.

## Air Quality Bricklet

The Brick Viewer has a Tab Air Quality showing the bricklet properties and data.

For the Python script, the UID is required, i.e. **Jvj**.

In the Python script, following configuration settings are used:

Status LED Config: 0 = OFF

Temperature Offset : 200, which is 2°C (10=0.1°C)

The offset has been measured against the real room temperature.

#### Action

Check from time-to-time if the offset value still ok against the real room temperature.

The Domoticz log shows the measured data including the temperature. Example Item Data:

```
{"iaqindex": 249, "iaqquality": "Worse", "iaqaccuracy": "Low", "temperature": 21.31, "humidity": 58.87, "airpressure": 1015.27, "status": "OK", "title": "get_bricklet_data"}
```

## Domoticz Configuration

### Custom Sensor

The Air Quality is assigned to a Domoticz Custom Sensor from the Dummy hardware (called VirtualSensors).

For this solution, the Custom Sensor enables to show the Indoor Air Quality Index 0-500 (IAQ Index) and the Air Quality Condition as part of the device name.

GUI > Setup > Hardware > Select Type Dummy > Press Create Virtual Sensors	Create the Virtual Sensor  Name: Wohnzimmer Sensor Type: Custom Sensor Axis Label: VOC  <i>Note</i>
--	---

	The German name reflects measuring the Air Quality in the living room.
--	--

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
354	VirtualSensors	00082354	1	Wohnzimmer Luftqualität	General	Custom Sensor	0 IAQ Index

### Note

In Domoticz, there is a sensor type Air Quality, but that is meant for CO<sub>2</sub> measures (in ppm).

## Hue Device

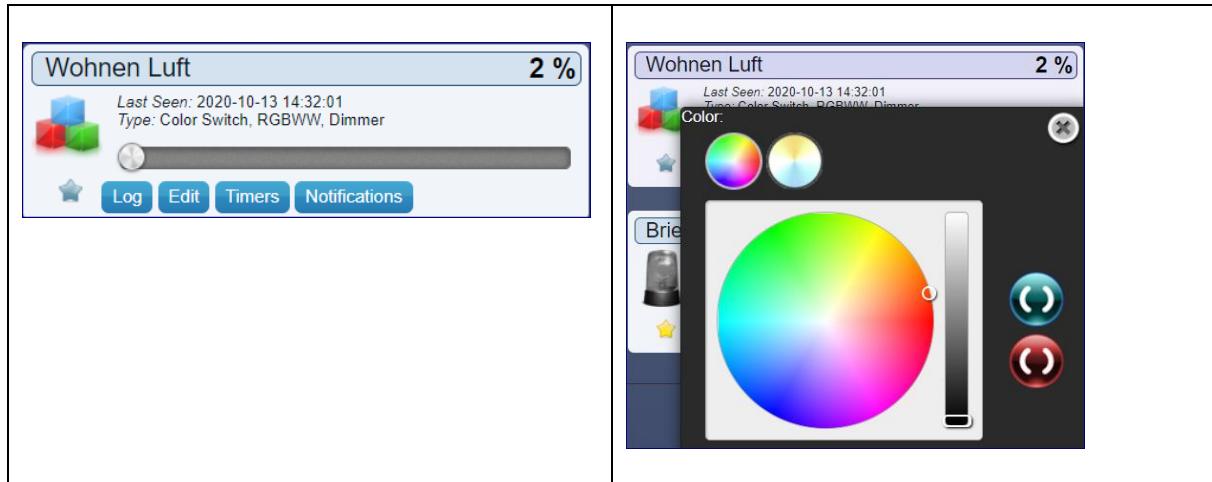
The Indoor Air Quality Index specific color is set for a Philips Hue Bulb via a Domoticz Type Color Switch, SubType RGBWW and Switch Type Dimmer.

Read chapter [Hue Light Add New](#) on how to add a new Hue device.

Device is assigned to the Hardware called Philips Hue Bridge 1

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
355	Philips Hue Bridge 1	00000009	1	Wohnen Luft	Color Switch	RGBWW	Set Color

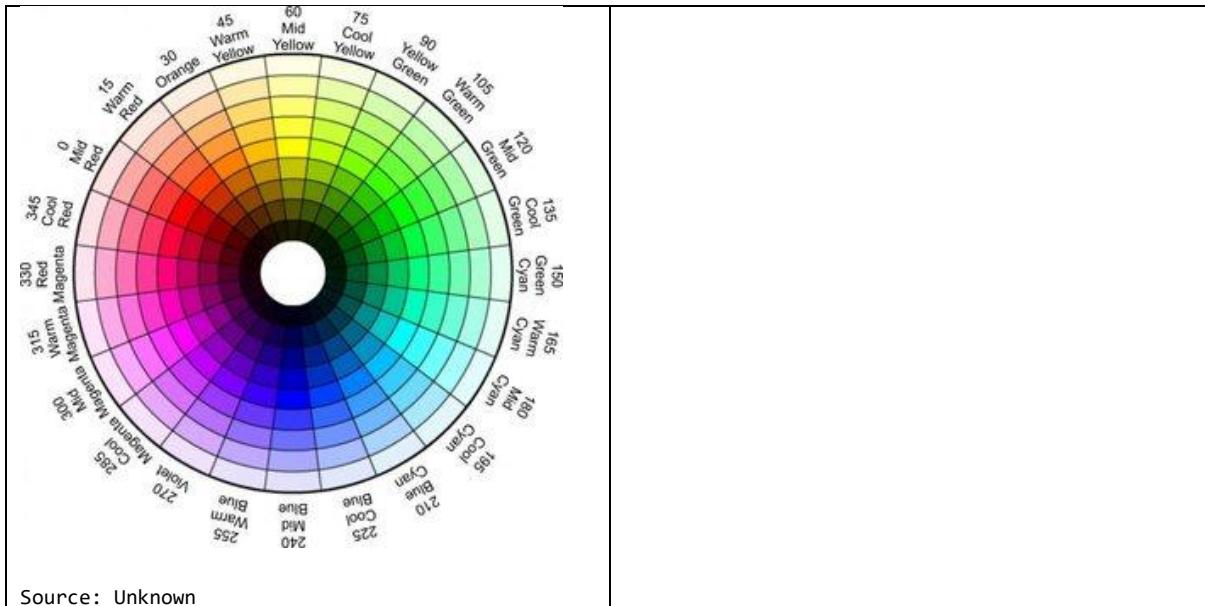
The device widget with brightness 2% and color 15° warm red.



The color is set, via an Automation Event dzVents, according this Hue Color Wheel:

Nr	IAQ Index	Hue Degrees
0	Unknown	240 – mid blue
1	0-50 Good	120 - mid green
2	51-100 Average	090 - yellow green
3	101-150 Little bad	045 - warm yellow
4	151-200 Bad	030 – orange
5	201-300 Worse	015 - warm red
6	301-500 Very bad	345 - cool red

*Note*  
The colors might change depending usage.



## Pseudo Code

### Brief Description

The dzVents Custom Event triggers a Python script to get the Air Quality Bricklet data and send it back to Domoticz. The Custom Event gets the data and updates the Domoticz Custom Sensor.

#### 1. dzVents Custom Event - Trigger Timer

- a. Execute Python script as background process (to not block Domoticz)

#### 2. Python Script

- a. Connect to Tinkerforge Master Brick
- b. Get Tinkerforge Air Quality Bricklet Data
- c. Send Domoticz HTTP API request to the Custom Event with JSON string as parameter holding the bricklet data

#### 3. dzVents Custom Event – Trigger Custom Events

- a. Get Item Data JSON
- b. Parse JSON to Lua Table

*Note:* This step is not required in Domoticz 2020.2 but in BETA 2020.2 (build 12467 or higher) it is required.

- c. Update sensor value and name from table properties
- d. Set the color of the Hue Bulb device – if the device is switched on  
The colors are defined according the Indoor Air Quality Index value.

# Python Script

Source: indoor\_air\_quality.py

Folder: /home/pi/domoticz/scripts/python

## Installation

Change the constants

- TFHOST, TFPORt (default 4223), TFUID, Bricklet Text (to the desired language).
- DOMIP, DOMPORT in case other than the defaults are used.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

indoor_air_quality.py
domoticz-workbook - function indoor_air_quality
Get Air Quality Bricklet data and submit in regular intervals the data to domoticz to update an air
quality device or other devices.
A domoticz custom event (indoor_air_quality.dzvents) handles the incoming data defined as json by
parsing the properties and update the devices.
The data parameter is a json string. Example:
{"iaqindex": 249, "iaqquality": "Worse", "iaqaccuracy": "Low", "temperature": 21.31, "humidity": 58.87,
"airpressure": 1015.27, "status": "OK", "title": "get_bricklet_data"}

CLI Run Example with output from folder /home/pi/domoticz/scripts/python:
$python3 indoor_air_quality.py
INFO:root:Calibration Duration Days [0=4,1=28]:0
INFO:root:Status LED:0
INFO:root:Temperature Offset:2.0
INFO:root:IAQ Index:94
INFO:root:IAQ Quality:Gut
INFO:root:IAQ Index Accuracy:Low
INFO:root:Temperature:18.82 °C
INFO:root:Humidity:59.4 %RH
INFO:root:Air Pressure:1016.08 hPa
INFO:root:OK,get_bricklet_data
INFO:root:OK,Custom Event

In case of an error, i.e. master brick not reachable
$python3 indoor_air_quality.py
ERROR:root:ERR,OS Error: No route to host
INFO:root:OK,Custom Event

Dependencies: indoor_air_quality.dzvents
20201013 rwbl
"""

import json
import requests
import time
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_air_quality import BrickletAirQuality
import logging as log
log.basicConfig(level=log.INFO)

## Tinkerforge parameter Host and Air Quality Bricklet
## Ensure to set the parameter right - test with brick viewer first
TFHOST = "192.168.1.114"
TFPORT = 4223
TFUID = "Jvj"

## Air Quality Bricklet
## Status led config: 0 = OFF
IQA_STATUS_LED = 0
## Air Quality Condition text
IQA_QUALITY_UNKNOWN = "Unbekannt" # Unknown
IQA_QUALITY_GOOD = "Sehr Gut" # Good
IQA_QUALITY_AVERAGE = "Gut" # Average
IQA_QUALITY_LITTLEBAD = "Befriedigend" # Little Bad
IQA_QUALITY_BAD = "Unbefriedigend" # Bad
IQA_QUALITY_WORSE = "Schlecht" # Worse
```

```

IQA_QUALITY_VERYBAD = "Sehr Schlecht"      # Very Bad

## Domoticz IP is local (as running on the Domoticz system) with default port
## Ensure the name of the custom event matches the automation event setting:
## on = { customEvents = { 'tfindoorairquality', }, },
DOMIP = "http://127.0.0.1"
DOMPORT = 8080
DOMCUSTOMEVENT = "indoorairquality"

## Loop counter with the delay in seconds between the loop
## This option can be used to get the data in intervals less than one minute
counter = 0      # count loop
maxcounter = 1   # run n times
delay = 10       # seconds

## Get the Tinkerforge bricklet data
## Returns JSON string (example with data):
## {"iaqindex": 249, "iaqquality": "Worse", "iaqaccuracy": "Low", "temperature": 21.31, "humidity": 58.87, "airpressure": 1015.27, "status": "OK", "title": "get_bricklet_data"}
def get_bricklet_data():
    # Create the JSON object
    data = {}
    # Connect to the master brick and get the data from the Air Quality Bricklet
    try:
        # Create IP connection
        ipcon = IPConnection()

        # Create device object
        aq = BrickletAirQuality(TFUID, ipcon)

        # Connect to brickd
        ipcon.connect(TFHOST, TFPORT)
        # Don't use device before ipcon is connected

        # Get calibration duration
        log.info("Calibration Duration Days [0=4,1=28]: " +
str(aq.get_background_calibration_duration()))

        # Set status LED
        aq.set_status_led_config(IQA_STATUS_LED)
        log.info("Status LED: " + str(aq.get_status_led_config()))

        # Set temperature offset by 2°C (10=0.1°C, measured against real room temperature)
        # Note: Check from time-to-time if still ok
        aq.set_temperature_offset(200)
        log.info("Temperature Offset: " + str(aq.get_temperature_offset() * 0.01))

        # Get all values
        iaq_index, iaq_index_accuracy, temperature, humidity, \
            air_pressure = aq.get_all_values()

        # Assign the values to the JSON object
        data['iaqindex'] = iaq_index
        log.info("IAQ Index: " + str(iaq_index))

        # IAQ index range 0-500: 0-50(good), 51-100(average), 101-150(little bad), 151-200(bad), 201-300(worse), 301-500(very bad).
        iaq_quality = IQA_QUALITY_UNKNOWN
        if iaq_index >= 0 and iaq_index < 51:
            iaq_quality = IQA_QUALITY_GOOD
        elif iaq_index > 50 and iaq_index < 101:
            iaq_quality = IQA_QUALITY_AVERAGE
        elif iaq_index > 100 and iaq_index < 150:
            iaq_quality = IQA_QUALITY_LITTLEBAD
        elif iaq_index > 150 and iaq_index < 201:
            iaq_quality = IQA_QUALITY_BAD
        elif iaq_index > 200 and iaq_index < 301:
            iaq_quality = IQA_QUALITY_WORSE
        elif iaq_index > 300 and iaq_index < 501:
            iaq_quality = IQA_QUALITY_VERYBAD
        data['iaqquality'] = iaq_quality
        log.info("IAQ Quality: " + iaq_quality)

        iaq_accuracy = "Unknown"
        if iaq_index_accuracy == aq.ACURACY_UNRELIABLE:

```

```

        iaq_accuracy = "Unreliable"
    elif iaq_index_accuracy == aq.ACURACY_LOW:
        iaq_accuracy = "Low"
    elif iaq_index_accuracy == aq.ACURACY_MEDIUM:
        iaq_accuracy = "Medium"
    elif iaq_index_accuracy == aq.ACURACY_HIGH:
        iaq_accuracy = "High"
    data['iaqaccuracy'] = iaq_accuracy
    log.info("IAQ Index Accuracy:" + iaq_accuracy)

    data['temperature'] = temperature/100.0
    log.info("Temperature:" + str(temperature/100.0) + " °C")

    data['humidity'] = humidity/100.0
    log.info("Humidity:" + str(humidity/100.0) + " %RH")

    data['airpressure'] = air_pressure/100.0
    log.info("Air Pressure:" + str(air_pressure/100.0) + " hPa")

    # Disconnect
    ipcon.disconnect()

    data['status'] = "OK"
    data['title'] = "get_bricklet_data"
    log.info(data['status'] + "," + data['title'])

    # Handle errors
    except OSError as e:
        data['status'] = "ERR"
        data['title'] = "OS Error: " + str(e.strerror)
        log.error(data['status'] + "," + data['title'])
        time.sleep(1)

    # Return the data
    return json.dumps(data)

## Send the HTTP API request to domoticz to update the devices
## In the custom event the idx of the devices are defined
def send_request():
    # increase the counter
    global counter
    counter = counter + 1
    # get the json data for the custom event request from tinkerforge bricklet
    json_data = get_bricklet_data()
    # create the url to signal the domoticz custom event
    url = "{}:{}/.json.htm?type=command&param=customevent&event={}&data={}".format(DOMIP, DOMPORT,
DOMCUSTOMEVENT, json_data)
    # log.info("Counter:{}, URL:{}".format(counter, url))
    # submit the url
    try:
        r = requests.get(url, timeout=3)
        r.raise_for_status()
        # log.info(r.status_code) # 200
        content_json = json.loads(r.content)
        # {'status': 'OK', 'title': 'Custom Event'}
        log.info(content_json['status'] + "," + content_json['title']) # OK, Custom Event
    except requests.exceptions.HTTPError as errh:
        log.error("Domoticz Update HTTP Error," + errh)
    except requests.exceptions.ConnectionError as errc:
        log.error("Domoticz Update Error Connecting," + errc)
    except requests.exceptions.Timeout as errt:
        log.error("Domoticz Update Timeout Error," + errt)
    except requests.exceptions.RequestException as err:
        log.error("Domoticz Update Something else went wrong," + err)

## Main
if __name__ == '__main__':
    while True:
        send_request()
        if counter == maxcounter:
            quit()
        time.sleep(delay)

```

Check if the script is running, by involving from the CLI:

```
python3 indoor_air_quality.py

INFO:root:Calibration Duration Days [0=4,1=28]:0
INFO:root:Status LED:0
INFO:root:Temperature Offset:2.0
INFO:root:IAQ Index:94
INFO:root:IAQ Quality:Gut
INFO:root:IAQ Index Accuracy:Low
INFO:root:Temperature:18.82 °C
INFO:root:Humidity:59.4 %RH
INFO:root:Air Pressure:1016.08 hPa
INFO:root:OK,get_bricklet_data
INFO:root:OK,Custom Event
```

## Automation Script

Source: indoor\_air\_quality.dzvents

### Installation

Create a new event GUI > Setup > More Options > Events > + > dzVents Minimal.

Clear the editor and copy & paste below script in the editor.

Change

- IDX of the Custom Device and save
- For tests, the timer trigger change to ‘every minute’.

Save and check the Domoticz log.

```
--[[{"name": "indoor_air_quality.dzvents", "trigger": "customEvent", "script": "To update in regular intervals the custom sensor Indoor Air Quality Index (IAQINDEX). Depending IAQ Index value the color of a Hue color lamp is set according Hue color wheel 0-360 degrees. (See function setHueDegrees) The custom event receives data from a Python script running on the Raspberry Pi CLI. The data is a JSON string. Example Item Data: {"iaqindex": 249, "iaqquality": "Worse", "iaqaccuracy": "Low", "temperature": 21.31, "humidity": 58.87, "airpressure": 1015.27, "status": "OK", "title": "get_bricklet_data"} The data is converted to a Lua table and used to update domoticz device(s). For now, only one device is used: Virtual sensor showing the IAQ Index 0 - 500 taken from the JSON key iaqindex.\nDependencies: indoor_air_quality.py\nLocation Python script: /home/pi/domoticz/scripts/python/indoor_air_quality.py\n\nDomoticz Log Example:\n...Status: dzVents: Info: TFINDOORAIRQUALITY: ----- Start internal script: indoor_air_quality:, trigger: "every 2 minutes"\n...(Philips Hue Bridge 1) Color Switch (Wohnen Luftqualität)\n...Status: dzVents: Info: TFINDOORAIRQUALITY: ----- Finished indoor_air_quality\n...Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua\n...Status: dzVents: Info: Handling Domoticz custom event for: "indoorairquality"\n...Status: dzVents: Info: TFINDOORAIRQUALITY: ----- Start internal script: indoor_air_quality: Custom event: "indoorairquality"\n...Status: dzVents: Info: TFINDOORAIRQUALITY: Status:OK\n...Status: dzVents: Info: TFINDOORAIRQUALITY: Index:73 (Gut), A:Low, T:20.95, H:51.81, P:1015.99, Color:90 Bri:1\n...Status: dzVents: Info: TFINDOORAIRQUALITY: ----- Finished indoor_air_quality\n...Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua\n...Status: setcolbrightnessvalue: ID: 163, bri: 1, color: '{m: 3, RGB: 7fff00, CWW: 0000, CT: 0}'\n\nBackground\nThe IAQ index is a measurement for the quality of air.\nTo calculate the IAQ index the Bricklet detects ethane, isoprene (2-methylbuta-1,3-diene), ethanol, acetone, and carbon monoxide (often called VOC, volatile organic components) by adsorption.\nThese gas measurements are combined with the measurements of air pressure, humidity, and temperature to calculate the final IAQ index."}]}
```

The IAQ index has a range of 0-500: 0-50(good),51-100(average),101-150(little bad),151-200(bad),201-300(worse),301-500(very bad).

```

20201013 rwbl
]]-- 

-- Idx of the devices
-- 354, VirtualSensors, Wohnzimmer Luftqualität, General, Custom Sensor, 0, IAQ Index
local IDX_IAQINDEX = 354
-- 355, Philips Hue Bridge 1, Wohnen Luft, Color Switch, RGBWW, Set Color
local IDX_IAQHUE = 355
-- Set the name of the device with the air quality condition (iaqquality)
local NAME_IAQINDEX = "Luft Wohnen (%s)"

-- Timers
local IAQHUE_TIMER = 'every 2 minutes' -- 'every minute'

-- Full path of the python script to get the bricklet data.
-- IMPORTANT: runs as an async background process (using the & sign) to not block the domoticz event
system
local CMD_PYTHON_SCRIPT = "python3 /home/pi/domoticz/scripts/python/inodoor_air_quality.py&"

-- Run external command: python in this case
local function osExecute(cmd)
    local fileHandle      = assert(io.popen(cmd, 'r'))
    local commandOutput   = assert(fileHandle:read('*a'))
    local returnTable     = {fileHandle:close()}
    return commandOutput,returnTable[3]
end

-- Sets the hue degrees 0-360, using the hue color wheel, depending iaqindex
-- IAQ index range 0-500 - see if statements below
-- Used by the dzVents function huedevice.setHue(hue, brightness, isWhite)
-- Returns hue degrees
-- Hint: Read more about colors: https://learnui.design/blog/the-hsb-color-system-practitioners-primer.html
local function setHueDegrees(iaqindex)
    huedegrees = 240
    if iaqindex >= 0 and iaqindex < 51 then huedegrees = 120 end
    if iaqindex > 50 and iaqindex < 101 then huedegrees = 90 end
    if iaqindex > 100 and iaqindex < 150 then huedegrees = 45 end
    if iaqindex > 150 and iaqindex < 201 then huedegrees = 30 end
    if iaqindex > 200 and iaqindex < 301 then huedegrees = 15 end
    if iaqindex > 300 and iaqindex < 501 then huedegrees = 345 end
    return huedegrees
    -- 0 unknown: 240
    -- 1 good: 120 mid green
    -- 2 average: 90 yellow green
    -- 3 little bad: 45 warm yellow
    -- 4 bad: 30 orange
    -- 5 worse: 15 warm red
    -- 6 very bad: 345 cool red
end

return
{
    on =
    {
        timer =
        {
            IAQHUE_TIMER, IAQHUE_TIMER_ON, IAQHUE_TIMER_OFF,
        },
        customEvents =
        {
            'inodoorairquality',
        },
    },
    logging =
    {
        level = domoticz.LOG_INFO,
        -- level = domoticz.LOG_DEBUG,
        marker = 'INDOORAIRQUALITY',
    },
    execute = function(domoticz, item)
        -- Timer starting the python script as a process
        if item.isTimer and item.trigger == IAQHUE_TIMER then
            -- run the external python script
            local output,rt = osExecute(CMD_PYTHON_SCRIPT)
            -- domoticz.log(string.format('OS Execute=%d:%s',rt,output), domoticz.LOG_INFO)
    
```

```

-- handle the result of the python command: 0=OK,>0=ERROR
end

-- Custom event handles the data triggered by the python script http api request
if item.isCustomEvent then
    -- domoticz.log(("Item Data: %s"):format(item.data))
    -- Read item data
    local data = item.data
    -- IMPORTANT: Domoticz 2020.2 no need to convert item data to json
    -- BUT in the 2020.2 BETA it is needed to convert item data to lua table
    -- local data = domoticz.utils.fromJSON(item.data)
    domoticz.log(string.format('Status:%s',data.status), domoticz.LOG_INFO)
    -- domoticz.utils.dumpTable(data)
    -- data is ok
    if data.status == "OK" then
        domoticz.log(("Index:%d (%s), A:%s, T:%.2f, H:%.2f, P:%.2f, Color:%d Bri:%d"):format(
            data.iaqindex, data.iaqquality,
            data.iaqaccuracy,data.temperature,data.humidity,data.airpressure,
            setHueDegrees(data.iaqindex), domoticz.devices(IDX_IAQHUE).level))
        -- Update the devices
        -- Info: Custom sensor air quality index
        domoticz.devices(IDX_IAQINDEX).updateCustomSensor(data.iaqindex)
        -- Notifier: Set hue air quality color if the device is not turned off
        if domoticz.devices(IDX_IAQHUE).state ~= 'Off' then
            domoticz.devices(IDX_IAQHUE).setHue(setHueDegrees(data.iaqindex),
domoticz.devices(IDX_IAQHUE).level, false)
        end
        -- Add another notifier, i.e. email if iaqindex is high
    else
        -- data has error. set the iaqquality to ERROR
        data.iaqquality = "ERROR"
    end

    -- Amend the air quality condition to the device name, i.e. "Luftqualität (Gut)"
    local newname = domoticz.utils.urlEncode(string.format(NAME_IAQINDEX, data.iaqquality))
    -- domoticz.log(string.format('New name:%s',newname), domoticz.LOG_INFO)
    local urlRequest =
string.format("http://127.0.0.1:8080/json.htm?type=setused&idx=%d&used=true&name=%s",
            IDX_IAQINDEX, newname)
    -- Set the new name via url without response
    domoticz.openURL({url = urlRequest}).afterSec(2)
end
end
}

```

&lt;TODO&gt;

**IMPORTANT:** Domoticz 2020.2 no need to convert item data to json  
**BUT in the 2020.2 BETA** it is needed to convert item data to lua table

```
local data = domoticz.utils.fromJSON(item.data)
```

Checkout after update to a newer Domoticz version.

## Domoticz Log

(snippet)

```
...Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
...Status: dzVents: Info: Handling Domoticz custom event for: "indoorairquality"
...Status: dzVents: Info: INDOORAIRQUALITY: ----- Start internal script: indoor_air_quality: Custom
event: "indoorairquality"
...Status: dzVents: Info: INDOORAIRQUALITY: Status:OK
...Status: dzVents: Info: INDOORAIRQUALITY: Index:203 (Schlecht), A:Low, T:21.05, H:56.49, P:1017.09,
Color:15 Bri:2
...Status: dzVents: Info: INDOORAIRQUALITY: ----- Finished indoor_air_quality
```

## Enhancements

These are just some thoughts on enhancing the solution.

### CO<sub>2</sub> measurement

CO<sub>2</sub> measurement using a Tinkerforge [CO<sub>2</sub> Bricklet](#).

The bricklet could be placed on the case front plate (opposite the Air Quality Bricklet placed on the case back plate).

### Alert Sensor

Use an Alert Sensor to indicate IAQ Index.

The Domoticz Alert Sensor has 1+4 levels: (0=grey, 1=green, 2=yellow, 3=orange, 4=red).

The IAQ Index has 6 levels. To use the Alert sensor, suggest combining levels:

IAQ Index Levels	Alert Sensor Levels
0: unknown	0: grey
1: 0-50 = good	1: 0-100 green
2: 51-100 = average	
3: 101-150 = little bad	2: 101-150 yellow
4: 151-200 = bad	3: 151-300 orange
5: 201-300 = worse	
6: 301-500 = very bad	4: 301-500 red

### Notification

Send a notification if Air Quality Index threshold exceeded

### RGB LED

Show Air Quality Index color by a Tinkerforge RGB LED Bricklet instead Hue Bulb.

### Hue Bulb Indicate Periodically

Show Air Quality Index color periodically, i.e. every hour for 1 minute, by a Hue Color Bulb which is normally used as a light.

#### Solution Pseudo Code HTTP API requests

TESTED, see [Change Color every min 2 secs.](#)

#### Timer Event

1. Set Hue color to red via HTTP request
2. Handle HTTP request to set color to green after 2 seconds via HTTP request
3. Do NOT handle the HTTP request
4. After 1 hour start at step 1

#### Solution dzVents

NOT TESTED, have not found a solution on how to use function forNNN and afterNNN with Hue functions to change the color.

#### Timer Event – Every Hour switch Hue on for one minute

1. Save the Hue Bulb Color and Level
2. Get the Air Quality Index Color
3. Set the Hue Bulb Air Quality Index Color & Level
4. Turn the device on for one minute - device.switchOn().forMin(1)
5. Set the Hue Bulb Color & Level
6. Switch the device back on - device.switchOn()

# Indoor Air Quality (Tinkerforge, Plugin)

## Purpose

- To measure the Indoor Air Quality Index, Condition and Accuracy, Air Pressure, Humidity, Temperature, Illuminance.
- To display values in an Indoor Air Quality Station and in Domoticz.
- To enable additional functionality by using scripts (preferred dzVents Lua scripts), i.e. switch LCD backlight, switch room light depending Lux threshold.

## Solution

An **Indoor Air Quality Station** is built out of [Tinkerforge](#) Building Blocks:  
Master Brick with WiFi Extension, Bricklets Air Quality, LCD 20x4, RGB LED, Ambient Light.



### Note

A screenshot of the prototype in action.

**Air Quality Bricklet** measures the IAQ Index (ppm), IAQ Condition, IAQ Accuracy, Air Pressure (mbar), Humidity (%), Temperature (C).

Referencing Tinkerforge [documentation](#): *The IAQ index is a measurement for the quality of air. To calculate the IAQ index the Bricklet detects ethane, isoprene (2-methylbuta-1,3-diene), ethanol, acetone, and carbon monoxide (often called VOC, volatile organic components) by adsorption. These gas measurements are combined with the measurements of air pressure, humidity, and temperature to calculate the final IAQ index.*

- **IAQ Condition Levels** (6) range=condition (color):  
0-50=Good (green), 51-100=Moderate (yellow), 101-150=Unhealthy sensitive groups (orange), 151-200=Unhealthy (red), 201-300=Very Unhealthy (purple), 301-500=Hazardous (maroon).
- **IAQ Index Accuracy Levels** (4): Unreliable, Low, Medium, High.

**Ambient Light Bricklet** measures the Illuminance (lx).

**LCD 20x4 Bricklet** displays the IAQ Index ppm, IAQ Condition, Temperature C, Humidity %, Air Pressure mbar, Illuminance lx.

**RBG LED Bricklet** indicates the IAQ Condition Level Color.

The plugin polls in regular intervals data from the **Indoor Air Quality Station**.

**Domoticz Indoor Air Quality Devices** created - Name (Type,SubType):

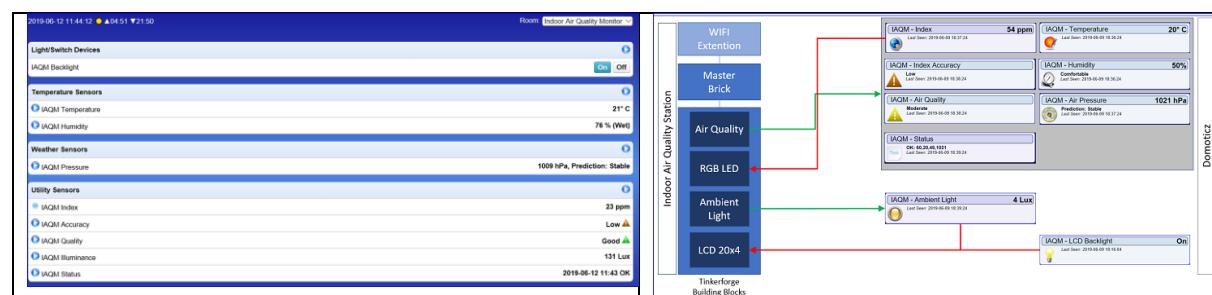
- Index (General, Custom Sensor), Index Accuracy (General, Alert), Air Quality (General, Alert [Text=Level]).
- Temperature (Temp, LaCrosse TX3), Humidity (Humidity, LaCrosse TX3), Air Pressure (General, Barometer), Ambient Light (Lux, Lux).
- LCD Backlight (Light/Switch, Switch), Status (General, Text).

*Note*

Enhanced functionality, like switch LCD backlight, switch lights (depending Lux) are realized with dzVents Lua scripts.

For more information or get the latest version, go [here](#) (GitHub).

See also [Python Plugin Development](#).



*Note*

The dashboard with room Indoor Air Quality Monitor (in mobile mode).

# Info Message

## Purpose

NOT USED – Replaced by the Alert Message.

To display an Info Message on the Dashboard.

The Info Message is populated by various dzVents events, i.e.

- Raspberry Pi monitor (event name: rpi\_monitor)
- Hue lamps living room timed switch (event name: hue\_wz\_timer)
- More see the dedicated functions

## Solution

Add a Virtual Sensor named “Control Message”, Type General, SubType Text.

52	VirtualSensors	00082051	1	Control Message	General	Text	Hue MakeLab Switching OFF 16:55:34
----	----------------	----------	---	-----------------	---------	------	------------------------------------

In the various dzVents events, the device is updated.

Control Message	Hue MakeLab switched OFF 17:18:05
 Text <i>Last Seen: 2018-09-05 17:18:05</i> <i>Type: General, Text</i>	
 Log 	

## Automation Script

Event name: info\_msg

```
local IDX_HUE_MAKELAB = 118;
local IDX_CONTROLMSG = 52

local function isnowtime(domoticz)
    return domoticz.time.rawTime
end

return {
    active = true,
    on = {
        devices = {
            IDX_HUE_MAKELAB
        }
    },
    execute = function(domoticz, device)
        if (device.state == 'On') then
            device.switchOff()
            domoticz.devices(IDX_CONTROLMSG).updateText(device.name .. ' switched OFF ' ..
isnowtime(domoticz))
        end
    end
}
```

# Key Dates (JSON)

## Purpose

To manage & display key dates, ordered by days-to-go descending, in a dedicated roomplan named “Termine”.

The key dates are defined as Domoticz Alert devices (Hardware VirtualSensors) and show the Alert level depending days-to-go till key date: Red=0, Yellow=1, Green > 1.

The devices are sorted by days-to-go starting lowest value (i.e. 0=today etc.).

### Screenshot Domoticz Dashboard (Mobile, simple-gray) with waste calender key dates



### Screenshot devices list (extract)

⚠️	328	VirtualSensors	82328	1	Gelbe-Tonne Pi	General	Alert	11-09-2020 (in 9 Tage)
⚠️	327	VirtualSensors	82327	1	Restabfall Pi	General	Alert	10-09-2020 (in 8 Tage)
⚠️	326	VirtualSensors	82326	1	Papier Pi	General	Alert	18-09-2020 (in 16 Tage)
⚠️	325	VirtualSensors	82325	1	Biomüll Pi	General	Alert	03-09-2020 (Morgen)

#### Note

There is a second solution using CSV input file.

*This has been the first solution and is not maintained and used anymore* – just kept for reference.

## Solution

The key dates are defined in the external file “keydates.json” in JSON format.

Key dates file location example RPi:

```
/home/pi/domoticz/scripts/dzVents/scripts/keydates.json
```

The JSON data holds an array with N key dates.

The JSON key:value pairs are:

Key	Value	Description
name	String “Keydate”	Name of the key date which is set as the name for the device with idx. See next key idx
idx	Number NNN	idx of the key date device (checkout Domoticz GUI > Setup > Devices) Note

		This key is optional as the automation event uses existing Key Date devices to be able to set the sort order. This means existing idx will be renamed depending days-to-go value.
date	Array ["DD-MM-YYYY", "DD-MM-YYYY"]	Array of key dates in format DD-MM-YYYY as string

Example JSON structure for 7 key dates:

```
[{"name": "KeyDate #1", "idx": 325, "dates": ["09-01-2020", "23-01-2020"]}, {"name": "KeyDate #2", "idx": 326, "dates": ["09-02-2020", "23-02-2020"]}, {"name": "KeyDate #3", "idx": 327, "dates": ["09-03-2020", "20-03-2020"]}, {"name": "KeyDate #4", "idx": 328, "dates": ["09-04-2020", "20-04-2020"]}, {"name": "KeyDate #5", "idx": 329, "dates": ["09-05-2020", "20-05-2020"]}, {"name": "KeyDate #6", "idx": 330, "dates": ["09-06-2020", "20-06-2020"]}, {"name": "KeyDate #7", "idx": 331, "dates": ["09-07-2020", "20-07-2020"]}]
```

The key date devices are updated once per day at 0015.  
 Updated are the device name, alert level & text assigned to an index.  
 The dates are maintained manually in the file.

## Domoticz Configuration

The key data devices (initially 7 – this depends on the number of key date entries in the JSON file, named KeydateN) are created via the Bash script “keydates\_create.sh”.

## Devices

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
331	VirtualSensors	82331	1	Keydate7	General	Alert	No Alert!
330	VirtualSensors	82330	1	Keydate6	General	Alert	No Alert!
329	VirtualSensors	82329	1	Keydate5	General	Alert	No Alert!
328	VirtualSensors	82328	1	Keydate4	General	Alert	No Alert!
327	VirtualSensors	82327	1	Keydate3	General	Alert	No Alert!
326	VirtualSensors	82326	1	Keydate2	General	Alert	No Alert!
325	VirtualSensors	82325	1	Keydate1	General	Alert	No Alert!

## Bash Script

Source: keydates\_create.sh

```
#!/bin/bash
# keydates_create.sh
# Create keydates devices from type General, Subtype Alert
# Devicetype: pTypeGeneral, 0xF3 - convert to dec: 243
# Devices subType: sTypeAlert 0x16 - convert to dec: 22
# Taken from https://github.com/domoticz/domoticz/blob/development/hardware/hardwaredtypes.h
# or checkout: https://www.domoticz.com/wiki/Developing_a_Python_plugin
# Ensure save bash file in Unix (LF) format and make executable: sudo chmod +x keydates_create.sh
#
# Example from CLI:
# Run the script from /home/pi/domoticz/scripts/dzVents/scripts with ./keydates_create.sh
# Keydates Devices
# Creating...
# {"idx" : "326","status" : "OK","title" : "CreateSensor"}
# Created: Keydate1
# ... and more
# Prior running this script set the number of devices to be created (adjust var devicecount).
# This depends on the JSON array size.
# 20200903 rwbl
```

```

#
# Define the vars
#
domoticzPort=8080
domoticzIP=127.0.0.1
# Hardware used virtual sensors
hardwareIdx=6
# Device from type General, subtype Alert
devicetype=243
devicesubtype=22
# Initial devicename
# If want to be hidden use escaped dollar sign: \$Keydate
sensorname=Keydate
# Create 7 devices as defined in the JSON array
devicecount=7

#
# Create the alert devices
# Example: curl
"http://$domoticzIP:$domoticzPort/json.htm?type=createdevice&idx=$hardwareIdx&sensorname=Keydate2&devicetype=243&devicesubtype=22"
#
echo "Keydates Devices"
counter=1
while [ $counter -le $devicecount ]
do
  echo "Creating device " $counter " ..."
  curl
"http://$domoticzIP:$domoticzPort/json.htm?type=createdevice&idx=$hardwareIdx&sensorname=$sensorname$counter&devicetype=$devicetype&devicesubtype=$devicesubtype"
  echo Created: Keydate$counter
  echo
  ((counter++))
done
echo "Keydates Devices created."
echo "Check Domoticz GUI > Setup > Devices"

```

## Automation Script

The dzVents Lua script event is time triggered once per day at 00:15. See source for details.  
Event name: keydates\_update.dzvents

```
--[[[

keydates_update.dzvents
Update the key date devices sorted by days-to-go (srting lowest).
The keydates are defined in external file keydates.json.
Location example RPi: '/home/pi/domoticz/scripts/dzVents/scripts/keydates.json'
The JSON holds an array with N keydates.
The key:value pairs are:
name = name of the key date which is set as name for the device with idx
idx= idx of the keydate device type general, alert (virtual sensor) [OPTIONAL]
date = array of dates in format DD-MM-YYYY
Example for 3 dates:
[
{"name": "KeyDate1", "idx": NNN, "dates": ["09-01-2020", "23-12-2020"]},
 {"name": "KeyDate2", "idx": NNN, "dates": ["09-01-2021", "23-12-2021"]},
 {"name": "KeyDate3", "idx": NNN, "dates": ["09-10-2021", "20-12-2022"]}
]
The key dates are updated once per day at 0015.
Th idx is optional if the devices are sorted depending days-to-go value.
DEVICE SORT & UPDATE
The key dates device idx must defined in order, i.e. 325 to 331 for 7 devices.
After parsing the JSON file with the key dates, the devices are sorted descending starting with the
lowest days-to-go value.
The sorted devices are then assigned to the key date's devices idx, i.e. the devices is renamed.
Example: After JSON parse, idx assigned based on days to go. Device 331 is renamed to Biomüll Pi
etc.
331:0=Biomüll Pi#03-09-2020 (Heute)
330:7=Restabfall Pi#10-09-2020 (in 7 Tage)
329:8=Gelbe-Tonne Pi#11-09-2020 (in 8 Tage)
328:9=Schadstoff Pi#12-09-2020 (in 9 Tage)
327:15=Papier Pi#18-09-2020 (in 15 Tage)
...
]]]
```

```

Dependencies: global_data
20200903 rwbl
]]--


-- Logging marker
local LM_KEYDATES_UPDATE = 'Keydates Update'
-- Path keydates file json format
local KEYDATES_FILE = '/home/pi/domoticz/scripts/dzVents/scripts/keydates.json'
-- threshold in days difference to notify
local TH_KEYDATE = 1
-- Msg no keydates
local MSG_NOKEYDATE = 'Kein neuer Termin'
-- Idx used sequentially
local IDX_KEYDATES_START = 331 -- down to 325 for 7 devices as defined by DEF_KEY_DATES_COUNT
local DEF_KEYDATES_COUNT = 7
-- Table for the device entries which will be sorted by key daystogo
local deviceTable = {}

-- Helper Functions
local function readFile(f)
    local file = io.open (f, 'r') -- open file in read mode
    local data = file:read('*a') -- read all
    file:close()
    return data
end

-- Set a new name for the alert device
local function renameDevice(domoticz, idx, name)
    local newname = domoticz.utils.urlEncode(name)
    local urlRequest =
        string.format("http://127.0.0.1:8080/json.htm?type=setused&idx=%d&used=true&name=%s", idx,
    newname)
    domoticz.openURL({url = urlRequest}).afterSec(0.1)
end

-- Set a key date from the dates array containing dates in format DD-MM-YYYY
local function setKeyDate(domoticz, keydata)
    -- define the device record to be added to the table
    -- flag to check if a keydate has been found
    local datefound = 0
    -- Loop over the dates and build the deviceRecord to be added to the table
    local deviceRecord = {}
    deviceRecord.name=keydata.name
    for index, keydate in ipairs(keydata.dates) do
        -- split the keydate into an array with 3 entries day, month, year (D,M,Y)
        local keydatesplit = domoticz.helpers.split(keydate, "-")
        -- calculate the daysdiff between now and the keydate and assign to daystogo
        local datediff = domoticz.helpers.datediffnow(keydatesplit[1], keydatesplit[2],
keydatesplit[3])
        -- check if daysdiff => 0 and the date is not found
        if datediff >= 0 and datefound == 0 then
            datefound = 1
            deviceRecord.daystogo = datediff
            -- define the alert level & message
            if datediff == 0 then
                deviceRecord.message = string.format('%s (%s)', keydate, 'Heute')
                deviceRecord.alertlevel = domoticz.ALERTLEVEL_RED
            end
            if datediff == 1 then
                deviceRecord.message = string.format('%s (%s)', keydate, 'Morgen')
                deviceRecord.alertlevel = domoticz.ALERTLEVEL_YELLOW
            end
            if datediff > 1 then
                deviceRecord.message = string.format('%s (in %d Tage)', keydate, datediff)
                deviceRecord.alertlevel = domoticz.ALERTLEVEL_GREEN
            end
        end
    end -- for
    -- check if a keydate has been found, if not then set no date found message
    if datefound == 0 then
        deviceRecord.daystogo = 99999
        deviceRecord.message = MSG_NOKEYDATE
        deviceRecord.alertlevel = domoticz.ALERTLEVEL_GRAY
    end
    table.insert(deviceTable, deviceRecord)

```

```

end

return {
  on = {
    timer = {
      'at 00:15'
      -- 'every minute' -- for tests
    },
  },
  logging = {
    level = domoticz.LOG_INFO,
    marker = LM_KEYDATES_UPDATE,
  },
}

execute = function(domoticz, item)
  -- Read and convert the JSON file to a table
  local keydatesJSON = domoticz.utils.fromJSON(readFile(KEYDATES_FILE))
  -- domoticz.log(string.format('Key Date Entries: %d', #keydatesJSON), domoticz.LOG_INFO)

  -- Loop over the keydates and build the table array deviceTable
  for key, keydata in ipairs(keydatesJSON) do
    setKeyDate(domoticz, keydata)
  end

  -- Sort the deviceTable ascending by daystogo
  table.sort(deviceTable, function(a,b) return a.daystogo < b.daystogo end)

  -- Loop over te deviceTable and update the devices starting with IDX_KEYDATES_START
  local idx = 0
  for i, device in ipairs(deviceTable) do
    idx = IDX_KEYDATES_START - i + 1
    domoticz.log(idx .. ':' .. device.daystogo .. '=' .. device.name .. '#' .. device.message)
    -- update the device alertlevel & text with the new date message
    domoticz.devices(idx).updateAlertSensor(device.alertlevel, device.message)
    -- update the device name
    renameDevice(domoticz, idx, device.name)
  end
end
}

```

## Roomplan

The key date devices are added to the roomplan “Termine” with IDX=4.

Idx	Name	Order
4	Termine	

Showing 1 to 1 of 1 entries (filtered from 7 total entries) 1 row selected

First | Previous | 1 | Next | Last

Devices

Add Clear

Show 25 entries

Idx	Name	Order
331	Biomüll Pi	
330	Restabfall Pi	
329	Gelbe-Tonne Pi	
328	Schadstoff Pi	
327	Papier Pi	

The room plan is used for the Domoticz default dashboard.

The order of the idx is important and must be sequential.

The idx are used to update the device name to get an order by days-to-go starting with the lowest value (see screenshot beginning of the chapter). This is handled by the automation event.

### Background

The devices are mapped to a plan in the Domoticz database table DeviceToPlansMap.

sqlite> select * from DeviceToPlansMap where PlanID=4;						
ID	DeviceRowID	DevSceneType	PlanID	Order	XOffset	YOffset
165	331	0	4	169	0	0
166	330	0	4	170	0	0
167	329	0	4	171	0	0
168	328	0	4	172	0	0

In the event create a table with sorted key dates which is then used to change the name of the key date devices.

# Key Dates (CSV)

## Important

This initial solution is replaced by the function Key Dates (JSON) and not maintained anymore.

## Purpose

To manage & display key dates in a dedicated roomplan “Termine”.

Device	Date
Biotonne Pi	06-02-2020
Gelber Sack Pi	31-01-2020
Papier Pi	07-02-2020
Restmuell Pi	30-01-2020
Schadstoff Pi	15-02-2020

## Brief Description

Various key dates are maintained in a comma-separated-values (CSV) text file located in the dzVents script folder. Each line of the CSV file is related to a Domoticz device (VirtualSensor, Type Text). Each line contains the idx and the various dates. Once per day, a dzVents Lua Automation scripts reads the CSV file line-by-line and updates the text of the device if the days from now till next date is greater or equal 0 (datediff). If the datediff is 1 (set by a threshold), a notification is sent to alert.

The background for this function was to manage waste calendar dates for locations. Therefore, the main key dates are waste calendar dates (“Abfall Termine”) for a city (location) and waste type. The waste types are (in German): Rest, Bio, Gelb, Papier, Schad. The dates are provided by the respective city waste management (“Abfallwirtschaft”) for the cities Pinneberg and Hohenfelde.

The roomplan “Termine” is defined (Domoticz GUI Setup > More Options > Plans > Roomplan) holding all devices as defined in the CSV file.

## Domoticz Configuration

The devices are created as Virtual Sensors Type General, SubType Text using the Hardware Type Dummy.

Idx	Name	Enabled	Type
6	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) <a href="#">Create Virtual Sensors</a>

98	VirtualSensors	00082098	1	Biotonne Pi	General	Text	06-09-2018
99	VirtualSensors	00082099	1	Papier Pi	General	Text	21-09-2018
100	VirtualSensors	00082100	1	Restmüll Pi	General	Text	30-08-2018
101	VirtualSensors	00082101	1	Gelber Sack Pi	General	Text	30-08-2018
102	VirtualSensors	00082102	1	Schadstoff Pi	General	Text	08-09-2018
103	VirtualSensors	00082103	1	Biotonne Ho	General	Text	27-08-2018
104	VirtualSensors	00082104	1	Papier Ho	General	Text	17-09-2018
105	VirtualSensors	00082105	1	Restmüll Ho	General	Text	27-08-2018
106	VirtualSensors	00082106	1	Gelber Sack Ho	General	Text	28-08-2018

## Concept

A dzVents Lua Automation script event, “keydates\_update” executed dzVents time trigger reads a CSV file, which contains per line the Device Idx, Interval, waste dates.

The device is updated based upon the difference in days between now and the waste date. The script also notifies via email (recipients defined in the Domoticz Settings Email and Notification) if the waste target date is 1 days from actual date.

## CSV Input File

The CSV file is read line by line, using per line, the Device Index idx (field 1) to update a date (fields 2 and more depending available dates).

CSV Line Syntax: idx, interval, date1, dateN

idx	NN 98, 99...
date	Format Day-Month-Year dd-mm-YYYY 09-08-2018

### Note

If a device is not used, ensure to take out the line in the keydates.csv file

OR

place -1, prior idx.

If the idx is used, remove the -1, entry. Example:

-1;999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

## Example CSV file line entry

The example shows for

- Idx 98 (“Biotonne Pi”): Number of dates for the remaining year 2018 and full year 2019 (only first few entries).
- Idx 999: Line starts with -1, means this device is not used.

98,13-12-2018,28-12-2018,10-01-2019,24-01-2019,07-02-2019 -1,999,32,23-03-2020,22-06-2020,21-09-2020,16-12-2020
--

## CSV File name & location

The CSV file keydates.csv, located in folder /home/pi/domoticz/scripts/dzVents/scripts.

The file is used by a dzVents Lua script *keydates\_update* created using the Domoticz Script editor.

## Automation Script

The dzVents Lua script event is time triggered once per day at 00:30. See source for details.  
Event name: *keydates\_update*

```
--[[[

    keydates_update.lua
    Read a csv file with keydates per device.
    The keydates file "keydates.csv" is in the dzVents scripts folder.
    (keydatesfile = '/home/pi/domoticz/scripts/dzVents/scripts/keydates.csv')
    Line entry csv file: idx,date1,dateN.
    Example:
    999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

    Action:
    Update the device text with the date if the first datediff in days >= 0

    Note:
    If a device is not used, ensure to take out the line in the keydates.csv file
    OR place -1, prior idx. If the idx is used, remove the -1, entry. Example:
    -1;999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

    Project: domoticz-homeautomation-workbook
    Interpreter: dzVents
    Author: Robert W.B. Linn
    Version: 20200127
]]--


-- threshold in days to notify
-- if the difference of the current day to a keydates day equals threshold
-- then notify
local TH_KEYDATES = 1

-- Helpers
function string:split( inSplitPattern, outResults )
    if not outResults then
        outResults = { }
    end
    local theStart = 1
    local theSplitStart, theSplitEnd = string.find( self, inSplitPattern, theStart )
    while theSplitStart do
        table.insert( outResults, string.sub( self, theStart, theSplitStart-1 ) )
        theStart = theSplitEnd + 1
        theSplitStart, theSplitEnd = string.find( self, inSplitPattern, theStart )
    end
    table.insert( outResults, string.sub( self, theStart ) )
    return outResults
end

-- the keydatescalendar file is updated once per year
local keydatesfile = '/home/pi/domoticz/scripts/dzVents/scripts/keydates.csv'
local sep = ","

-- read the csv file
function readandupdate(domoticz)
    -- read the file line by line.
    -- each line contains an array with device[1], dates[2...]
    for line in io.lines(keydatesfile) do
        -- split the line by comma , (sep) and assign to a table
        local keydatestable = line:split(sep)
        local idx = tonumber(keydatestable[1])
        -- flag to check if a keydate has been found
        local datefound = 0
        -- the idx must be > 0 else not used
        if idx > 0 then
            -- check the dates, start at 2 because 1=idx
            for i = 2, #keydatestable do
                -- get the keydatesdate array entry
                local keydate = keydatestable[i]
                -- split the keydate into an array with 3 entries day, month, year (D,M,Y)
            end
        end
    end
end
]]--]
```

```

local keydatesplit = keydate:split("-")
-- calculate the daysdiff between now and the keydate
local daysdiff = domoticz.helpers.datediffnow(keydatesplit[1], keydatesplit[2],
keydatesplit[3])
-- get the month which is array entry index 2
local month = tonumber(keydatesplit[2])
-- check if daysdiff >= 0 and the date is not found
if daysdiff >= 0 and datefound == 0 then
    datefound = 1
    -- update the device text with the new date
    domoticz.devices(idx).updateText(keydate)
    -- check number of days left compared to the threshold
    -- i.e. if 1 day left, then notify using device name & text.
    -- the device text contains the date
    -- NOT USED = handled by the dzVents script keydates_calendar_notifier
    if daysdiff == TH_KEYDATE then
        -- print ('Device ' .. idx .. ' notifying...')
        message = '[ACHTUNG] ' .. domoticz.devices(idx).name .. ' ' ..
domoticz.devices(idx).text
        -- update alert message
        -- msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
        domoticz.log(message, domoticz.LOG_INFO)
        domoticz.notify(message)
    end
end

-- check if for the last entry daysdiff less 0, then no new date available
-- this is used f.e. for hohenfelde as last keydate is sep or oct
if i == #keydatestable and daysdiff < 0 then
    datefound = 1
    print (idx .. '=' .. keydate .. ', days=' .. daysdiff .. ', KEIN TERMIN')
    -- update the device text with the info that there is no date (yet) available
    domoticz.devices(idx).updateText('Kein neuer Termin')
end

end -- for

-- check if a keydate has been found, if not then set Kein termin message
if datefound == 0 then
    print (idx .. '=' .. 'KEIN TERMIN')
    -- update the device text with the info that there is no date (yet) available
    domoticz.devices(idx).updateText('Kein neuer Termin')
end

-- JUST SOME LEARNERS
-- print the content
-- for i = 1, #keydatestable do
--   print( keydatestable[i] )
-- end
-- print device idx and length of the csv string
-- print (keydatestable[1] .. "Length:" .. #keydatestable)
end -- if idx > 0
end
end

return {
on = {
    timer = {
        -- for tests use every minute
        -- 'every minute'
        'at 00:30'
    }
},
execute = function(domoticz)
    readandupdate(domoticz)
end
}
}

```

## Python CSV File Generation Script

A Python 2 script is used to create the keydates.csv file from an iCal file.

## Script

```

# -*- coding: cp1252 -*-
## File: wastecalictstocsv.py (Python 2)
## Purpose: Create CSV file with waste dates from ICS iCal file Pinneberg and empty entries Hohenfelde

## Notes:
## The generated csv file wastecal.csv must be placed in /home/pi/domoticz/scripts/dzVents/scripts
## Domoticz Device Idx and Interval
## Biotonne Pi=98,16
## Papier Pi=99,32
## Restmuell Pi=100,16
## Gelber Sack Pi=101,16
## Schadstoff Pi=102,32
## Biotonne Ho=103,16
## Papier Ho =104,32
## Restmuell Ho=105,16
## Gelber Sack Ho=106,16
## The generated file must be merged with the keydates.csv file located on the Domoticz system:
## /home/pi/domoticz/scripts/dzVents/scripts
import csv, io, sys
import datetime
from datetime import date

wastecalendarics = 'wastecal-pi-2019.ics'

# Define the wastecalender csv file located in the same folder as the python script.
# CSV Format: wastetype, date1, date2 ...
# Date format dd-mm-YYYY, i.e. 09-08-2018
wastecalendarcsv = 'wastecal-2019.csv'

# Check the python version to set the file open arguments
if sys.version_info[0] == 2: # Not named on 2.6
    icsaccess = 'rb'
    icskargs = {}
    csvaccess = 'wb'
    csvkargs = {}
else:
    icsaccess = 'rt'
    icskargs = {'newline':''}
    csvaccess = 'a'
    csvkargs = {}
    #csvkargs = {'newline':''}

# Define the ics tokens
TOKEN_EVENT_BEGIN = "BEGIN:VEVENT"
TOKEN_EVENT_END = "END:VEVENT"
TOKEN_EVENT_DESCRIPTION = "DESCRIPTION:"
TOKEN_EVENT_START = "DTSTART:"
TOKEN_REST = "Restabfall 2"
TOKEN_BIO = "Bio"
TOKEN_SCHAD = "Schad"
TOKEN_GELB = "Gelb"
TOKEN_PAPIER = "Papier"

# Define the lists holding the dates per waste types. The lists are used to create the csv file
pirestlist = []
pibiolist = []
pischadlist = []
pigelblist = []
pipapierlist = []
horestlist = []
hobiolists = []
hogelblist = []
hopapierlist = []

# Define helpers
eventfound = 0
eventdescription = ""
eventstart = 0
eventyear = 0
eventmonth = 0
eventday = 0

```

```

def csv2string(data):
    si = io.BytesIO()
    cw = csv.writer(si)
    cw.writerow(data)
    return si.getvalue().strip('\r\n')

# Get the actual year month day
s = datetime.date.today().strftime("%d-%m-%Y")
# Get the date now in format i.e. 09-08-2018
datenow = datetime.datetime.strptime(s, "%d-%m-%Y")

# Set the waste type as the first entry in the waste lists
## Biotonne Pi=98,16
## Papier Pi=99,32
## Restmuell Pi=100,16
## Gelber Sack Pi=101,16
## Schadstoff Pi=102,32
## Biotonne Ho=103,16
## Papier Ho =104,32
## Restmuell Ho=105,16
## Gelber Sack Ho=106,16

pibiolist.append('98,16')
pipapierlist.append('99,32')
pirestlist.append('100,16')
pigelblist.append('101,16')
pischadlist.append('102,32')
hobiolist.append('103,16')
hopapierlist.append('104,32')
horestlist.append('105,16')
hogelblist.append('106,16')

# Open the wastecalendar ics file in read access
with open(wastecalendarics, icsaccess, **icskwargs) as icsfile:
    # lines = icsfile.readlines()
    # Loop thru the file line by line
    for line in icsfile:
        # Strip the CRLF from the line
        linet = line.strip()
        # Print the line for tests
        # print linet
        # Check the entries

        # If token event end is found and an event has been built then add a waste entry date to the
        # list.
        if linet.startswith(TOKEN_EVENT_END):
            if eventfound == 1 and len(eventdescription)> 0 and eventstart > 0:
                wastedate = "%s-%s-%s" % (eventday,eventmonth,eventyear)
                # print "Adding the line ", eventdescription, eventstart, eventyear, eventmonth,
                eventday

                # Workout the waste types and add the date
                if eventdescription.startswith(TOKEN_REST):
                    pirestlist.append(wastedate)
                    # print "Adding ", eventdescription, "-", wastedate

                if eventdescription.startswith(TOKEN_BIO):
                    pibiolist.append(wastedate)

                if eventdescription.startswith(TOKEN_PAPIER):
                    pipapierlist.append(wastedate)

                if eventdescription.startswith(TOKEN_GELB):
                    pigelblist.append(wastedate)

                if eventdescription.startswith(TOKEN_SCHAD):
                    pischadlist.append(wastedate)

            # Reset the globals which are used for the next event
            eventdescription = ""
            eventstart = 0
            eventyear = 0
            eventmonth = 0
            eventday = 0
            eventfound = 0

```

```
# Set the flag if an new event is found
if linet.startswith(TOKEN_EVENT_BEGIN):
    eventfound = 1

# Event was found and now the description
if eventfound and linet.startswith(TOKEN_EVENT_DESCRIPTION):
    eventdescription = linet.replace(TOKEN_EVENT_DESCRIPTION, "")

# Event was found and now the start date
# Extract from the date, the year, month, day
if eventfound and linet.startswith(TOKEN_EVENT_START):
    # Example = DTSTART:20180105T053000Z
    eventstartstr = linet.replace(TOKEN_EVENT_START, "")
# Get the date in format 20180105
index = eventstartstr.index('T')
eventstartstr = eventstartstr[0:index]
eventstart = int(eventstartstr)
# Get the year, month, day
eventyear = eventstartstr[0:4]
eventmonth = eventstartstr[4:6]
eventday = eventstartstr[6:8]
# print eventyear, eventmonth, eventday

# Write the lists to the CSV file. Each row contains a waste type with dates.
# Sample: 98,16,12-01-2018,09-02-2018, ...
with open(wastecalendarcsv, csvaccess, **csvkwargs) as csvfile:
    # Pinneberg
    csvfile.write(",".join(pipapierlist))
    csvfile.write("\n")
    csvfile.write(",".join(pirestlist))
    csvfile.write("\n")
    csvfile.write(",".join(pibiolist))
    csvfile.write("\n")
    csvfile.write(",".join(pigelblist))
    csvfile.write("\n")
    csvfile.write(",".join(pischadlist))
    csvfile.write("\n")
    # Hohenfelde - the dates needs to be added manually
    csvfile.write(",".join(hopapierlist))
    csvfile.write("\n")
    csvfile.write(",".join(horestlist))
    csvfile.write("\n")
    csvfile.write(",".join(hobiolist))
    csvfile.write("\n")
    csvfile.write(",".join(hogelblist))
    csvfile.write("\n")
    # Close with message
    csvfile.close()
    print "Created CSV file", wastecalendarcsv

# Exit the script
icsfile.close()
sys.exit()
```

# Music Player (Kodi)

## Purpose

- To Remote Control, via Domoticz, music of a Kodi Media Player running on a Raspberry Pi (or other hardware) with LibreELEC.
- To explore setting up Kodi, communicating with Kodi using JSON-RPC API requests, control & monitor from various systems (Remote Browser, Domoticz and other).
- To use the Kodi Media Player, as-is without developing additional add-ons or change the Kodi environment - strictly use JSON-RPC API requests.
- To use Domoticz Virtual Sensors and a single Automation Event dzVents Lua script.
- Although Kodi is a media player, the focus is on playing music only.

## Solution

Prior creating the Music Player, explored how to communicate with Kodi using JSON-RPC API requests. Recommend reading first the chapter [Kodi Music Player](#).

The description next uses several screenshots as easier to explain. If hard to read, zoom in the page.

The screenshot below outlines the functionality with the various Domoticz Devices grouped in a Roomplan “Kodi Music Player”.

**Light/Switch Devices**

- Player Playback: Play, Pause, Prev, Next
- Player Volume: On, Off
- Player System: Stop, Shutdown
- Player Remote Control: On, Off
- Player Selection: Player A, Player B

**Utility Sensors**

- Player Playing: Artist - Title
- Player Progress: 67.2% / 00:03:14 / 00:04:49

**Domoticz Dashboard Mobile**

Room plan with all devices used by the Music Player.

ACTION: Switch Devices to Control the Player.

Play or pause current music title. Select next or previous title.

Turn the volume On or Off.

Set the volume between 0 – 100%.

Stop the player or shutdown. To start, a push-button connected to the Raspberry Pi GPIO pin #3 is used. Also possible to shutdown the Raspberry Pi.

Use Domoticz to Remote Control. When connected to a player, turn On followed by Play.

Select a player. The player is defined in a user variable which is updated by the switch selection level action. More players can be added by enhancing the switch selector levels.

INFO: Text Devices updated every 10 seconds by Automation Script dzVents Lua.

Current song playing or event status message depending switch selected.

Progress: Percentage & time played; Total title play time.

The remote control is On, playback mode Play and the text devices Playing & Progress are updated every 10 seconds.

## Installation

System	Action
Kodi	Setup a Kodi Player or use an existing – required is the IP address of the player. More than one player can be used.
Domoticz	Create the devices to remote control the player using the hardware “Dummy” (name, type, subtype, switch type): <ul style="list-style-type: none"> <li>• Player Playing, General, Text</li> <li>• Player Progress, General, Text</li> <li>• Player Playback, Light/Switch, Selector Switch</li> <li>• Player Volume, Light/Switch, Switch, Dimmer</li> <li>• Player Remote Control, Light/Switch, Switch</li> <li>• Player Selection, Light/Switch, Selector Switch</li> <li>• Player System , Light/Switch, Selector Switch</li> </ul>
Domoticz	Create two user variables <ul style="list-style-type: none"> <li>• Name: URL_PLAYER, Type: string, Value: http://player-ip:8080/jsonrpc</li> <li>• Name: URL_DOMOTICZ, Type: string, Value: http://domoticz-ip:8080/json.htm</li> </ul>
Domoticz	Create an event dzVents Lua Music_Player. <ul style="list-style-type: none"> <li>• Copy the music_player.dzVents.</li> <li>• Set the idx of the devices.</li> <li>• Set event to On and save the event</li> </ul>
Kodi	Start the Kodi Player via browser (Chorus) or app like Kore and play album etc.
Domoticz	Select the player > select On > select Play. After 10 seconds the information playing, and progress are updated.
Domoticz	Check the log file for errors.
Domoticz	Create a roomplan and /or floorplan (with the roomplan)

## Overview Devices & User Variables

### Devices

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	123	VirtualDevices	00082123	1	Player Progress	General	Text	1.3% / 00:00:03 / 00:03
	117	VirtualDevices	00082117	1	Player Playing	General	Text	Artist - Title
	118	VirtualDevices	000140C6	1	Player Playback	Light/Switch	Selector Switch	Set Level: 10 %
	125	VirtualDevices	000140CD	1	Player System	Light/Switch	Switch	Off
	126	VirtualDevices	000140CE	1	Player Remote Control	Light/Switch	Switch	On
	127	VirtualDevices	000140CF	1	Player Selection	Light/Switch	Selector Switch	Set Level: 10 %
	124	VirtualDevices	000140CC	1	Player Volume	Light/Switch	Switch	Set Level: 49 %

### User Variables

Idx	Variable name	Variable type	Current value
4	URL_DOMOTICZ	String	http://192.168.1.179:8080/json.htm
3	URL_PLAYER	String	http://192.168.1.62:8080/jsonrpc

### Device Player Selection (idx=127)

#### Selector Levels with action to set the user variable URL\_PLAYER

Selector Levels:

Level	Level name
0	Off
10	Player A
20	Player B

Level name:  Add

Selector actions:

Level	Action
0	10 http://192.168.1.179:8080/json.htm? type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.62:8080/jsonrpc 20 http://192.168.1.179:8080/json.htm? type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.58:8080/jsonrpc

Domoticz is running on a Raspberry Pi with IP address 192.168.1.179.  
The music player is running on a Raspberry Pi with IP address 192.168.1.62.  
When the sector level changes, the user variable (idx=3, name=URL\_PLAYER) is updated.  
The Domoticz URL is used to clear logs from the devices to avoid spoiling the database with not needed entries.

Idx: 118  
Name: Player Playback  
Switch Type: Selector

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
Protected:   
Selector Style:  Button set  Select menu  
Hide Off level:

Selector Levels:

Level	Level name
0	OFF
10	Play
20	Pause
30	Prev
40	Next

Idx: 124  
Name: Player Volume  
Switch Type: Dimmer

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
On Action:  
Off Action:  
Protected:

Description:

Idx: 127  
Name: Player Selection  
Switch Type: Selector

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
Protected:   
Selector Style:  Button set  Select menu  
Hide Off level:

Selector Levels:

Level	Level name
0	Off
10	Player A
20	Player B

Level name:  Add

Selector actions:

Level	Action
0	10 http://192.168.1.179:8080/json.htm? type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.62:8080/jsonrpc 20 http://192.168.1.179:8080/json.htm? type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.58:8080/jsonrpc

Idx: 125  
Name: Player System  
Switch Type: Selector

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
Protected:   
Selector Style:  Button set  Select menu  
Hide Off level:

Selector Levels:

Level	Level name
0	OFF
10	Stop
20	Shutdown

Idx: 126  
Name: Player Remote Control  
Switch Type: On/Off

Switch Icon:

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
On Action:  
Off Action:  
Protected:

Description:

## Roomplan and Floorplan

Roomplan “Kodi Music Player”	Floorplan “Kodi Music Player”																																		
<p><b>Room plans</b></p> <p>Show 25 entries Search: kodi</p> <table border="1"> <thead> <tr> <th>Idx</th> <th>Name</th> <th>Order</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>Kodi Music Player</td> <td></td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (filtered from 11 total entries) 1 row selected</p> <p>First   Previous   1   Next   Last</p> <p><b>Devices</b></p> <p>Add Clear</p> <p>Show 25 entries Search:</p> <table border="1"> <thead> <tr> <th>Idx</th> <th>Name</th> <th>Order</th> </tr> </thead> <tbody> <tr> <td>117</td> <td>Player Playing</td> <td></td> </tr> <tr> <td>118</td> <td>Player Playback</td> <td></td> </tr> <tr> <td>123</td> <td>Player Progress</td> <td></td> </tr> <tr> <td>124</td> <td>Player Volume</td> <td></td> </tr> <tr> <td>125</td> <td>Player System</td> <td></td> </tr> <tr> <td>126</td> <td>Player Remote Control</td> <td></td> </tr> <tr> <td>127</td> <td>Player Selection</td> <td></td> </tr> </tbody> </table> <p>Showing 1 to 7 of 7 entries</p> <p>First   Previous   1   Next   Last</p>	Idx	Name	Order	11	Kodi Music Player		Idx	Name	Order	117	Player Playing		118	Player Playback		123	Player Progress		124	Player Volume		125	Player System		126	Player Remote Control		127	Player Selection		<p><b>Floorplan “Kodi Music Player”</b></p> <p><b>Floorplan</b></p> <p>Show 10 entries</p> <table border="1"> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> <tr> <td> Music Player</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries (filtered from 1 total entries)</p> <p>Add   Edit   Delete</p> <p><b>Room Plans</b> (Select Floorplan first to Edit...)</p> <p>Show 10 entries</p> <table border="1"> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> <tr> <td>Kodi Music Player</td> </tr> </tbody> </table> <p>Showing 1 to 1 of 1 entries</p>	Name	Music Player	Name	Kodi Music Player
Idx	Name	Order																																	
11	Kodi Music Player																																		
Idx	Name	Order																																	
117	Player Playing																																		
118	Player Playback																																		
123	Player Progress																																		
124	Player Volume																																		
125	Player System																																		
126	Player Remote Control																																		
127	Player Selection																																		
Name																																			
Music Player																																			
Name																																			
Kodi Music Player																																			

## Remote Control Functions

The next screenshots outline the various Remote Control Functions.

<b>Player A Remote Control On</b>	<b>Player A Playback Play</b>	<b>Player A Playback Pause</b>	<b>Remote Control Off</b>
<p>2020-05-24 13:28:30 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Remote Control turned On, press Play to listen</p> <p>Player Progress: 0%</p>	<p>2020-05-24 13:31:40 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Artist - Title</p> <p>Player Progress: 88.5% / 00:09:27 / 00:10:40</p>	<p>2020-05-24 13:23:44 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Artist - Title</p> <p>Player Progress: Player paused - 29.4% / 00:03:08 / 00:10:40</p>	<p>2020-05-24 13:25:38 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Remote Control turned Off</p> <p>Player Progress: 0%</p>
<p>The remote control is On and asking to press Play to start listening. If the player is not playing, the device Playing states "Not playing".</p> <p>If the remote control is not On, no other switch can be used.</p>	<p>The remote control is On and Play is selected. Devices Playing Artist &amp; Title and the Progress are updated every 10 seconds.</p>	<p>The remote control is On and Playback is set to Pause. Device Playing Progress states Pause with last progress information.</p>	<p>The remote control is Off. Device Playing is updated and Progress set to 0%. The Playback selection is Off (not visible).</p>
<b>Player B Selected</b>	<b>Player B Not Reachable</b>	<b>Player A Stopped</b>	
<p>2020-05-24 13:43:30 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Selected another player, press On and Play to listen</p> <p>Player Progress: 0%</p>	<p>2020-05-24 13:46:10 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Couldn't connect to server - Remote Control turned Off</p> <p>Player Progress: 0%</p>	<p>2020-05-24 13:55:14 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Player stopped</p> <p>Player Progress: 0%</p> <p>2020-05-24 13:58:00 ● Room: ▲0:05 ▼21:29 Kodi Music Player</p> <p><b>Light/Switch Devices</b></p> <p>Player Playback: Play, Pause, Prev, Next</p> <p>Player Volume: On / Off</p> <p>Player System: Stop, Shutdown</p> <p>Player Remote Control: On / Off</p> <p>Player Selection: Player A, Player B</p> <p><b>Utility Sensors</b></p> <p>Player Playing: Not playing</p> <p>Player Progress: 0%</p>	
<p>Switched from Player A to Player B. The remote control and playback are turned Off. To start listening press On and Play.</p>	<p>Pressed On and Play, but Player B is not turned On. Device Playing displays the error message and the remote control is turned Off.</p> <p>See also the Domoticz log for more info.</p>	<p>Player A stopped. The initial message is Player stopped, which is followed after 10 seconds by message "Not playing" and disabling device System.</p> <p>The remote control is still On.</p> <p>Use the Kodi User Interface or any other to start playing again. If playing, this is recognized by the event and the devices Playing &amp; Progress gets updated after max 10 seconds.</p>	
<b>Player A Shutdown</b>			
<p>If the player is shutdown, the device Playing states "Player shutdown" followed by the message that the server is not reachable, which is followed by turning Off the remote control.</p>			

## Automation Script

A single dzVents Lua script is used to control the Music Player.

The source code of the script, kodi\_music\_player.dzvents, is not included as having around 450 lines.

The way the script works in a nutshell – look at the source for details.

There are 3 triggers:

**timer** = { 'every hour' }

*Function*

Clear the device logs every hour to avoid spoiling the database with entries not used.

**devices** = { IDX\_PLAYBACK, IDX\_SETVOLUME, IDX\_SYSTEM,  
IDX\_REMOTECONTROL, IDX\_PLAYERSELECTION }

*Function*

Handle device changes and submit JSON-RPC API commands via method openURL().  
These are according to the screenshot shared earlier.

**httpResponses** = { CALLBACK\_GETITEM, CALLBACK\_GETPROPERTIES,  
CALLBACK\_PLAYBACK }

*Function*

Handle the callback response of the JSON-RPC API commands.

### Update Devices Playing & Progress

The devices are updated every 10 seconds

### Pseudocode

```
DELAYSECONDS = 10
Trigger openURL(GETITEM, CALLBACK_GETITEM)
Handle CALLBACK_GETITEM
  Run openURL(GETPROPERTIES, CALLBACK_GETPROPERTIES)
  Handle CALLBACK_GETPROPERTIES
  Run openURL(GETITEM, CALLBACK_GETITEM).afterSec(DELAYSECONDS)
```

### Notes

1. The trigger to start device updates is by switching the device “Remote Control” On,
2. The change of the switch device triggers the first openURL for the JSON-RPC API GETITEM,
3. The GETITEM callback runs openURL for the JSON-RPC API GETPROPERTIES,
4. The GETPROPERTIES callback runs GETITEM again, but after a delay, set by the constant DELAYSECONDS to 10 seconds,
5. After the delay, the loop starts again from the GETITEM callback (3).

# Notes

## Purpose

To create notes with a subject & content and functions to save, copy, notify, export and clear. The background developing this function, was to be able to quickly save scribbles / short information when using various devices (PC, Tablets, Smartphones) in the local network. The standard Domoticz HTTP API calls are used for the functions save/copy/notify/export/clear.

It is not possible to change a saved note, delete a selected note (see below under Limitations), because of not available as HTTP API calls.

## Solution

- Text device “Notes” to store the notes
- Switch device “Notes Export” to export the textlog to a local file
- Custom page "Notes.html" with forms and buttons
- dzVents Automation script “Export\_Notes”

The file custom page is stored in folder

```
~domoticz/www/templates
```

which holds all custom pages.

The screenshot shows the Domoticz interface with a sidebar containing 'Custom Page' and 'Devices'. The main area displays the 'Notes' custom page. The page has sections for 'Notes' and 'Log'. In the 'Notes' section, there is a text area with a subject ('TODO: Notes various') and content ('1) select device for dedicated notes...'). Below this are 'Save', 'Copy', and 'Notify' buttons. In the 'Log' section, there is a dropdown menu set to 'TODO: Notes various' and buttons for 'Export' and 'Clear'. At the bottom, a log entry shows a note posted at 14:46:02. To the right, four modal dialogs are shown corresponding to the buttons: [Save], [Copy], [Notify], and [Log]. The [Save] dialog shows a success message. The [Copy] dialog shows the note content. The [Notify] dialog shows a notification message. The [Log] dialog shows the log entry again.

## Form Note

The form contains the elements:

- Input - set the note subject (mandatory).
- Textarea – set the note content (optional).
- Buttons to handle the note subject & content:
  - Save - the note to the assigned Domoticz text device.

- Copy - the note to the clipboard.
- Notify - the note to all enabled notification systems.

## Form Device History Log

The form contains the elements:

- Select - a note from the device history log and display the subject and content in the form Note fields subject & content.
- Buttons for the textlog:
  - Export the textlog to a local file.
  - Clear the text device history log.

Additional elements:

- Label - show status message after any action.
- Button Info - show about.

## Remarks

- Dialogs make use of Bootbox (Bootstrap framework) as also used by Domoticz.
- CSS classes are used from the Domoticz CSS styles (see folder domoticz/www/css).
- IMPORTANT: do not place buttons inside form tags - experienced strange behaviour with buttons or bootbox.

## Format of a Note

The text is stored in the Domoticz database table lightinglog as format: subject|content  
To split the note subject and note content, the delimiter | (pipe) is used.  
The content is by default stored without linefeeds. A placeholder replaces the textarea content “\r\n” with a placeholder text “CRLF”.

When a note is selected from the device history:

- the subject is extracted from the text,
- the delimiter | is removed,
- in the content, the placeholder “CRLF” is replaced by “\r\n” to show proper linefeeds in the textarea.

Example note stored with subject|content delimiter and placeholder CRLF:

This is a Test Note The content of this test note.CRLFThanks.
---

## Limitations

Because standard API calls are used, a note cannot be changed nor deleted. It is only possible to save a note or delete all notes from the database.

This means if f.e. want to change a note, a new note must be created.

It might be possible to change or delete a note via API calls for any future Domoticz releases – but not known so far.

A workaround – but be careful – is direct access the Domoticz database via [SQLite Shell](#).

The notes are stored in table lightinglog.

## Alternatives

Thoughts for another way to store the notes:

JSON array with note objects (like [this](#) Node-RED based solution) stored in an single User Variable, user variable from type String.

```
[{"idx":1, "subject":"My Note 1", "content": "Line 1"}, {"idx":2, "subject":"My Note 2", "content": "Line 1#CRLF#Line 2#CRLF#"}]
```

*Status: Not feasible as the max length of a user variable string is 255 characters.*

Instead storing in a single User variable, store each note in a separate User Variable.

*Status: like previous = notes limited by max 255 characters.*

BUT if the string length limitation is not an issue, as notes are short (headlines like), then these solutions could be an option.

User Variables can be added, updated, and deleted (CRUD commands) via HTTP JSON-API requests.

The vtype=2 and the name & value are to be provided without “” (recommend escaping special characters, i.e. space by %20).

The CRUD commands to be assigned to buttons in the form. The buttons onClick function executes an HTTP JSON-API command.

```
http://domoticz-ip:8080/json.htm?type=command...

ADD
param=adduservariable&vname=USERVARIABLENAME&vtype=USERVARIABLETYPE&vvalue=USERVARIABLEVALUE

UPDATE
param=updateuservariable&vname=USERVARIABLENAME&vtype=USERVARIABLETYPE&vvalue=USERVARIABLEVALUE

DELETE
param=deleteuservariable&idx=IDX

VIEW
param=getuservariable&idx=IDX

Example updating a User Variable named DEF_NOTES from type String
http://domoticz-
ip:8080/json.htm?type=command&param=updateuservariable&vname=DEF_NOTES&vtype=2&vvalue>Hello%20World
```

## Domoticz HTTP API

The HTTP requests to retrieve or store the data make use of Domoticz HTTP API requests submitted via AJAX calls.

## Domoticz Configuration

There are 2 devices required, a text and switch. The switch is used by dzVents automation script “Notes\_Export” to save the log of the text device.

The custom page is stored in

```
~domoticz/www/templates/Notes.html
```

### Text Device

Create a text device to hold the note entries.

Domoticz > GUI > Setup > Hardware > Create Virtual Sensor (Hardware: VirtualSensors):

Name: Notes, Sensor type: Text.

After adding the Domoticz > Setup > GUI lists the new device:

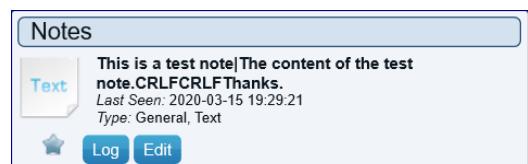
```
Idx=240, Hardware=VirtualSensors, Name=Notes, Type=General, SubType=Text
```

#### Notes

The device idx is required for the custom page "Notes.html".

The device widget is accessible from the Domoticz GUI > Tab Utility.

**Example** with subject, delimiter | to split the subject from the content, CRLF placeholder for the content.



### Switch Device

Create a switch device to export the device log to a text file.

Domoticz > GUI > Setup > Hardware > Create Virtual Sensor (Hardware: VirtualSensors):

Name: Notes Export, Sensor type: Switch.

After adding the Domoticz > Setup > GUI lists the new device:

```
Idx=241, Hardware=VirtualSensors, Name=Notes Export, Type=Light/Switch,  
SubType=Switch
```

Select the device widget and change the Switch Type from On/Off to Push On Button.

## Custom Menu

Enable Domoticz > GUI > Setup > Settings > System > Active Menus: Custom.

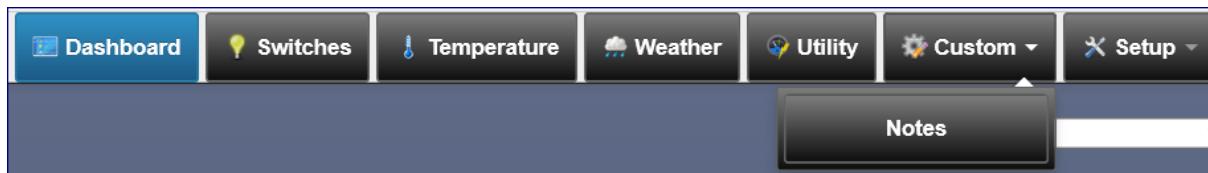
The Domoticz GUI shows the tab Custom.

All custom pages stored in the Domoticz folder

```
~/domoticz/www/templates
```

Are listed in the dropdown menu.

### Screenshot Custom tab and dropdown with only the Notes custom page



## dzVents

Create a new automation script and paste the content of the file *Notes\_Export.lua*.

Save the event as Export\_Notes.

Set the Idx of the devices created:

```
-- idx of the devices (idx,hardware,name,type,subtype,switchtype)
local IDX_NOTES_SWITCH = 241;      --241,VirtualDevices,Notes Save Log,Light/Switch,Switch,push On Button
local IDX_NOTES_TEXT = 240;        --240,VirtualDevices,Notes,General,Text
```

Enable Domoticz > GUI > Setup > Settings > System > Active Menus: Custom.

The Domoticz GUI shows the tab Custom.

All custom pages found in the Domoticz folder

```
~/domoticz/www/templates
```

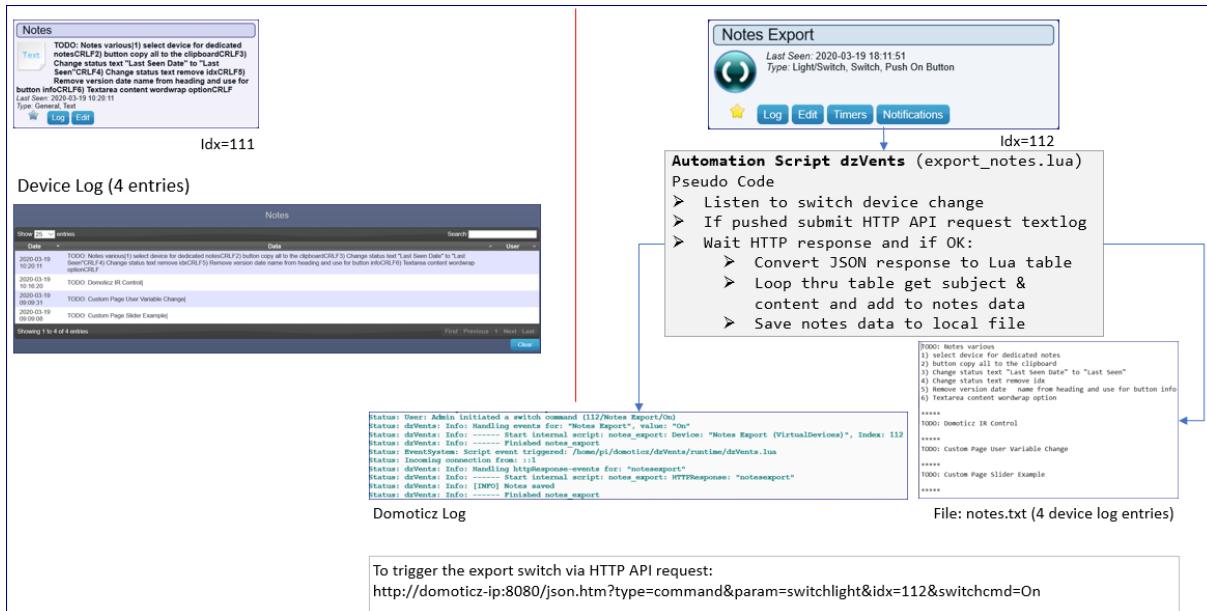
are listed.

## Export Notes

The text log can be exported to a local file which is triggered by the custom page Export button by submitting a HTTP API request to push the switch device on.

The dzVents script listens to changes of the switch device and if on, the text log is requested via HTTP API request. The response is a JSON string which is converted to a Lua table. The Lua table is looped to get and build the notes data. The notes data is stored to a local file.

### Concept with devices configured on the Domoticz development system



# Database Records

## Table

The table *LightingLog* of the Domoticz database holds the records for the function Notes. The field *DeviceRowID* is the idx of the device which is used in SQL statements.

Domoticz database operations can be performed using the [SQLite Shell](#).

### Example Select Notes

```
$ cd domoticz
$ sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
sqlite> .header on
sqlite> select * from lightinglog where devicerowid=240;
DeviceRowID|nValue|sValue|Date|User
240|0|Hello World|2020-03-15 19:04:28|
240|0|This is a test note|The content of the test note.CRLF
240|0|This is a test note 2|The content of the test note 3.CRLF
240|0|This is a test note 3|The content of the test note 3CRLF
CRLF|.2020-03-16 09:09:04|
sqlite> .exit
$
```

To change or delete a note, the *Date* field must be used in the SQL statement WHERE clause together with the *DeviceRowID*.

**... but as mentioned earlier, this function intent is just to save notes and be able to copy or notify ...**

## CRUD Operations

To show how to use CRUD operations on the Domoticz database = Create (Insert), Read (Select), Update, Delete. Included comments (//) for the operations.

```
$cd domoticz
$sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
sqlite> .open domoticz.db
sqlite> .header on

// Select all records for the text device with idx=111 holding the notes
sqlite> select * from lightinglog where devicerowid=111;
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 1|The content of test note 1.CRLF
111|0|This is test note 2|The content of test note 2.CRLF
CRLF|.2020-03-16 16:31:19

// Select a specific record for a date
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-16 16:31:19";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 2|The content of test note 2.CRLF
CRLF|.2020-03-16 16:31:19
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-16 16:31:19";
count(*)
1

// Insert a new record
sqlite> insert into lightinglog (DeviceRowID,nValue,sValue,User,Date)
...> values (111,0,"This is test note 3|The content of test note 3.CRLF
Thanks.",","", "2020-03-17 08:09:10");
```

```
// Check if the new record is inserted
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 3|The content of test note 3.CRLFThanks.||2020-03-17 08:09:10

// Update an existing record using where for a specific deviceid and date
sqlite> update lightinglog
...> set svalue="This is test note 3|The content of test note 3.CRLFUpdate 1CRLFThanks."
...> where date="2020-03-17 08:09:10";

// Check if the record is updated
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 3|The content of test note 3.CRLFUpdate 1CRLFThanks.||2020-03-17 08:09:10
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
count(*)
1

// Delete a record with where for a specific deviceid and date
sqlite> delete from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";

// Check if the record is deleted using count()
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
count(*)
0
sqlite>
```

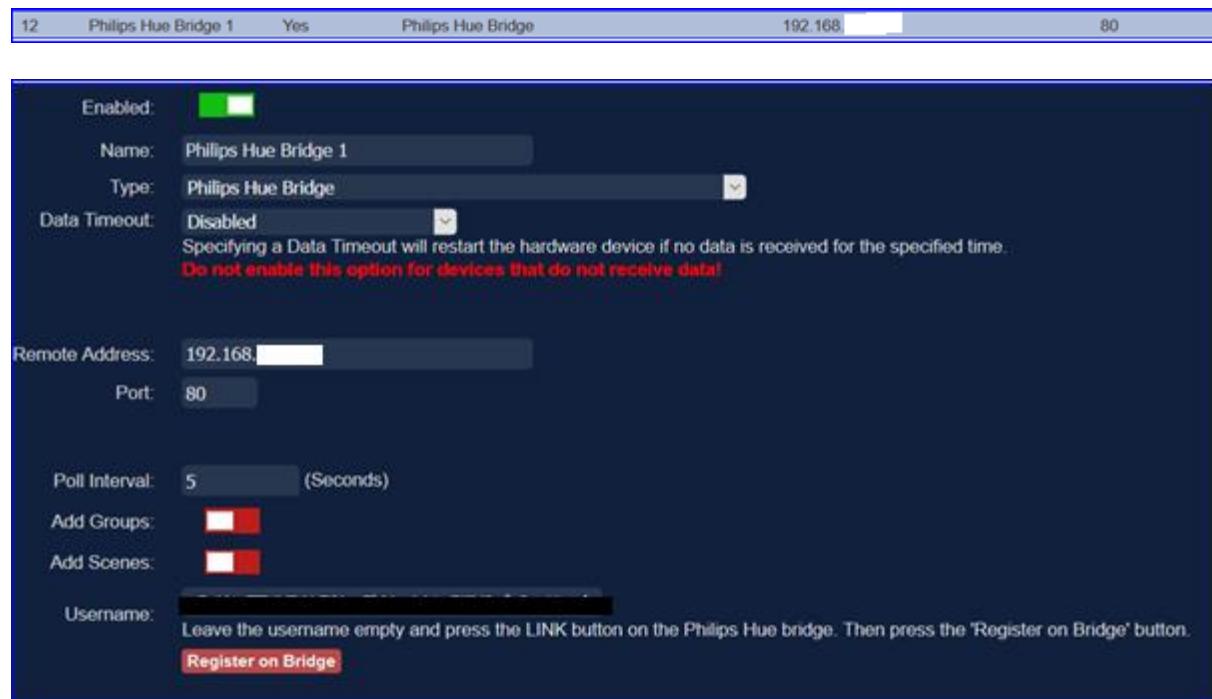
# Philips Hue

## Purpose

To control devices connected to the Philips Hue Bridge.

## Hue Bridge Configuration

There is one Hue Bridge (Hardware IDX=12) connected to the Router.



### Note

Initially Groups & Scenes are not used.

## Hue Devices

The list of Hue Devices, which are automatically created by Domoticz (Setup > Settings > Hardware allow for 5 minutes), in order of their idx:

Idx	Hardware	ID	Unit	Name	Type	SubType
110	Philips Hue Bridge 1	0000001	1	Hue Haustür	Lighting 2	AC
111	Philips Hue Bridge 1	0000002	1	Hue WZ Ablage	Color Switch	WW
112	Philips Hue Bridge 1	0000003	1	Hue WZ TV	Lighting 2	AC
113	Philips Hue Bridge 1	0000004	1	Hue WZ Robs Ecke	Lighting 2	AC
118	Philips Hue Bridge 1	0000005	1	Hue MakeLab	Lighting 2	AC

## Hue Light Add New

Example adding a Hue white light.

First the new Hue Lamp is added to the Hue Bridge.

This is done by using the Hue App (used the Android App ).

In the app, select icon Settings > Light setup > Add light > Search (for new devices).

Tip: if the device cannot be found, then enter the Serial No. (six characters, i.e. 8D80D2) prior pressing Search.

If the lamp is found, a new entry is listed, i.e. Hue color lamp 1 for a White and color ambiance light.

Press symbol I to change the name of the device.

After the Hue Lamp has been added to the Hue Bridge, Domoticz adds the device (IDX=118) to the list (after Setup > Settings > Hardware allow for 5 minutes).

### Notes

This might take few minutes.

If the devices is not added, try to disable / enable the Hardware Philips Hue.

Check the Domoticz log if the device is listed.

### Example

118	Philips Hue Bridge 1	0000005	1	Hue white lamp 4	Lighting 2	AC
-----	----------------------	---------	---	------------------	------------	----

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
194	Philips Hue Bridge 1	0000007	1	Hue Bloom Schrankwand	Color Switch	RGBW	Set Color

### Domoticz Log

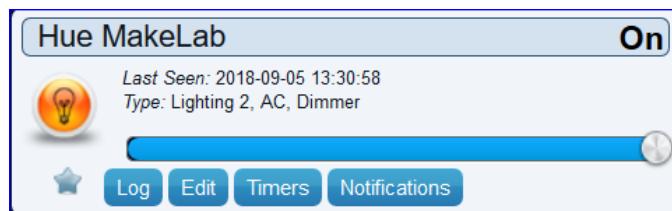
Listen to new devices and add the new Hue white lamp 4.

2018-09-05 13:22:14.995 Status: New sensors allowed for 5 minutes...
2018-09-05 13:25:41.749 (Philips Hue Bridge 1) Lighting 2 (Hue white lamp 4)

In the list of devices (GUI > Setup > Devices>, add the device (i.e. sets the device to used) with name “Hue MakeLab”.

To check if the device is working:

Select GUI > Switches > Widget “Hue MakeLab” and set the level to 100%.



118	Philips Hue Bridge 1	0000005	1	Hue MakeLab	Lighting 2	AC	Set Level: 100 %
-----	----------------------	---------	---	-------------	------------	----	------------------

### Domoticz Log

Hue Lamp Level Change

2018-09-05 13:37:55.133 Status: User: Admin initiated a switch command (118/Hue MakeLab/Set Level)
--

## Hue Control Tests

Let us begin playing around with dzVents and a Hue Lamp Hue MakeLab (IDX=118).  
The Hue Lamp can be handled like a dimmer switch, with methods to be used like  
device.dimTo(pct), switchOff(), switchOn().  
See [this](#) documentation, chapter Switch.

To get an overview of the object properties, use device.dump() first.

secondsSinceMidnight: 49075 ruleIsBeforeCivilTwilightStart() daysAgo: 0 updateBarometer() setState() kodiSwitchOff() description: updateMode() hardwareType: Philips Hue Bridge updateRadiation() updateWind() dump() updateElectricity() updateWaterflow() updateYouless() updateVoltage() updateVisibility() stop() playFavorites() updateUV() updatePercentage() isGroup: false updateCustomSensor() deviceId: 0000005 changed: true updateText() updateLux() deviceType: Lighting 2 updateTempHumBaro() isScene: false isHTTPResponse: false updateSetPoint() updateTempHum()	updateSoilMoisture() bState: false isTimer: false usedByCamera: false unit: 1 update() armAway() setNightMode() setDiscoMode() setWhiteMode() baseType: device setKelvin() kodiSetVolume() updateP1() hardwareID: 12 updateAirQuality() isSecurity: false play() deviceSubType: AC hardwareTypeValue: 38 levelVal: 5 kodiStop() lastLevel: 33 updateCounter() setVolume() cancelQueuedCommands() updatePressure() active: false data: lastUpdate: 2018-09-05 13:37:55 description: data: hardwareTypeValue: 38	unit: 1 _state: Off hardwareType: Philips Hue Bridge maxDimLevel: 15 icon: dimmer levelVal: 5 usedByCamera: false protected: false _nValue: 0 hardwareName: Philips Hue Bridge 1 hardwareID: 12 id: 118 switchType: Dimmer baseType: device signalLevel: 12 name: Hue MakeLab switchTypeValue: 7 deviceID: 0000005 lastLevel: 33 changed: true batteryLevel: 255 deviceType: Lighting 2 rawData: 1: 15 subType: AC timedOut: false id: 118 rawData: 1: 15 isVariable: false _nValue: 0
---	---	--

## Automation Script – Device Trigger

If the Hue Lamp is switched ON, then switch OFF again.

Event name: hue\_control

```
local IDX_HUE_MAKELAB = 118;

return {
  on = {
    devices = {
      IDX_HUE_MAKELAB
    }
  },
  execute = function(domoticz, device)
    domoticz.log('Device ' .. device.name .. ' changed to ' .. device.state, domoticz.LOG_INFO)
    if (device.state == 'On') then
      domoticz.log('Switching OFF', domoticz.LOG_INFO)
      device.switchOff()
      domoticz.log('OK', domoticz.LOG_INFO)
    end
    -- domoticz.log(device.dump(), domoticz.LOG_INFO)
  end
}
```

## Automation Script – Timer Trigger

Switch the Hue Lamp ON at 'HH:MM' and switch it OFF again one minute later.

This type of script could be used, to switch

- the Hue Lamps in the Living Room, ON 'at sunset' and OFF 'at 23:00'.
- The Hue Lamp Outside ON 'at sunset' and OFF 'at sunrise'.

Event name: hue\_control2

```
-- Idx of the devices
local IDX_HUE_MAKELAB = 118;

return {
  active = true,
  on = {
    timer = {
      'at 15:49',
      'at 15:50'
    }
  },
  execute = function(domoticz, timer)
    if (timer.trigger == 'at 15:49') then
      domoticz.devices(IDX_HUE_MAKELAB).switchOn()
    end

    if (timer.trigger == 'at 15:50') then
      domoticz.devices(IDX_HUE_MAKELAB).switchOff()
    end
  end
}
```

## Hue Timed Switch Lights

Switch the Hue lights in the living room ('WZ'), ON '30 minutes before sunset' and OFF 'at 23:00'.

Update the Alert Message.

### Automation Script

Event name: hue\_wz\_timer

```
--[[[  
    hue_wz_timer.lua  
    Turn WZ Hue lamps on & off by given time: ON at sunset, OFF at 23:00.  
    Update the alert message.  
    Ensure the timer and the timer trigger are the same.  
]]--  
  
-- External modules  
local utils = require('utils')  
local msgbox = require('msgbox')  
  
-- Idx of the devices  
local IDX_HUE_WZABLAGE = 111  
local IDX_HUE_WZTV = 112  
local IDX_HUE_HAUSTUER = 110  
  
-- Update alert message with alert level green.  
local function setalertmsg(domoticz, state)  
    local message= 'Hue Lampen ' .. state .. ' ' .. utils.isnowhhmm(domoticz)  
    msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)  
end  
  
local function hueswitchon(domoticz)  
    domoticz.devices(IDX_HUE_WZABLAGE).switchOn()  
    domoticz.devices(IDX_HUE_WZTV).switchOn()  
    domoticz.devices(IDX_HUE_HAUSTUER).switchOn()  
    setalertmsg(domoticz, 'an')  
end  
  
local function hueswitchoff(domoticz)  
    domoticz.devices(IDX_HUE_WZABLAGE).switchOff()  
    domoticz.devices(IDX_HUE_WZTV).switchOff()  
    domoticz.devices(IDX_HUE_HAUSTUER).switchOff()  
    setalertmsg(domoticz, 'aus')  
end  
  
return {  
    active = true,  
    on = {  
        timer = {  
            '60 minutes before sunset',  
            '30 minutes before sunset',  
            'at 23:00',  
            'at 07:00',  
            'at 08:00'  
        }  
    },  
    execute = function(domoticz, timer)  
  
        -- in the evening send message hue lights will be switched on  
        if (timer.trigger == '60 minutes before sunset') then  
            -- format date & time see www.lua.org/pil/22.1.html. X = time.  
            local now=os.time()  
            local nowplus30minutes = utils.converttimehhmm(os.date("%X",now+(30*60)))  
            local message= 'Hue Lampen gehen an ' .. nowplus30minutes  
            msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)  
        end  
  
        -- switch the hue lights on & off in the evening  
        if (timer.trigger == '30 minutes before sunset') then hueswitchon(domoticz) end  
        if (timer.trigger == 'at 23:00') then hueswitchoff(domoticz) end  
  
        -- switch the hue lights on & off in the morning
```

```
-- NOT USED
-- if (timer.trigger == 'at 07:00') then hueswitchon(domoticz) end
-- if (timer.trigger == 'at 08:00') then hueswitchoff(domoticz) end

end
}
```

## Domoticz Log

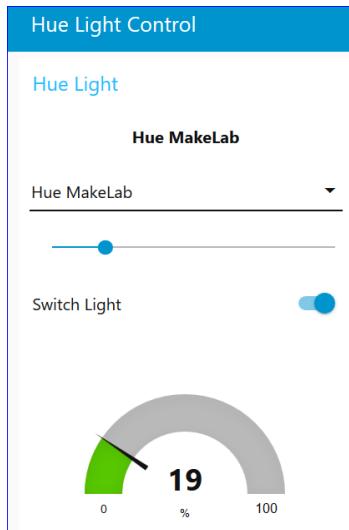
```
2018-09-05 20:02:00.394 (Philips Hue Bridge 1) Color Switch (Hue WZ Ablage)
2018-09-05 20:02:00.417 (Philips Hue Bridge 1) Lighting 2 (Hue WZ TV)
2018-09-05 20:02:00.259 Status: dzVents: Info: ----- Start internal script: hue_wz_timer:, trigger: at sunset
2018-09-05 20:02:00.276 Status: dzVents: Info: ----- Finished hue_wz_timer
2018-09-05 20:02:00.357 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

## Hue Lights Control Node-RED

An example of controlling Hue lights via Node-RED Dashboard UI in a browser. Select a Hue and switch the light On|Off or set Brightness Level.

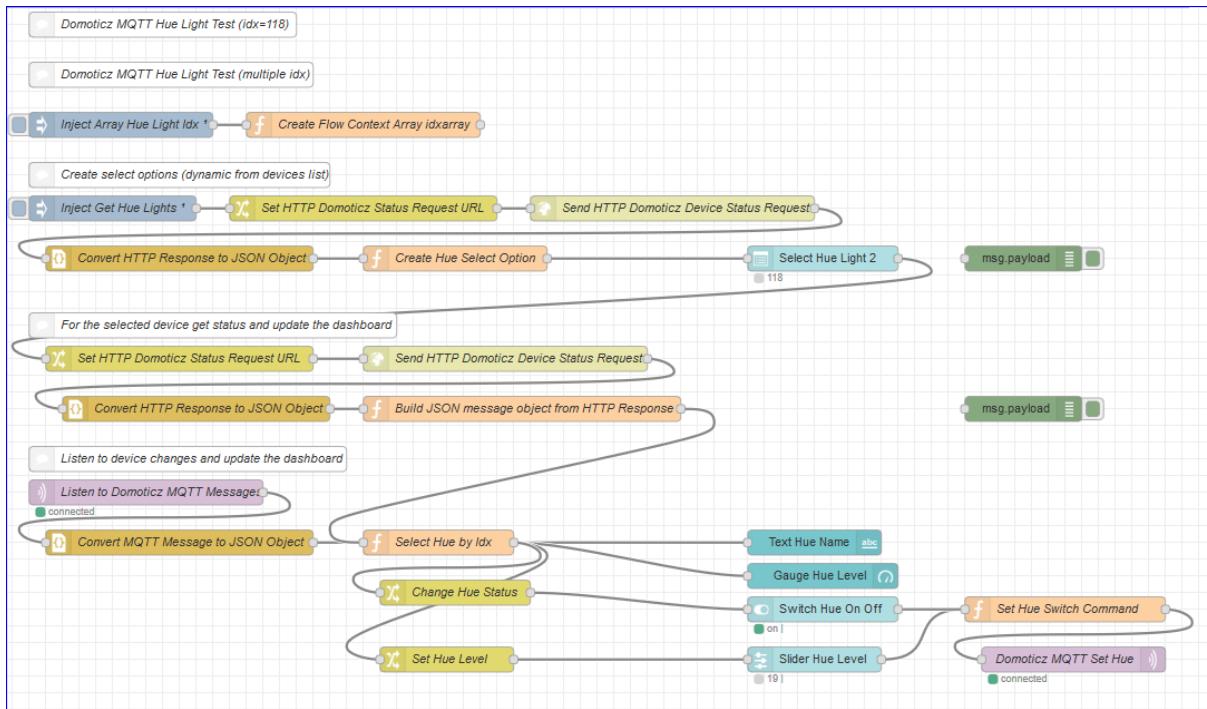
Read more about Node-RED [Appendix Node-RED](#).

### Dashboard



The dashboard ui contains the tab Hue Light Control with a single group Hue Light. The dropdown lists the Hue devices to control. If selected the state gets updated first, i.e the slider position, switch and the gauge level are set. Change the light by moving the slider or set the switch. If the device gets updated in the browser by the Domoticz GUI, then the Node-RED dashboard is updated.

## Flow



### Note

When the flow starts, a flow context array “idxarray” is defined with the Hue deviceidx’es, which are used for the dropdown with the Hue lights. An initial Hue is set in the dropdown. The dropdown notifies a change and requests the selected Hue device information via HTTP request. The HTTP response contains the full device information from which only a subset are used. A new JSON message is created with the required properties. These are sent to the node selecting the Hue. The properties are used to update the Dashboard UI nodes, i.e. text\_ui, gauge\_ui, switch\_ui and slider\_ui.

If a Hue device changes (f.e. via the browser Domoticz GUI), the MQTT listener node recognized this and the message payload is handled also by the node selecting the Hue with the connected nodes.

So, the “Select Hue by Idx” node has two inputs

1. from a converted HTTP request and
2. from the MQTT listener.

# Hue Direct Access

## Introduction

Next some examples on how to direct access (control) a Hue Light without the use of the Domoticz Hardware “Philips Hue Bridge”.

The direct access possibilities are explored more out of curiosity.

To enable direct access with the Philips Hue Bridge, following information is required:

- the IP address of the Philips Hue Bridge
- a Philips Hue authorized username

Recommend to read in depth about Philips Hue Development at [MeetHue](#) and [Get Started](#).

*Abbreviations used in the next examples*

- Hue = Philips Hue Light
- Hue 5 = Hue light number 5, named "Hue white lamp 4", from the list of Hue devices

For the next examples, HTTP commands are used to direct access the Hue 5.

Supported are four HTTP methods (see documentation Philips MeetHue).

- GET: fetch all information about the addressed resource
- PUT: modify an addressed resource
- POST: create a new resource inside the addressed resource
- DELETE: delete the addressed resource

The HTTP response is JSON formatted string.

## HTTP - Browser

The communication with the Hue Bridge is direct using HTTP.  
The HTTP commands are submitted from a browser.

### HTTP GET Request State Hue 5

Request

```
http://hue-bridge-ip/api/username/lights/5
```

Result

```
{
  "state": {"on": true, "bri": 89, "alert": "select", "mode": "homeautomation", "reachable": true},
  "swupdate": {"state": "noupdates", "lastinstall": "2018-12-05T17:48:30"},
  "type": "Dimmable light",
  "name": "Hue white lamp 4",
  "modelid": "LWB010",
  "manufacturername": "Philips",
  "productname": "Hue white lamp",
  "capabilities": {"certified": true},
  "control": {"mindimlevel": 2000, "maxlumen": 806},
  "streaming": {"renderer": false, "proxy": false},
  "config": {"archetype": "classicbulb", "function": "functional", "direction": "omnidirectional",
  "startup": {"mode": "safety", "configured": true}},
  "uniqueid": "00:17:88:01:03:b5:e2:68-0b",
  "swversion": "1.46.13_r26312",
  "swconfigid": "322BB2EC",
  "productid": "Philips-LWB010-1-A19DLv4"
}
```

### HTTP PUT Request Switch Hue 5

To change the state of a Hue light, submit an HTTP PUT request with a message body containing the state action as JSON string.

URL

```
http://hue-bridge-ip/api/username/lights/lightnr/state
```

Body Payload JSON

```
{"on":true}
```

or

```
{"on":false}
```

## HTTP - curl

The communication with the Hue Bridge is direct using a *curl command* requesting HTTP PUT with JSON Data.

*curl* is a tool to transfer data from or to a server, using a protocol, i.e. FTP, HTTP and many more.

The *curl command* is submitted on a Raspberry Pi from the CLI (Command Line Interface, i.e. terminal).

### Action: curl HTTP PUT request switching Hue with ID 5 ON | OFF

#### curl Command Switch Hue 5 ON

```
curl --request PUT --data "{\"on\":true}" http://hue-bridge-ip/api/username/lights/5/state
```

Response

```
[{"success": {"/lights/5/state/on": true}}]
```

#### curl Command Switch Hue 5 OFF

```
curl --request PUT --data "{\"on\":false}" http://hue-bridge-ip/api/username/lights/5/state
```

Response

```
[{"success": {"/lights/5/state/on": false}}]
```

## HTTP - dzVents Lua Script

To access Hue 5, two dzVents Automation scripts are shared:

- Get State Hue 5 – using openURL
- Put State Hue 5 – using os.execute curl command

### **IMPORTANT**

State 20200122, Domoticz V4.11637:

The function openURL does not support the HTTP PUT method (only GET, POST).

To change the state of a Hue, PUT is required (see previous info under Hue Direct Access).

### **dzVents Lua Script State Hue 5 – HTTP**

```
-- Hue Light Control
-- Test to obtain the state of a hue devices via direct http rest request (GET method)
-- The request is triggered by a switch with idx=5

local HUEBRIDGEIP = 'hue-bridge-ip'
local HUELIGHT5RES = "huelight5"
local HUEUSERNAME = "*****"
local HUELIGHTNR = 5
local HUELIGHT5IDX = 73

return {
  on = {
    devices = { HUELIGHT5IDX },
    httpResponses = { HUELIGHT5RES }
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
      local urlRequest = string.format("http://%s/api/%s/lights/%s",HUEBRIDGEIP,HUEUSERNAME,HUELIGHTNR)
      domoticz.log(urlRequest)
      domoticz.openURL({url = urlRequest,method = 'GET',callback = HUELIGHT5RES,})
    end

    if (item.isHTTPResponse) then
      domoticz.log('HUELIGHT5 Status Code = ' .. item.statusCode .. ', Name=' .. item.callback)
      if (item.statusCode == 200 and item.callback == HUELIGHT5RES) then
        if (item.isJSON) then
          -- This will give the full device status: domoticz.log('Item data:' .. item.data)
          -- Let's pick state and bri
          local result = item.json
          domoticz.log(string.format("Hue Light 5 state:%s, bri:%s",result.state.on, result.state.bri))
        end
        return
      else
        domoticz.log('[ERROR] Handling the request:' .. item.statusText, domoticz.LOG_ERROR)
        -- domoticz.log(item, domoticz.LOG_ERROR)
      end
    end
  end
}
```

This is an example of a JSON string returned, from which attributes can be selected, i.e.

```
local result = item.json
result.state.on
result.state.bri
```

```
{
  ["manufacturername"]="Philips", ["productname"]="Hue white lamp",
  ["state"]=
  {
    ["mode"]="homeautomation", ["on"]=false, ["alert"]="select", ["reachable"]=false, ["bri"]=102,
    ["swupdate"]=[{"lastinstall"]="2018-12-05T17:48:30", ["state"]="noupdates"
  },
  ["capabilities"]=
  {
    ["certified"]=true, ["control"]=[{"mindimlevel"}=2000, ["maxlumen"]]=806
  },
  ["streaming"]=
  {
    ["proxy"]=false, ["renderer"]=false
  },
  ["uniqueid"]="00:17:88:01:03:b5:e2:68-0b",
  ["modelid"]="LWB010",
  ["name"]="Hue white lamp 4",
  ["swconfigid"]="322BB2EC",
  ["config"]=
  {
    ["archetype"]="classicbulb",
    ["startup"]=
    {
      ["configured"]=true, ["mode"]="safety"
    },
    ["direction"]="omnidirectional",
    ["function"]="functional"
  },
  ["swversion"]="1.46.13_r26312",
  ["productid"]="Philips-LWB010-1-A19DLv4",
  ["type"]="Dimmable light"
}
```

## dzVents Lua Script Switch Hue 5 – curl

```
-- Hue Light Control
-- Test direct switching on off via curl with json command and http put request

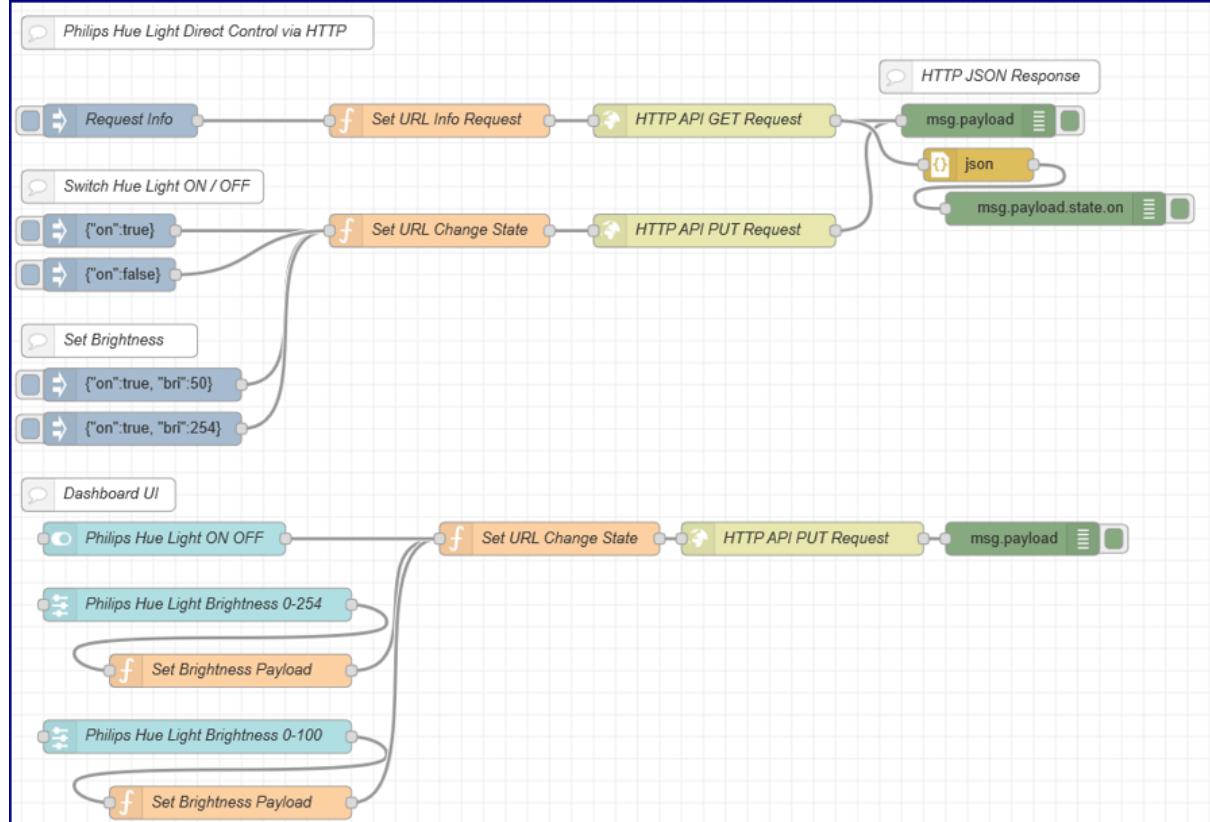
local HUEBRIDGEIP = 'hue/bridge/ip'
local HUELIGHT5RES = "huelight5"
local HUEUSERNAME = "*****"
local HUELIGHTNR = 5
local HUELIGHT5IDX = 73

return {
  on = {
    devices = { HUELIGHT5IDX },
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
      -- curl --request PUT --data "{\"on\":true}" http://hue-bridge-ip/api/username/lights/5/state
      local switchState = true
      if (domoticz.devices(HUELIGHT5IDX).state == "Off") then
        switchState = false
      end
      local curlRequest = string.format('curl --request PUT --data "{\"on\":%s}"'
http://%s/api/%s/lights/%s/state',switchState,HUEBRIDGEIP,HUEUSERNAME,HUELIGHTNR)
      domoticz.log(curlRequest)
      local handle = os.execute(curlRequest)
      domoticz.log(handle)
    end
  end
}
```

## HTTP - Node-RED

Example Node-RED flow to switch or set the brightness of Hue 5.

Source: hue\_direct\_control\_example.flow



### Switch Hue 5

Flow nodes: inject > function > http request > debug

### Inject Node Set Payload – the message body

```
{“on”:true}
OR
{“on”:false}
```

### Function Node Set URL Change State

```
// Set the url message
// Example: http://hue-bridge-ip/api/username/lights/lightnr/state
const urlBridge = “http://hue-bridge-ip/api/”;
const 212sername = “username”;
const lightNr = 5;
msg.url = urlBridge + 212sername + “/lights/” + lightNr + “/state”;
node.warn(msg.url);
return msg;
```

### HTTP Request Node

Method PUT

URL is set by the previous function node.

**Debug Node**

Log the HTTP request result.

Example switch light off

```
[{"success":{"/lights/5/state/on":false}}]
```

**Node-RED Brightness Hue 5**

Basically, as previous, but the message body, which is the payload, with JSON properties on = true and bri = 76.

```
{"on":true, "bri":76}
```

with HTTP result

```
[{"success":{"/lights/5/state/on":true}}, {"success":{"/lights/5/state/bri":76}}]
```

## Hue Color Setting

A Hue White & Color Ambiance bulb enables to set 16 million color.

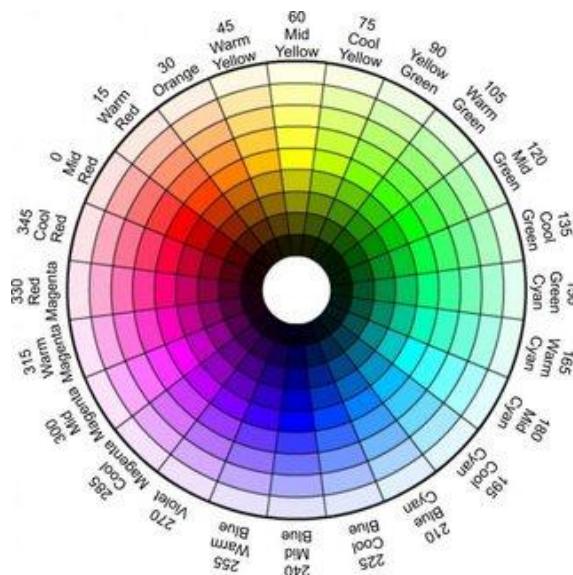
The Domoticz device is Type Color switch, SubType RGBWW, SwitchType Dimmer.

Example:

Idx	Hardware	ID	Unit	Name	Type	Sub Type	Data
355	Philips Hue Bridge 1	00000009	1	Wohnen Luft	Color Switch	RGBWW	Set Color

## Color Wheel

One way to set the color is by using a color wheel. Hue is a number between 0 and 360 degrees



Source: Unknown

In dzVents, the color for the device, can be set with the function:

```
setHue(hue, brightness, isWhite)
```

### Example dzVents Snippet with function setHue

```
local IDX_IAQHUE = 355
if domoticz.devices(IDX_IAQHUE).state ~= 'Off' then
    -- Set color to 30 = Orange with actual device brightness
    -- There is no saturation parameter
    domoticz.devices(IDX_IAQHUE).setHue(105, domoticz.devices(IDX_IAQHUE).level, false)
end
```

The Domoticz log shows the function setcolbrightnessvalue

```
setcolbrightnessvalue: ID: 163, bri: 2, color: '{m: 3, RGB: ff7f00, CWWN: 0000, CT: 0}'
```

## RGB

### Example dzVents Snippet with function setColorBrightnessValue via HTTP API Request

```
setColorBrightnessValue(r, g, b, cw, ww, m, t)
```

```
Parameter
r: 0..255, Red level
g: 0..255, Green level
b: 0..255, Blue level
br: 0..100, brightness
cw: 0..255, Cold white level
ww: 0..255, Warm white level (also used as level for monochrome white)
m:
 0=Illegal.
1=White; Valid fields: none.
2=White with color temperature; Valid fields: t.
3=Color; Valid fields: r, g, b.
4=Custom color + white; Valid fields: r, g, b, cw, ww, depending on device capabilities.
t= 0..255, Color temperature (warm / cold ratio, 0 is coldest, 255 is warmest)
```

*Note*

Parameter “r, g, b” required, others optional.

Parameter “br” is handled by the dedicated parameter “brightness” in the URL.

```
local IDX_IAQHUE = 355 -- Hue device
local SWITCHCOLOR = 11 -- Switch triggering to set the hue color via http api request
return
{
  on = { devices = { SWITCHCOLOR }, },
  logging = { level = domoticz.LOG_INFO, marker = 'HUESETCOLOR', },

  execute = function(domoticz, item)
    -- Create table with color properties
    -- Color examples:
    -- Yellow r=255,g=255,b=0,cw=0,ww=0
    -- Orange r=255,g=128,b=0,cw=0,ww=0
    -- Green r=0,g=255,b=0,cw=0,ww=0
    -- White r=0,g=255,b=255,cw=0,ww=0
    c = {}
    c['r']=255
    c['g']=255
    c['b']=255
    c['br']=50
    c['cw']=0
    c['ww']=0
    c['m']=3
    c['t']=0
    -- The table is converted to a JSON color object
    urlRequest = ("http://domoticz-
ip:port/json.htm?type=command&param=setcolbrightnessvalue&idx=%d&color=%s&brightness=%d"):format(
      IDX_IAQHUE,
      domoticz.utils.toJSONString(c),
      c['br'])
    domoticz.log(urlRequest)
    domoticz.openURL({url = urlRequest})
  end
}
```

## Change Color every min 2 secs

Source: hue\_indicator.dzvents

```
--[[  
hue_indicator  
Test to switch a hue color bulb every minute to red for 2 seconds' else light green.  
This solution could be used, to indicate every hour the indoor air quality index for one minute.  
  
Timer Event  
1. Set Hue color to red via HTTP request  
2. Handle HTTP request to set color to green after 2 seconds via HTTP request  
3. Do NOT handle the HTTP request  
4. After 1-minute start at step 1  
]]--  
  
-- Idx devices - the hue device must a color switch RGBWW dimmer  
local IDX_HUE = 355  
  
local HTTPRESPONSE_SETHUEINDICATOR = "SETHUEINDICATOR"  
local HTTPRESPONSE_SETHUENORMAL = "SETHUENORMAL"  
local TIMERRULE = 'every minute'  
local INDICATORDELAY = 2      -- seconds  
  
-- Set the color of a hue bulb  
-- /json.htm?type=command&param=setcolbrightnessvalue&idx=99&hue=274&brightness=40&iswhite=false  
-- hue degrees: 0..360, brightness: 0..100, delay: seconds, httpresponse: as defined in the constants  
local function setLight(domoticz, hue, brightness, delay, httpresponse)  
    urlRequest = ("http://domoticz-  
ip:port/json.htm?type=command&param=setcolbrightnessvalue&idx=%d&hue=%d&brightness=%d&iswhite=false"):f  
ormat(  
    IDX_HUE, hue, brightness)  
    domoticz.openURL({url = urlRequest, method = "GET", callback = httpresponse }).afterSec(delay)  
end  
  
return {  
  
on = {  
    timer = { TIMERRULE },  
    httpResponses = {HTTPRESPONSE_SETHUENORMAL, HTTPRESPONSE_SETHUEINDICATOR},  
},  
data =  
{  
    -- Data with Lua tables. Access: domoticz.data.hueNormal.hue  
    -- hue: 105 warm green, 0 mid red  
    hueNormal = { initial = {[['state']] = "Off", ['hue'] = 105, ['brightness'] = 20} },  
    hueIndicator = { initial = {[['state']] = "On", ['hue'] = 0, ['brightness'] = 60} }  
},  
logging =  
{  
    level = domoticz.LOG_INFO,  
    marker = 'HUEINDICATOR',  
},  
  
execute = function(domoticz, item)  
    -- Timer Trigger  
    if item.isTimer then  
        -- Set the indicator color via URL with 0 delay = immediate  
        setLight(domoticz,  
            domoticz.data.hueIndicator.hue, domoticz.data.hueIndicator.brightness,  
            0, HTTPRESPONSE_SETHUEINDICATOR)  
    end  
    -- HTTP Response Trigger: Handle only the response from the indicator response  
    if item.isHTTPResponse and item.trigger == HTTPRESPONSE_SETHUEINDICATOR then  
        -- Set the hue to normal color via URL after NN seconds  
        setLight(domoticz,  
            domoticz.data.hueNormal.hue, domoticz.data.hueNormal.brightness,  
            INDICATORDELAY, HTTPRESPONSE_SETHUENORMAL)  
    end  
end  
}
```



# Postbox Notifier (HmIP-SWDO)

## Purpose

To watch the postbox for post and notify via Email Notification and Alert Message.

## Solutions

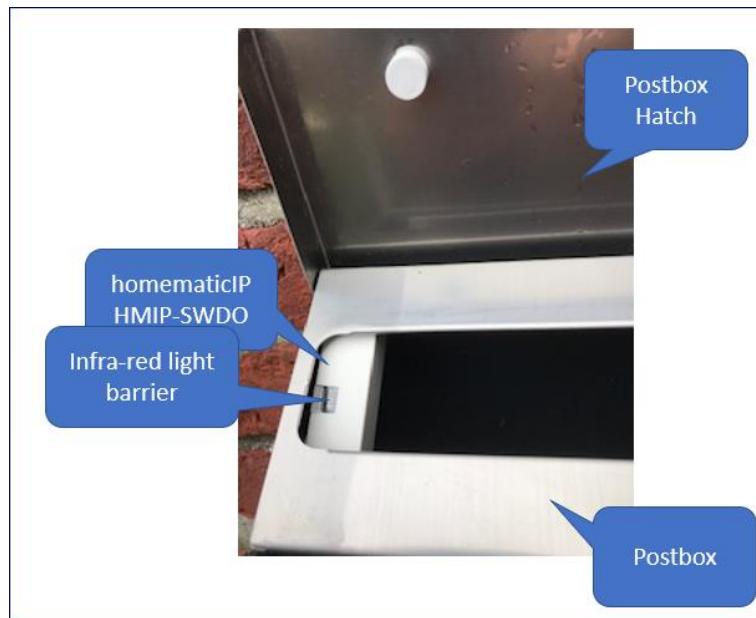
A detector (Homematic IP Window and Door Contact with an integrated infra-red light barrier) detects opening the hatch of the postbox and triggers sending an Email Notification displayed in the [Alert Message](#).

This solution is part of a RaspberryMatic CCU (Homematic compatible) which has several connected HomematicIP devices communicating with/between Domoticz via

- Homematic CCU XML-API Addon or
- Domoticz HTTP API Requests

See also chapter Explore [RaspberryMatic](#).

The hardware used is a *HomematicIP Window and Door Contact – optical* (HMIP-SWDO, Product ref: 140733A09) and Metal Postbox with dimensions (W x H x D) 35 x 40 x 10 cm. The postbox is ~40 m distance from the CCU (which is in range of 300 m according HMIP-SWDO datasheet).



### Hint

The device HmIP-SWDO is light sensitive, means if its dark then there might be no signal. If that is an issue, use the device HmIP-SWDM = window / door contact which magnet.

Whilst evolving, several solutions have been developed. The third solution is in place.

Solution	Description
Alert	If the postbox is opened, set the Alert message, and send email notification (no manual intervention required).
Alert + Switch	As previous plus functionality: Turn a switch on, indicating the postbox has been opened. Manual turn the switch to off after checking the postbox. This solution handles the situation where the postbox latch is kept open instead of closing after post has been dropped. If the latch remains open, the HomematicIP SWDO device keeps on sending messages. This is avoided by sending a message onl once and manual reset the switch after the postbox has been checked.
Alert + Switch + Voltage <b>This solution is in place.</b>	As previous plus functionality: Monitor the voltage of the HomematicIP device SWDO and send Alert if the value is below threshold. The threshold is a Domoticz User Variable. The HomematicIP device SWDO is not an outdoor device. The first experience showed that the battery (1.5V) is getting low after ~3 month and the device is not reachable by the RaspberryMatic system.

## Solution Alert

### Communication Flow

1. HMIP-SWDO detects opening the hatch of the postbox.
2. HMIP-SWDO triggers a script which submits a HTTP API Request to the Domoticz system to update Alert Message device to Level 4 with message “Postbox opened”.
3. Domoticz updates the device and sends out Email notification as triggered by comparing Alert Level 4 against user variable TH\_ALERTTOEMAIL (which has integer value 4).

#### Issue

If the hatch remains open, the HMIP-SWDO keeps on sending alert messages. To avoid this, the alert message script checks if an alert is sent again within a short timeframe threshold, but

- in case other alert messages are received, these are not captured and
- also, if the threshold is reached, alerts are still sent.

Therefor option two is developed which requires manual intervention.

### RaspberryMatic CCU Configuration

#### Step 1

The device HMIP-SWDO is added to the CCU following the Teach-in devices procedure (used entering the Key and SGTIN data).

The screenshot shows two parts of the RaspberryMatic interface:

- (1) Add new device:** A screenshot of the "Device inbox" page. It shows a table with columns: Type description, Picture, Description, Serial number, Interface category, Transmission mode, Name, Function, Room, Functional test, Action, and Done. A row is selected for "HMIP-SW DO". On the right, there are buttons for Test, Delete, OK, Set, and Done. A blue oval highlights the "Add new device" link at the top left of the table.
- (2) Device Status and control:** A screenshot of the "Status and control > Devices" page. It shows a table with columns: Channel, Room, Function, Last modified, and Control. A row for "HMIP-SWDO 0000DA498D5859:1" is selected. The "Control" column contains "Open" and "Closed" buttons. A blue oval highlights the "Status and control" tab at the top left of the table.

#### Step 2

Create a RaspberryMatic program to trigger sending the Alert Message Level 4 (RED) to Domoticz in case the device state changes to open.

Domoticz handles the Alert via dzVents Lua script event “alertmsg\_monitor”.

```
IF HMIP-SWDO..:1 is OPEN THEN trigger WHEN updated SCRIPT
```

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
SWDOS5859	Postbox Alert	Channel status: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b> Device selection: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated   				
<b>Activity: Then...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). Script: Function: Postbox Alert string sFunction = "Postbox Alert" ... immediately				
<b>Activity: Else...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				

## RaspberryMatic Script

The script makes use of a CUxD device (see Addon CUxD) to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz Alert Message device (IDX=55) with Alert Level 4 (nvalue) and the message (svalue):

```
wget -q -O - 'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=IDX&nvalue=4&svalue=MESSAGE'
```

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

### Script Code

#### RaspMatic Script: postbox\_notifier.script

```
string sFunction = "Postbox Alert";
string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! A domoticz alert sensor is used
string nIdx = 55; ! Type: General; SubType: Alert
string nLevel = 4; ! Red
string sText = sDate # "%20" # sTime # "%20-%20Postbox%20opened!";
! Build the domoticz HTTP API Request url
string cAmp = "&";
string sDomUrl = "http://domoticz-ip:8080/json.htm"; ! Production
string sUrl =
sDomUrl#"?type=command#cAmp#param=udevice#cAmp#idx="#nIdx#cAmp#"nvalue="#nLevel#cAmp#"svalue="#sText#"";

! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: script running on CCU waits until completion=ensure not to execute commands take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RET").State();
! WriteLine("VT="#sRes.VarType()#"/"#sRes); ! VT=4, {"status" : "OK","title" : "Update Device"}
! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
  dl.Variable(sFunction#"-Last update OK")
}
else {
  dl.Variable(sFunction#"Last update ERROR")
};
! WriteLine("VT="#dl.VarType()); ! VT=9
! WriteLine(dl.Variable()); ! shows the last update string
```

## Domoticz Configuration

This solution makes use of the [Alert Message](#). No additional configuration required.

## Automation Script

Event name: alertmsg\_monitor

```
--[[  
alertmsg_monitor.lua  
If level (nValue) of the Alert Messages device > threshold  
(set by uservariable TH_ALERTTOEMAIL), send email notification.  
]]--  
  
-- Idx of the devices used  
local IDX_ALERTMSG = 55  
local IDX_TH_ALERTTOEMAIL = 14  
  
return {  
    -- Check which device(s) have a state change  
    on = {  
        devices = { IDX_ALERTMSG}  
    },  
    data = {  
        -- keep the prev notified date  
        prevtimenotified = { initial = os.time() }  
    },  
    execute = function(domoticz, device)  
        -- Send email notification for level = 4  
        -- (or use other level but then change uservar TH_ALERTTOEMAIL)  
        -- Only the subject is used in the email notification  
        if (device.nValue == domoticz.variables(IDX_TH_ALERTTOEMAIL).value)  
        and (domoticz.helpers.timediff(domoticz.data.prevtime) > 60) then  
            domoticz.notify(device.text, '', domoticz.PRIORITY_HIGH)  
            domoticz.data.prevtime = os.time()  
        end  
    end  
}
```

*Note*

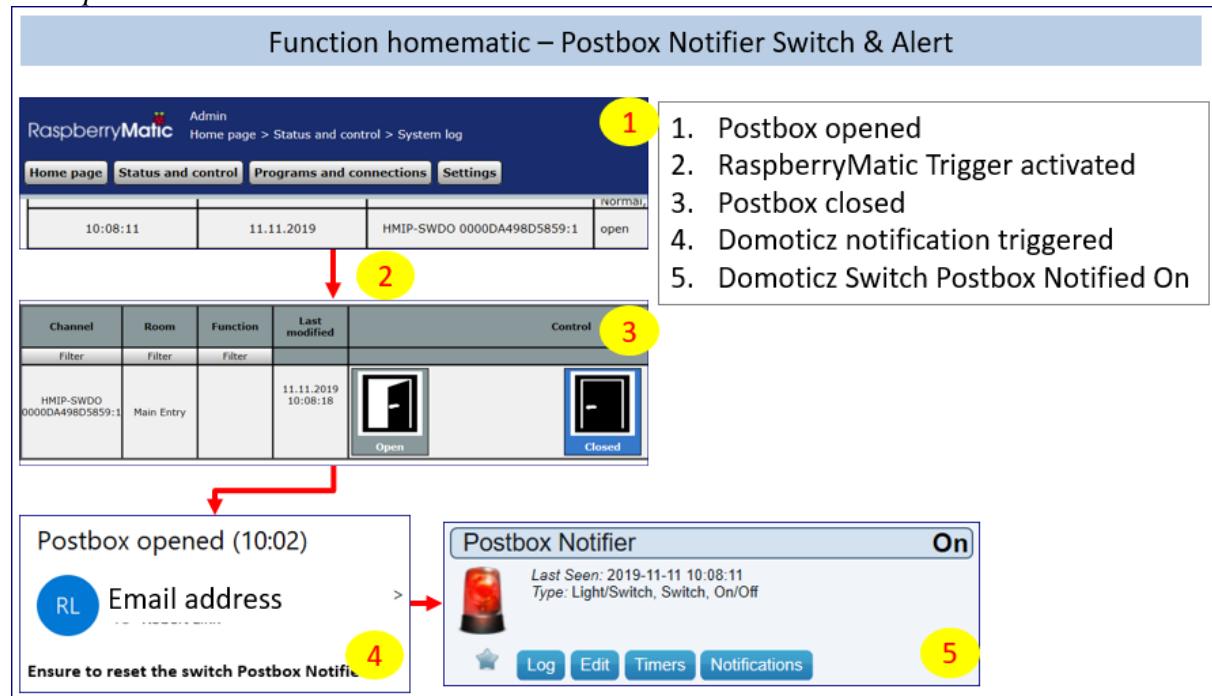
To avoid sending bulk emails within short timeframe, the notified time is checked against the previous notified time.

## Solution Alert + Switch

### Communication Flow

1. HMIP-SWDO detects opening the hatch of the postbox.
2. HMIP-SWDO triggers script which submits HTTP API Request to Domoticz system to set the state of the switch “Postbox Notifier” (idx: 189) to “On” and sends out an Alert Message
  - a. subject “Postbox opened (HH:MM)”
  - b. body “Ensure to reset the switch Postbox Notifier.”
3. The switch “Postbox Notifier” is set to “Off” manually.

#### Example



## RaspberryMatic CCU Configuration

### Step 1

The device HMIP-SWDO is added to the CCU following the Teach-in devices procedure (used entering the Key and SGTIN data).

The screenshot shows two main sections of the RaspberryMatic interface:

- (1) Add new device:** A screenshot of the "Device inbox" page where a new device, "HMIP-SWDO", has been added. The device details are: Type description: HMIP-SWDO, Picture: door icon, Description: Homematic IP Window / Door Contact - optical, Serial number: 0000DA4498D5859, Interface category: HmIP-RF, Transmission mode: Secured, Name: HMIP-SWDO, Function: 0000DA498D5859. Action buttons include Test, OK, Delete, Set, and Done.
- (2) Device Status and control:** A screenshot of the "Status and control > Devices" page showing the device "HMIP-SWDO" with ID 0000DA498D5859:1. It lists Channel, Room, Function, Last modified, and Control (Open/Closed) buttons.

### Step 2

Create a program to trigger sending the Alert Message Level 4 (RED) to Domoticz in case the device state changes to open.

Domoticz handles the Alert via dzVents Lua script event “alertmsg\_monitor.lua”.

```
IF HMIP-SWDO...:1 is OPEN THEN trigger WHEN updated SCRIPT
```

The screenshot shows the configuration of a dzVents rule:

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
SWDOS59	Postbox Alert	Channel status: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b> Device selection: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated OR DR				
<b>Activity: Then...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). Script: Postbox Alert string sFunction = "Postbox Alert"... immediately				
<b>Activity: Else...</b> <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				

## RaspberryMatic Script

The script makes use of a CUxD device to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz Alert Message device (IDX=55) with Alert Level 4 (nvalue) and the message (svalue):

```
wget -q -O - 'http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=IDX&switchcmd=CMD'
```

*Note*

The switch command “CMD” is case sensitive: On | Off.

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

## Script Code

```

! Function: Postbox Notifier, 20191110 by rwbl
string sFunction = "Postbox Notifier";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
! string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";

! A domoticz alert sensor is used
string nIdx = 189; ! Type: Light/Switch; SubType: On/Off
string sOn = "On";
string sText = "Postbox%20opened%20(" # sDate # "%20" # sTime # ")";

! Build the domoticz HTTP API Request url
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm'; ! Production
string sUrl =
sDomUrl#"?type=command#cAmp#param=switchlight#cAmp#"idx="#nIdx#cAmp#"switchcmd="#sOn#"';
WriteLine(sUrl);

! OPTION: Run the command without a return result
! res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);

! OPTION: Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: The script running on the CCU waits until the completion - ensure not to execute commands which
take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RET").State();
! WriteLine("VT="#sRes.VarType()#/##sRes); ! VT=4, {"status" : "OK","title" : " SwitchLight"}

! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
    dl.Variable(sFunction#"-Last update OK:#sDate#" "#sTime ")
}
else {
    dl.Variable(sFunction#"Last update ERROR:#sDate#" "#sTime")
};
! WriteLine("VT="#dl.VarType()); ! VT=9
! WriteLine(dl.Variable()); ! shows the last update string

```

## Domoticz Configuration

This solution makes use of the

- [Alert Message](#). No additional configuration required.
- Switch. Type: Light/Switch, SubType: Switch.

## Automation Script

Event name: postbox\_notifier

```
-- postbox_notifier.lua
local IDX_POSTBOXNOTIFIER = 189

return {
    -- Check which device(s) have a state change
    on = {
        devices = { IDX_POSTBOXNOTIFIER }
    },
    data = {
        -- set a flag if already notified
        pbnnotified = { initial = 0 }
    },
    -- Handle the switch if its state has changed to On
    execute = function(domoticz, device)
        -- Send email notification in case device is switched on
        -- Only the subject is used
        if (domoticz.devices(IDX_POSTBOXNOTIFIER).state == 'On') and (domoticz.data.pbnnotified == 0) then
            local subject = 'Postbox opened (' .. domoticz.helpers.isnowhhmm(domoticz) .. ')'
            local msg = 'Ensure to reset the switch Postbox Notifier.'
            domoticz.data.pbnnotified = 1
            domoticz.notify(subject, msg, domoticz.PRIORITY_HIGH)
        end
        if (domoticz.devices(IDX_POSTBOXNOTIFIER).state == 'Off') and (domoticz.data.pbnnotified == 1) then
            domoticz.data.pbnnotified = 0
        end
    end
}
```

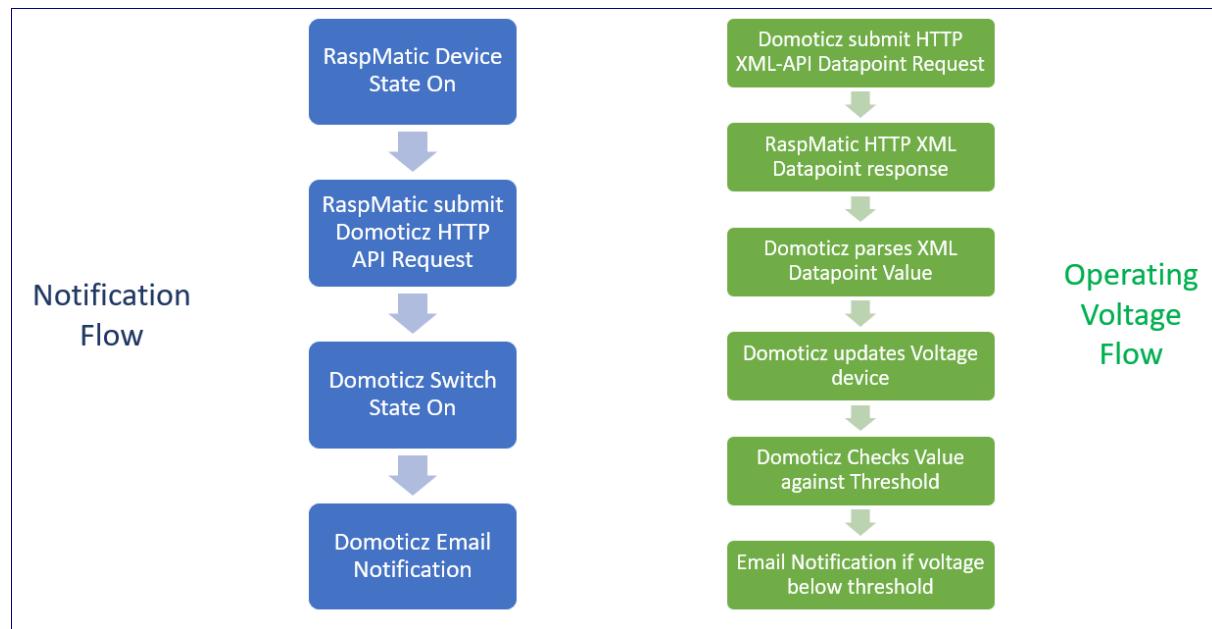
## Solution Alert + Switch + Voltage

This solution is in place. It is based upon the “Solution Alert + Switch” and enhanced with monitoring the Operating Voltage of the HomematicIP device SWDO.

### Communication Flow

There are two communication flows:

1. Notification – RaspMatic pushes the SWDO device state change to Domoticz.
2. Operating Voltage – Domoticz pulls the SWDO device Operating Voltage value.



### RaspberryMatic CCU Configuration

As “Solution Alert + Switch”.

### RaspberryMatic Script

As “Solution Alert + Switch”.

## Domoticz Configuration

### Domoticz State Device “Postbox Notifier State”

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
189	VirtualSensors	0001410D	1	Postbox Notifier State	Light/Switch	Switch	On

**Postbox Notifier State Off**

Last Seen: 2019-12-11 18:46:31  
Type: Light/Switch, Switch, On/Off

**Postbox Notifier State On**

Last Seen: 2019-12-11 18:52:49  
Type: Light/Switch, Switch, On/Off

**Email Notification**

Robert Linn Postbox opened (11:38)  
Postbox Notifier: Ensure to reset switch. <end>

Wed 11/12/2019 11:38 47 KB

**Domoticz Log Entry Switch On**

```
2019-12-11 11:38:25.383 Status: User: Admin initiated a switch command (189/Postbox Notifier State/On)
2019-12-11 11:38:25.578 Status: dZvents: Info: Handling events for: "Postbox Notifier State", value: "On"
2019-12-11 11:38:25.578 Status: dZvents: Info: ----- Start internal script: postbox_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189
2019-12-11 11:38:25.578 Status: dZvents: Info: Device Postbox Notifier State changed On = 0
2019-12-11 11:38:25.579 Status: dZvents: Info: POSTBOXNOTIFIER Postbox Notifier State notification sent: Postbox opened (11:38) (Postbox Notifier: Ensure to reset switch.)
2019-12-11 11:38:25.580 Status: dZvents: Info: ----- Finished postbox_notifier
2019-12-11 11:38:25.580 Status: EventSystem: Script event triggered: /home/pi/domoticz/dZvents/runtime/dZvents.lua
2019-12-11 11:38:25.586 Status: Notification: Postbox opened (11:38)
2019-12-11 11:38:26.648 Notification sent (email) => Success
```

**Domoticz Log Entry Switch Off**

```
2019-12-11 11:38:13.305 Status: User: Admin initiated a switch command (189/Postbox Notifier State/Off)
2019-12-11 11:38:13.409 Status: dZvents: Info: Handling events for: "Postbox Notifier State", value: "Off"
2019-12-11 11:38:13.410 Status: dZvents: Info: ----- Start internal script: postbox_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189
2019-12-11 11:38:13.410 Status: dZvents: Info: Device Postbox Notifier State changed Off = 0
2019-12-11 11:38:13.411 Status: dZvents: Info: ----- Finished postbox_notifier
```

### Domoticz Voltage Device “Postbox Notifier Voltage”

Create a user variable to set the threshold of the Operating Voltage of the homematicIP device SWDO. If the threshold is reached, than a notification is sent out by the Automation script “postbox\_notifier.lua”. The value of the threshold is the same as defined in RaspberryMatic Set device / channel parameter.

**Create Virtual Sensor**

Name: Postbox Notifier Voltage  
Sensor Type: Voltage

**Idx** **Hardware** **ID** **Unit** **Name** **Type** **SubType** **Data**

190	VirtualSensors	00082190	1	Postbox Notifier Voltage	General	Voltage	1.5 V
-----	----------------	----------	---	--------------------------	---------	---------	-------

**Postbox Notifier Voltage 1.5 V**

Last Seen: 2019-12-11 18:56:00  
Type: General, Voltage

**Log Edit Notifications**

## Domoticz User Variable “TH\_POSTBOXNOTIFIER\_VOLTAGE”

Create a user variable to set the threshold of the Operating Voltage of the homematicIP device SWDO.  
If the threshold is reached, than a notification is sent out by the Automation script “postbox\_notifier.lua”.  
The value of the threshold is the same as defined in RaspberryMatic Set device / channel parameter.

The screenshot shows two parts of the Domoticz interface. The top part is a table for managing user variables, showing one entry for 'TH\_POSTBOXNOTIFIER\_VOLTAGE' with a value of 1.0. The bottom part is the 'Set device / channel parameter' screen for a 'Postbox Notifier' device. In the 'Channel parameter' section, the 'Low. bat. threshold' is set to 1.00 V. The 'Parameter' table includes rows for 'Cyclically status message', 'Number of messages that are left out', 'Number of unchanged status messages that are left out', 'Low. bat. threshold' (highlighted in red), 'Lock inset via device button', 'Routing active', 'Event delay', 'Message in position open', and 'Message in position closed'.

## Domoticz Log

2019-12-11 18:24:00.564 Status: dzVents: Info: ----- Start internal script: postbox\_notifier:, trigger: every minute  
 2019-12-11 18:24:00.571 Status: dzVents: Info: ----- Finished postbox\_notifier

2019-12-11 18:24:00.942 Status: dzVents: Info: Handling httpResponse-events for: "RESPOSTBOXNOTIFIER"  
 2019-12-11 18:24:00.942 Status: dzVents: Info: ----- Start internal script: postbox\_notifier: HTTPResponse: "RESPOSTBOXNOTIFIER"  
 2019-12-11 18:24:00.942 Status: dzVents: Info: Postbox Notifier Voltage:Voltage=1.5  
 2019-12-11 18:24:00.959 Status: dzVents: Info: ----- Finished postbox\_notifier

2019-12-11 18:26:51.854 (VirtualSensors) Light/Switch (Postbox Notifier State)  
 2019-12-11 18:26:51.841 Status: User: Admin initiated a switch command (189/Postbox Notifier State/On)  
 2019-12-11 18:26:51.861 Status: Incoming connection from: 192.168.1.225  
 2019-12-11 18:26:52.061 Notification sent (http) => Success  
 2019-12-11 18:26:52.032 Status: dzVents: Info: Handling events for: "Postbox Notifier State", value: "On"  
 2019-12-11 18:26:52.032 Status: dzVents: Info: ----- Start internal script: postbox\_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189  
 2019-12-11 18:26:52.033 Status: dzVents: Info: Device Postbox Notifier State changed On = 0  
 2019-12-11 18:26:52.034 Status: dzVents: Info: POSTBOXNOTIFIER Postbox Notifier notification sent: Postbox opened (18:26) (Postbox Notifier: Ensure to reset switch.)  
 2019-12-11 18:26:52.034 Status: dzVents: Info: ----- Finished postbox\_notifier

2019-12-11 18:28:30.907 (VirtualSensors) Light/Switch (Postbox Notifier State)  
 2019-12-11 18:28:30.903 Status: User: Admin initiated a switch command (189/Postbox Notifier State/Off)  
 2019-12-11 18:28:31.009 Status: dzVents: Info: Handling events for: "Postbox Notifier State", value: "Off"  
 2019-12-11 18:28:31.009 Status: dzVents: Info: ----- Start internal script: postbox\_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189  
 2019-12-11 18:28:31.010 Status: dzVents: Info: Device Postbox Notifier State changed Off = 1  
 2019-12-11 18:28:31.011 Status: dzVents: Info: ----- Finished postbox\_notifier

For tests running automation script every minute

RaspMatic HTTP XML-API request operating voltage

Postbox opened. RaspMatic scripts sets Domoticz switch to On

Domoticz switch reset to off by user

## Domoticz Roomplan & Dashboard

The screenshot shows the Domoticz software interface. At the top, there's a navigation bar with 'Roomplan' selected. Below it, the 'Devices' section lists two entries: 'Postbox Notifier State' and 'Postbox Notifier Voltage'. The 'Dashboard' section displays real-time data for these devices: 'Postbox Notifier State' is set to 'On' and 'Postbox Notifier Voltage' is at '1.5 V'. The bottom part of the dashboard shows a timestamp '2019-12-11 18:40:34' and some system status indicators.

## Automation Script

Event name: postbox\_notifier

```
-- postbox_notifier.lua
-- Domoticz Idx
local IDX_POSTBOXNOTIFIER_STATE = 189
local IDX_POSTBOXNOTIFIER_VOLTAGE = 190
-- Domoticz Idx of the user variable for the operating voltage threshold (as float)
local TH_POSTBOXNOTIFIER_VOLTAGE = 16
-- Homematic device datapoint id (device HMIP-SWDO)
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
-- datapoint
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux
local DATAPOINT2543 = JSON:decode('{"name":"Postbox Notifier Voltage","id":2543,"xpath":"/datapoint[@ise_id=2543]/@value","response":"RESPONSEPOSTBOXNOTIFIER"}');

return {
    -- Combined rules defined
    on = {
        -- Check operating voltage datapoint SWDO HomematicIp device submit http xml-pi request
        timer = { 'every 30 minutes' },
        -- Check state postbox notifier switch has changed to on as triggered by raspmatic script
        devices = { IDX_POSTBOXNOTIFIER_STATE },
        -- Handle the response of the http xml-api url request
        httpResponses = { DATAPOINT2543.response, }
    },
    data = {
        -- set a flag if already notified
        pbnnotified = { initial = 0 }
    },
    -- Handle rules for the item isDevice, isTimer, isHTTPResponse
    execute = function(domoticz, item)
        -- Handle state change
        if (item.isDevice) then
            -- Send email notification in case device is switched on
            -- Only the subject is used
            if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state == 'On') and
                (domoticz.data.pbnnotified == 0) then
                local subject = 'Postbox opened (' .. domoticz.helpers.isnowhhmm(domoticz) .. ')'
                local message = 'Postbox Notifier: Ensure to reset state.'
                domoticz.data.pbnnotified = 1
                -- notify via email
                domoticz.notify(subject, message, domoticz.PRIORITY_HIGH)
                -- update alert level orange (3) = does not send an email if user var TH_ALERTTOEMAIL = 4
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, message)
            end
        end
    end
}
```

```
-- Reset the notify flag is switch is manually switched Off
if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state == 'Off') and
    (domoticz.data.pbnnotified == 1) then
    domoticz.data.pbnnotified = 0
end
end

-- check if the item is a device, then request datapoint information
if (item.isTimer) then
    domoticz.openURL({url=URL_RASPMATIC..DATAPPOINT2543.id,method='GET',callback=
DATAPPOINT2543.response,})
end

-- check if the item is a httpresponse from the openurl callback
if (item.isHTTPResponse) then
    if (item.statusCode == 200) then
        -- domoticz.log(item.data);
        -- Select the callback
        if (item.callback == DATAPPOINT2543.response) then
            -- Get the operating voltage
            local voltage = tonumber(domoticz_applyXPath(item.data, DATAPPOINT2543.xpath))
            domoticz.log(DATAPPOINT2543.name .. ':' .. voltage)
            -- Update the domoticz device
            domoticz.devices(IDX_POSTBOXNOTIFIER_VOLTAGE).updateVoltage(voltage)
            -- Check if below threshold
            if voltage < domoticz.variables(IDX_TH_POSTBOXNOTIFIER_VOLTAGE).value then
                local message= DATAPPOINT2543.name .. ':LOW ' .. voltage .. ' '
domoticz.helpers.isnowhhmm(domoticz)
                domoticz.log(message)
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
            end
        end
    else
        domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
    end
end
end
}
```

# Radiator Thermostat (HmIP-eTRV)

The HomematicIP Radiator Thermostats HmIP-eTRV-B & HmIP-eTRV-2 are installed at several radiators in the house.

The HomematicIP devices are connection partner Radiator Thermostats (manual operation, transmitter) and connected to the [RaspberryMatic](#) system ([Homematic](#) compatible).

## Purpose

- To set the temperature setpoint of a radiator thermostat.
- To measure, in regular intervals, the room temperature.
- To check, in regular intervals, the low battery state of the device.

Various solutions have been developed to explore what's possible.

The first solution is currently in place for several thermostats.

1. Domoticz Thermostat SetPoint device  
(USED)
2. Domoticz Selector Switch Device connected to a Thermostat SetPoint device  
(NOT USED)
3. Domoticz Python Plugin for a Thermostat with various devices  
(NOT USED)

## Solution Thermostat SetPoint Device

This solution uses for each Homematic IP Thermostat device a dedicated Domoticz Thermostat SetPoint device.

There are two actions:

### 1. Update Domoticz Thermostat Setpoint triggered by RaspberryMatic

To silent (to avoid endless communication between RaspberryMatic and Domoticz) set the Domoticz Thermostat Setpoint triggered by either changing the RaspberryMatic WebUI or the knob/buttons on the Homematic IP Thermostat device.

Communication RaspberryMatic to Domoticz, via the Domoticz HTTP API triggering a Domoticz Custom Event with JSON Data. This is handled by a RaspberryMatic Script.

```
http://domoticz-ip:<port>/json.htm?type=command&param=customevent&event=raspmatic_thermostat_sync&data={"id":NNNN,"idx":NNN,"setpoint":NN.N,"temperature":NN.N,"sync":1}
```

RaspberryMatic Script	Domoticz Custom Event
raspmatic_thermostat_sync.script	raspmatic_thermostat_sync.dzvents

### 2. Set the RaspberryMatic Thermostat Setpoint triggered by Domoticz

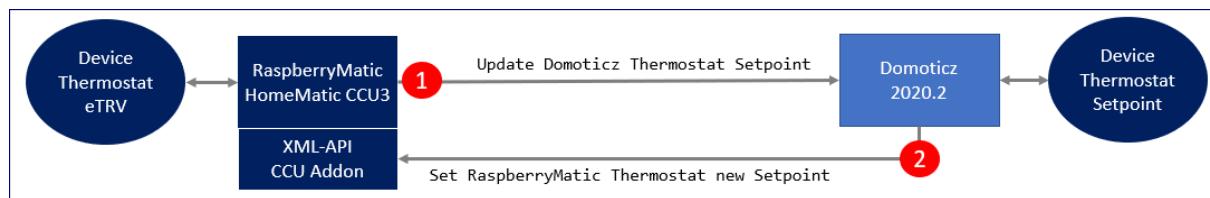
To change the setpoint of the Homematic IP Thermostat Setpoint via a Domoticz Thermostat Setpoint device.

Communication Domoticz to RaspberryMatic via the RaspberryMatic XML-API CCU Addon. This is handled by a Domoticz Automation Event dzVents Lua.

```
http://raspmatic-ip/addons/xmlapi/statechange.cgi?ise_id=NNNN&new_value=NN.N
```

RaspberryMatic Script	Domoticz Event
n/a	raspmatic_thermostat_setsetpoint.dzvents

### Communication Flow



## Domoticz Configuration

For each of the Homematic IP thermostats, a Domoticz Virtual Sensor Type Thermostat, SubType SetPoint is defined.

The Domoticz thermostat device description holds, in JSON format, the RaspberryMatic datapoint id (see [RaspberryMatic Datapoints](#)):

```
{"iseid":NNN}
```

The id is used in the Domoticz Automation Event “raspmatic\_thermostat\_setsetpoint.dzvents” for the HTTP XML-API request to change the new value of the setpoint in Raspberry Matic.

**Domoticz Devices – just one of many devices listed**

Idx	Hardware	ID	Unit	Name	Type	SubType	
341	VirtualSensors	00141A5	1	Bad	Thermostat	SetPoint	21.0

**Widget Idx=341**

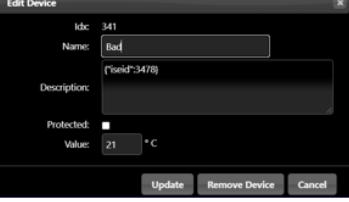


Widget Edit Device with RaspberryMatic device datapoint defined as JSON

**Dashboard Mobile with setpoint change dialog**





## RaspberryMatic Datapoints

The list of RaspberryMatic defined thermostat devices is obtained using the XML-API addon:

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

Only the datapoint SET\_POINT\_TEMPERATURE is shown as used to control the thermostat setpoint via HTTP XML-API GET or SET requests.

**GET** the thermostat SET\_POINT\_TEMPERATURE:

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=DATAPOINT_ISE_ID
```

**SET** the thermostat SET\_POINT\_TEMPERATURE:

```
http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=DATAPOINT_ISE_ID
```

Example Thermostat datapoint SET\_POINT\_TEMPERATURE.  
The “Thermostat Bad” is used as an example, with ise\_id: 3478.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="3435" name="Thermostat Bad" config_pending="false" unreach="false">
<datapoint ise_id="3478" name="HmIP-RF.00201A99D2097:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144602" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="3489" name="Thermostat Dusche" config_pending="false" unreach="false">
<datapoint ise_id="3532" name="HmIP-RF.00201A49952CB8:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588139539" valueunit="°C" valuetype="4" value="6.000000"/>
</device>
<device ise_id="3637" name="Thermostat Esszimmer" config_pending="false" unreach="false">
<datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144916" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="3691" name="Thermostat Flur" config_pending="false" unreach="false">
<datapoint ise_id="3734" name="HmIP-RF.000A1A49A0D8A5:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144818" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
<device ise_id="1541" name="Thermostat MakeLab" config_pending="false" unreach="false">
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144325" valueunit="°C" valuetype="4" value="20.000000"/>
</device>
<device ise_id="3543" name="Thermostat Wohnzimmer-1" config_pending="false" unreach="false">
<datapoint ise_id="3586" name="HmIP-RF.00201A49952CB1:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144907" valueunit="°C" valuetype="4" value="12.000000"/>
</device>
<device ise_id="3375" name="Thermostat Wohnzimmer-2" config_pending="false" unreach="false">
<datapoint ise_id="3418" name="HmIP-RF.00201A499D76F8:1 SET_POINT_TEMPERATURE" type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144908" valueunit="°C" valuetype="4" value="21.000000"/>
</device>
</stateList>
```

## Domoticz Thermostat Device Define Datapoint

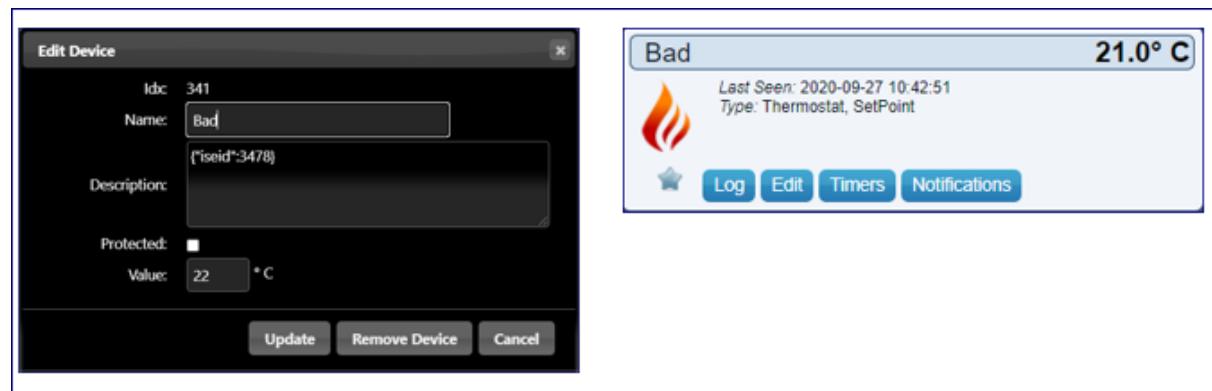
The datapoints are defined in the Domoticz Thermostat Description as JSON.

For the dzVents Lua automation script, the RaspberryMatic Thermostats datapoint ise\_id for the type SET\_POINT\_TEMPERATURE is required.

This is defined for each Domoticz Thermostat Device in the device description as JSON string:

```
{"iseid":NNNN}
```

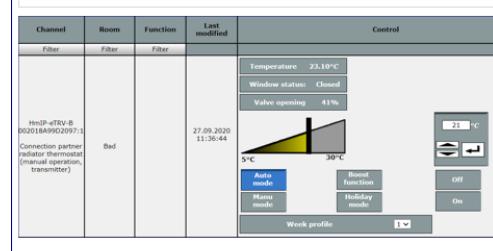
The datapoint ise\_id is taken from the XML-API statelist request (see previous). This is an example for the device “Thermostat Bad” with datapoint 3478.



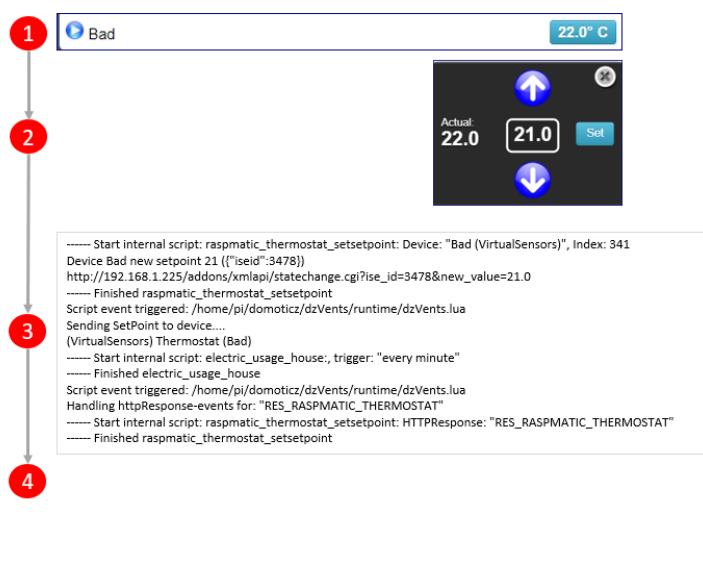
## Action: Domoticz Thermostat Setpoint Change

The flow of actions changing the setpoint of a Domoticz thermostat device via the Domoticz Dashboard.

1. Click the temperature button [22.0 °C] of the Thermostat device from the Domoticz Dashboard
2. Set the new setpoint to 21.0 and click button [Set] to change the setpoint
3. Automation Event dzVents Lua “raspmatic\_thermostat\_setsetpoint” is triggered to submit the new setpoint to RaspberryMatic via HTTP XML-API request datapoint SET\_POINT\_TEMPERATURE
4. RaspberryMatic to update the thermostat device setpoint (Homematic WebUI)



### Domoticz Dashboard Mobile



## Automation Script

```
-- raspmatic_thermostat_setsetpoint.dzvents
-- Set new setpoint Homematic IP thermostat triggered by a Domoticz thermostat device setpoint change.
-- The RaspberryMatic device datapoint is deviced as JSON string in the Domoticz device description

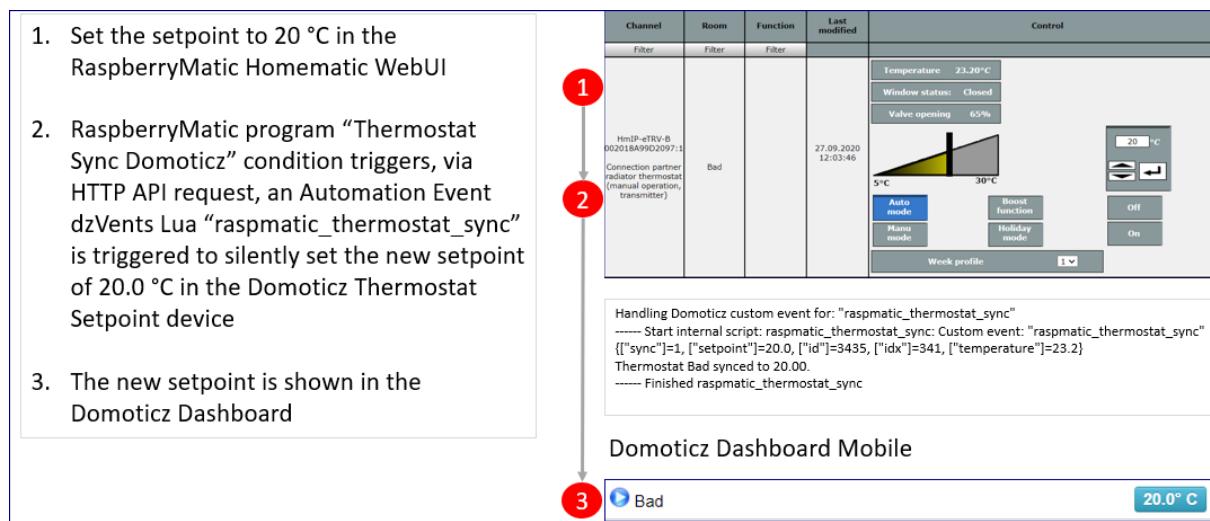
-- url http xml-api request to set a new value
local URL_RASPMATIC = 'http://raspmatic-
ip/addons/xmlapi/statechange.cgi?ise_id=#ISEID#&new_value=#NEWVALUE#';
-- define the unique (across all dzVents) callback
local RES_RASPMATIC = "RES_RASPMATIC_THERMOSTAT";
-- list of the idx of all thermostats (see GUI > Setup > Devices)
local idxdevicestable = {336,337,338,339,340,341,342};

return {
  on = {
    devices = idxdevicestable,
    httpResponses = { RES_RASPMATIC }
  },
  execute = function(domoticz, item)
    -- Device changes a setpoint
    if (item.isDevice) then
      local device = domoticz.devices(item.id)
      local setpoint = device.setPoint;
      domoticz.log(string.format('Device %s new setpoint %.0f (%s)', device.name, device.setPoint,
device.description), domoticz.LOG_INFO)
      -- Read and convert the JSON description to a table
      local jsondesc = domoticz.utils.fromJSON(device.description)
      -- Get the raspmatic datapoint d for the device setup
      local iseid = jsondesc.iseid
      -- Build the url to set the new setpoint in raspmatic
      -- Replace the placeholders with the id and newvalue
      local url = URL_RASPMATIC:gsub("#ISEID#", iseid):gsub("#NEWVALUE#",setpoint)
      domoticz.log(url)
      domoticz.openURL({ url = url, method = 'POST', callback = RES_RASPMATIC })
    end
    --
    if (item.isHTTPResponse) then
      if (item.ok and item.callback == RES_RASPMATIC ) then
        -- domoticz.log(item.data, domoticz.LOG_INFO)
      else
        domoticz.log(string.format('Problem handling the request: %s',url), domoticz.LOG_ERROR)
        domoticz.log(item.data, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## Action: RaspberryMatic Thermostat Setpoint Change

The flow of actions RaspberryMatic trigger Thermostat Setpoint Change via Homematic WebUI.

The flow of actions are also triggered in case the thermostat is manually changed via knob or +/- buttons (depending thermostat model). The change is also reflected in the Homematic WebUI.



## RaspberryMatic Program updates Domoticz Thermostat Setpoint

Define condition for all thermostat devices to trigger a script to update the Domoticz thermostat setpoint silently:

if the thermostat setpoint temperature is updated ("trigger when updated") having a value of more than 4.5°C

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
Thermostat Sync Domoticz	Sync the thermostat setpoint with the corresponding Domoticz device.	Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Setpoint temperature within value range / with value more than 4.50° C trigger when updated	Script: immediately run	<input type="checkbox"/> intrinsic

Conditions: If...

Activity: Then...  Stop all current delays before performing the activity (e.g. retriggering).

Script: Function: Quick Access Mobile ! If the setpoint of a therm...

Activity: Else...  Stop all current delays before performing the activity (e.g. retriggering).

## RaspberryMatic Script updates Domoticz Thermostat Setpoint

If the previous defined conditions apply, then the activity is to execute the script below which submits a HTTP API request to Domoticz, handled by a Custom Event. The Custom Event parameter “data” contains in JSON format the required fields to update the setpoint of the thermostat device in Domoticz.

```

! If the setpoint of a thermostat gets updated, update the corresponding domoticz thermostat silent (no
active setpoint change).
! Via cuxd an http api request for a customevent is submitted to domoticz.
! Custom event data parameter must be in json format.
! Note: Use http://ccu-ip/config/xmlapi/statelist.cgi to get device iseid and datapoints.
! The thermostat trigger is value > 4.5 when updated
string cAmp = "&";
! Domoticz system url base for the http api request
string urlBase = "'http://domoticz-ip:port/json.htm'; ! Production
! string urlBase = "'http://domoticz-ip:port/json.htm"; ! Development
string customEvent = "raspmatic_thermostat_sync";
string deviceID = "UNKNOWN";      ! 1541
string setPoint = "";            ! 20.5
string actTemperature = "";     ! 20.3
string deviceIdx = "UNKNOWN";   ! 336
string syncState = "1";          ! must be 1 for silent update in domoticz
! Get the object datapoint of the thermostat which setpoint is updated
object objDatapoint = dom.GetObject ("$src$");
if (objDatapoint) {
    if (objDatapoint.Value()) {
        object objChannel = dom.GetObject (objDatapoint.Channel());
        deviceID = objChannel.Device();
        setPoint = objChannel.DPByHssDP("SET_POINT_TEMPERATURE").Value();
        actTemperature = objChannel.DPByHssDP("ACTUAL_TEMPERATURE").Value();
    }
}
! Map the raspmatic thermostat datapoint to the domoticz idx
if (deviceID == "3489") { deviceIdx = 342; }  ! Dusche
if (deviceID == "3435") { deviceIdx = 341; }  ! Bad
if (deviceID == "3691") { deviceIdx = 340; }  ! Flur
if (deviceID == "3375") { deviceIdx = 339; }  ! Wohnzimmer-2
if (deviceID == "3543") { deviceIdx = 338; }  ! Wohnzimmer-1
if (deviceID == "3637") { deviceIdx = 337; }  ! Esszimmer
if (deviceID == "1541") { deviceIdx = 336; }  ! MakeLab
! Custom event data parameter must be in json format
string data =
'{"id": "#deviceID#", "idx": "#deviceIdx#", "setpoint": "#setPoint#", "temperature": "#actTemperature#", "sync": "#syncState#"}';
! Build the Domoticz http rest request url to update the device using customevent
string urlRequest =
urlBase#"?type=command#&param=customevent#&event="#customEvent#&data="#data#"";
WriteLine(urlRequest);
! Run the command without a return result
cmdRes = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#urlRequest);

```

## Domoticz dzVents Lua Automation Script sync new setpoint RaspberryMatic

Domoticz dzVents Lua Automation Script sync new setpoint RaspberryMatic with the Domoticz thermostat device.

```
-- raspmatic_thermostat_sync.dzvents
-- Sync the domoticz thermostat setpoint with the raspberrymatic thermostat setpoint.
-- The event is triggered by raspmatic setpoint change program.

local CUSTOMEVENT_NAME = "raspmatic_thermostat_sync"

return {
  on = {
    customEvents = {
      CUSTOMEVENT_NAME
    }
  },
  data = {},
  logging = {},
  execute = function(domoticz, item)
    if (item.isCustomEvent) then
      -- {[{"setpoint":23.0, "temperature":22.1, "sync":1, "id":3489, "idx":342}]
      domoticz.log(item.data);
      if item.data.sync == 1 then
        domoticz.devices(item.data.idx).updateSetPoint(item.data.setpoint).silent()
        domoticz.log(string.format("Thermostat %s synced to %.2f.", 
domoticz.devices(item.data.idx).name, item.data.setpoint))
        end
      else
        -- second parameter can be anything, number, string, boolean or table
        -- domoticz.emitEvent('MyEvent', 'Some data')
      end
    end
  }
}
```

## Solution Selector Switch & Thermostat

This is a simple solution making use of a selector switch to select from the list of thermostats. The setpoint for the selected thermostat is set by using a thermostat device.

Communication between Domoticz and RaspberryMatic using the RaspberryMatic XML-API CCU Addon.

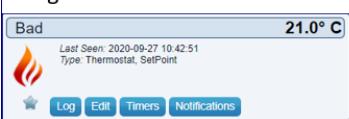
### Domoticz Configuration

**Domoticz Devices – just one of many devices listed**

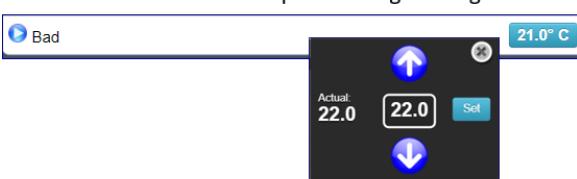
Idx	Hardware	ID	Unit	Name	Type	SubType
341	VirtualSensors	00141A5	1	Bad	Thermostat	SetPoint

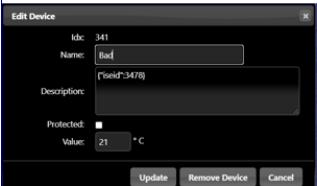
**Widget Idx=341**

**Dashboard Mobile with setpoint change dialog**

**Widget Edit Device with RaspberryMatic device datapoint**



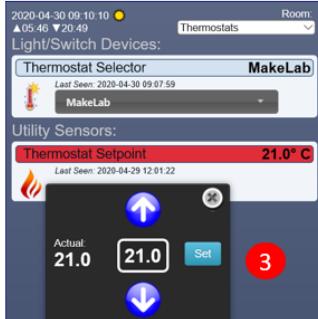
```
Domoticz Log Automation Events dyVents Lua
2020-09-27... Handling events for: "Bad", value: "22.00"
2020-09-27... ----- Start internal script: raspmatic_thermostat_setsetpoint: Device: "Bad (VirtualSensors)", Index: 341
2020-09-27... ----- Device Bad never setpoint 22 ("iseid":3478)
2020-09-27... http://192.168.1.225/addons/xmlapi/statechange.cgi?se_id=3478&new_value=22.0
2020-09-27... ----- Finished raspmatic_thermostat_setsetpoint

2020-09-27... Handling httpResponse-events for: "RES_RASPMATIC_THERMOSTAT"
2020-09-27... ----- Start internal script: raspmatic_thermostat_setsetpoint: HTTPResponse: "RES_RASPMATIC_THERMOSTAT"
2020-09-27... ----- Finished raspmatic_thermostat_setsetpoint
2020-09-27... Handling Domoticz custom event for: "raspmatic_thermostat_sync"
2020-09-27... ----- Start internal script: raspmatic_thermostat_sync: Custom event: "raspmatic_thermostat_sync"
2020-09-27... [{"setpoint":22.0, ["idx":341, ["temperature":23.1, ["id":5435, ["sync":1]}]
2020-09-27... Thermostat Bad synced to 22.0
2020-09-27... ----- Finished raspmatic_thermostat_sync
```

### Select Thermostat & Setpoint change

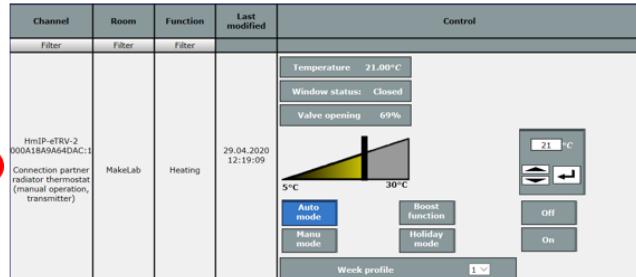
1. Select the thermostat:  
Domoticz device: "Thermostat Selector"
2. Get the RaspberryMatic selected thermostat device value for the datapoint SET\_POINT\_TEMPERATURE & update the Domoticz device: "Thermostat Setpoint" (via HTTP XML-API request)
3. Change the setpoint:  
Domoticz device: "Thermostat Setpoint"
4. Update RaspberryMatic thermostat device setpoint (via HTTP XML-API request)

1



4

1



2



## Thermostat Devices

Get all devices defined in RaspberryMatic CCU by sending an HTTP XML-API request to the CCU with parameter the script “statelist.cgi”.

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

From the HTTP XML response, only the datapoint type SET\_POINT\_TEMPERATURE is required as used to control the thermostat, gain, via HTTP XML-API requests with script as parameter.

(extract from the statelist request)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="3435" name="Thermostat Bad" config_pending="false" unreach="false">
<datapoint ise_id="3478" name="HmIP-RF.002018A99D2097:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144602" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="3489" name="Thermostat Dusche" config_pending="false" unreach="false">
<datapoint ise_id="3532" name="HmIP-RF.00201A49952CB8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588139539" valueunit="°C" valuetype="4"
value="6.00000"/>
</device>
<device ise_id="3637" name="Thermostat Esszimmer" config_pending="false" unreach="false">
<datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144916" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="3691" name="Thermostat Flur" config_pending="false" unreach="false">
<datapoint ise_id="3734" name="HmIP-RF.000A1A49A0D8A5:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144818" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
<device ise_id="1541" name="Thermostat MakeLab" config_pending="false" unreach="false">
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144325" valueunit="°C" valuetype="4"
value="20.00000"/>
</device>
<device ise_id="3543" name="Thermostat Wohnzimmer-1" config_pending="false" unreach="false">
<datapoint ise_id="3586" name="HmIP-RF.00201A49952CB1:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144907" valueunit="°C" valuetype="4"
value="12.00000"/>
</device>
<device ise_id="3375" name="Thermostat Wohnzimmer-2" config_pending="false" unreach="false">
<datapoint ise_id="3418" name="HmIP-RF.00201A499D76F8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144908" valueunit="°C" valuetype="4"
value="21.00000"/>
</device>
</stateList>
```

### **GET the thermostat SET\_POINT\_TEMPERATURE**

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=DATAPOINT_ISE_ID
```

### **SET the thermostat SET\_POINT\_TEMPERATURE**

```
http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=DATAPOINT_ISE_ID
```

The datapoints are defined in a dzVents Lua Automation Script.

## Automation Script

For the dzVents Lua automation script, the RaspberryMatic Thermostat datapoint ise\_id for the type SET\_POINT\_TEMPERATURE, is required.

The datapoints are used to get or set the setpoint.

In dzVents an array is defined according to the thermostats defined in the selector switch:

```
[level]=ise_id SET_POINT_TEMPERATURE
```

```
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418 , [40]=3680, [50]=3478, [60]=3734, [70]=3532};
```

Name	Ise_id	Level	Level Name
Thermostat MakeLab	1584	10	MakeLab
Thermostat Wohnzimmer-1	3586	20	Wohn-1
Thermostat Wohnzimmer-2	3418	30	Wohn-2
Thermostat Esszimmer	3680	40	Essen
Thermostat Bad	3478	50	Bad
Thermostat Flur	3734	60	Flur
Thermostat Dusche	3532	70	Dusche

**IMPORTANT:** if the level order is changed in the device widget, the array must be changed as well!

Event name: thermostat\_control\_selector\_switch.dzvents  
(below code is stripped)

```
local IDX_THERMOSTAT_SELECTOR = 114, IDX_THERMOSTAT_SETPOINT = 115;
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418 , [40]=3680, [50]=3478, [60]=3734, [70]=3532};
local datapointSelected;
local URL_RASPMATIC_STATECHANGE = 'http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=',
URL_RASPMATIC_STATE = 'http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=';
local RES_RASPMATIC_SETSETPOINT = 'res_raspmatic_setsetpoint', RES_RASPMATIC_GETSETPOINT =
'res_raspmatic_getsetpoint';
local DECIMALS = 1;
function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end
return {
    on = {devices = {IDX_THERMOSTAT_SELECTOR, IDX_THERMOSTAT_SETPOINT},
        httpResponses = {RES_RASPMATIC_SETSETPOINT,RES_RASPMATIC_GETSETPOINT,}},
        data = {updateraspmatic = { initial = 1 }},
        execute = function(domoticz, item)
            if (item.isDevice) then
                datapointSelected = ID_THERMOSTAT_DATAPOINTS[domoticz.devices( IDX_THERMOSTAT_SELECTOR
).level];
                if (item.idx == IDX_THERMOSTAT_SELECTOR) then
                    domoticz.data.updateraspmatic = 0;
                    local urlrm = string.format("%s%d", URL_RASPMATIC_STATE, datapointSelected);
                    domoticz.openURL({url = urlrm, method = 'POST', callback = RES_RASPMATIC_GETSETPOINT});
                end
                if (item.idx == IDX_THERMOSTAT_SETPOINT) then
                    if (domoticz.data.updateraspmatic == 1) then
                        local newsetpoint = round(item.setPoint,DECIMALS);
                    local urlrm = string.format("%s%d&new_value=%2f", URL_RASPMATIC_STATECHANGE, datapointSelected,
newsetpoint);
                    domoticz.openURL({url = urlrm, method = 'POST', callback = RES_RASPMATIC_SETSETPOINT});
                    end
                    if (domoticz.data.updateraspmatic == 0) then domoticz.data.updateraspmatic = 1; end
                end
            end
        end
}
```

```
if (item.isHTTPResponse) then
    if (item.statusCode == 200) then
        if (item.callback == RES_RASPMATIC_GETSETPOINT) then
            datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level];
            local setpoint = domoticz_applyXPath(item.data,'//datapoint[@ise_id="" ..
datapointSelected .. '"]/@value');
            setpoint = round(setpoint, DECIMALS);
            local urldom = string.format("http://domoticz-
ip:8080/json.htm?type=command&param=setsetpoint&idx=%d&setpoint=%2f",IDX_THERMOSTAT_SETPOINT,setpoint)
;
            domoticz.openURL({url = urldom, method = 'POST'});
        end
        if (item.callback == RES_RASPMATIC_SETSETPOINT) then datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level]; end
        else
            domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
end
}
```

## Solution Plugin

- Various Domoticz device timers are put in place to control the temperature depending the radiator location.
- If an open window is detected by the thermostat, the setpoint is decreased to 12 °C until the window is closed, then the setpoint is back to previous value.
- Each device has a battery which is regularly checked and alert given if below threshold 2.2V (defined in the Homematic WebUI for each device).
- Just as an experiment: a [Node-RED Dashboard](#).

A Domoticz Python Plugin has been created, called **HomematicIP Radiator Thermostat (HmIP-eTRV)**.

## Installation

These steps apply for every Radiator Thermostat Device:

**Homematic WebUI (RaspberryMatic):**

Add the thermostat device to RaspberryMatic (Teach-in)

Download the HmIP-eTRV plugin repository from [here](#)

On the Domoticz system, create folder: /home/pi/domoticz/plugins/hmip-etrv

Copy the file plugin.py to the folder: /home/pi/domoticz/plugins/hmip-etrv

Restart Domoticz either via command line or GUI:

- sudo service domoticz.sh restart
- GUI > Setup > More Options > Restart System

The plugin requires the parameters:

- CCU IP Address
- Device id of the thermostat device HmIP-eTRV-B or HmIP-eTRV-2
- Datapoint ids for the selected device:  
SET\_POINT\_TEMPERATURE, ACTUAL\_TEMPERATURE, LOW\_BAT, LEVEL  
as comma separated list in this order

*Note*

Obtain the device & datapoint id's by requesting the device state list from the CCU via

URL: <http://ccu-ip/addons/xmlapi/statelist.cgi>

Get the devices state list: <http://ccu-ip-address/addons/xmlapi/statelist.cgi> with XML tree result.

1. Select Device HMIP-eTRV-B with name "Thermostat Dining Room"
2. Get the Device ID **3637**
3. Get the Datapoints IDs for the properties:  
SET\_POINT\_TEMPERATURE=3680,ACTUAL\_TEMPERATURE=3663,LOW\_BAT=3645,LEVEL=3672

The ids are used to

1. get the actual temperature, battery low voltage status & level
2. set the thermostat setpoint.

```
<device config_pending="False" unreach="False" ise_id="3637" name="Thermostat Dining Room">
  <channel ise_id="3638" name="Thermostat Dining Room" operate="true" visible="true" index="0">
    <datapoint ise_id="3638" name="HmIP-RF-0000A1A49A0D0878:0.DUTY_CYCLE" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="0" type="DUTY_CYCLE"/>
    <datapoint ise_id="3643" name="HmIP-RF-0000A1A49A0D0878:0.DUTY_CYCLE_OPERATIONS" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="0" type="DUTY_CYCLE"/>
    <datapoint ise_id="3645" name="HmIP-RF-0000A1A49A0D0878:0.LOW_BAT" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="0" type="LOW_BAT"/>
    <datapoint ise_id="3646" name="HmIP-RF-0000A1A49A0D0878:0.OPERATING_VOLTAGE" operations="5" timestamp="1577009479" valueunit="" valuetype="4" value="3.000000" type="OPERATING_VOLTAGE"/>
    <datapoint ise_id="3651" name="HmIP-RF-0000A1A49A0D0878:0.OPERATING_VOLTAGE_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
    <datapoint ise_id="3652" name="HmIP-RF-0000A1A49A0D0878:0.RSSI_PEER" operations="5" timestamp="1576990804" valueunit="" valuetype="0" value="200" type="RSSI_PEER"/>
    <datapoint ise_id="3653" name="HmIP-RF-0000A1A49A0D0878:0.UNREACH" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="0" type="UNREACH"/>
    <datapoint ise_id="3657" name="HmIP-RF-0000A1A49A0D0878:0.UPDATE_PENDING" operations="5" timestamp="15769940478" valueunit="" valuetype="2" value="0" type="UPDATE_PENDING"/>
  </channel>
  <channel ise_id="3661" name="HmIP-eTRV-2 0000A1A49A0D0878:1" operate="true" visible="true" index="1">
    <datapoint ise_id="3662" name="HmIP-RF-0000A1A49A0D0878:1.ACTIVE_PROFILE" operations="7" timestamp="1577009479" valueunit="" valuetype="16" value="1" type="ACTIVE_PROFILE"/>
    <datapoint ise_id="3663" name="HmIP-RF-0000A1A49A0D0878:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1577009479" valueunit="" valuetype="4" value="22.700000" type="ACTUAL_TEMPERATURE"/>
    <datapoint ise_id="3664" name="HmIP-RF-0000A1A49A0D0878:1.ACTUAL_TEMPERATURE_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
    <datapoint ise_id="3665" name="HmIP-RF-0000A1A49A0D0878:1.BOOT_TIME" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="BOOT_TIME"/>
    <datapoint ise_id="3666" name="HmIP-RF-0000A1A49A0D0878:1.BOOST_TIME" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="BOOST_TIME"/>
    <datapoint ise_id="3667" name="HmIP-RF-0000A1A49A0D0878:1.CONTROL_DIFFERENTIAL_TEMPERATURE" operations="2" timestamp="0" valueunit="" valuetype="4" value="4" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
    <datapoint ise_id="3668" name="HmIP-RF-0000A1A49A0D0878:1.CONTROL_MODE" operations="2" timestamp="0" valueunit="" valuetype="16" value="0" type="CONTROL_MODE"/>
    <datapoint ise_id="3669" name="HmIP-RF-0000A1A49A0D0878:1.DURATION_UNIT" operations="5" timestamp="0" valueunit="" valuetype="16" value="0" type="DURATION_UNIT"/>
    <datapoint ise_id="3670" name="HmIP-RF-0000A1A49A0D0878:1.FROST_PROTECTION" operations="5" timestamp="0" valueunit="" valuetype="16" value="0" type="FROST_PROTECTION"/>
    <datapoint ise_id="3671" name="HmIP-RF-0000A1A49A0D0878:1.LEVEL_OPERATIONS" operations="7" timestamp="1577009479" valueunit="" valuetype="4" value="0.120000" type="LEVEL"/>
    <datapoint ise_id="3672" name="HmIP-RF-0000A1A49A0D0878:1.LEVEL_OPERATIONS" operations="7" timestamp="1577009479" valueunit="" valuetype="4" value="0.120000" type="LEVEL"/>
    <datapoint ise_id="3673" name="HmIP-RF-0000A1A49A0D0878:1.LEVEL_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="LEVEL_STATUS"/>
    <datapoint ise_id="3674" name="HmIP-RF-0000A1A49A0D0878:1.PARTY_MODE" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="PARTY_MODE"/>
    <datapoint ise_id="3675" name="HmIP-RF-0000A1A49A0D0878:1.PARTY_SET_POINT_TEMPERATURE" operations="5" timestamp="0" valueunit="" valuetype="4" value="0.000000" type="PARTY_SET_POINT_TEMPERATURE"/>
    <datapoint ise_id="3676" name="HmIP-RF-0000A1A49A0D0878:1.PARTY_TIME_END" operations="7" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_END"/>
    <datapoint ise_id="3677" name="HmIP-RF-0000A1A49A0D0878:1.PARTY_TIME_START" operations="7" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_START"/>
    <datapoint ise_id="3678" name="HmIP-RF-0000A1A49A0D0878:1.QUICK_VETO_TIME" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="QUICK_VETO_TIME"/>
    <datapoint ise_id="3679" name="HmIP-RF-0000A1A49A0D0878:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1577009479" valueunit="" valuetype="4" value="21.000000" type="SET_POINT_TEMPERATURE"/>
    <datapoint ise_id="3680" name="HmIP-RF-0000A1A49A0D0878:1.SWITCH_POINT_OCCURED" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="0" type="SWITCH_POINT_OCCURED"/>
    <datapoint ise_id="3682" name="HmIP-RF-0000A1A49A0D0878:1.VALVE_ADAPTION" operations="7" timestamp="0" valueunit="" valuetype="2" value="0" type="VALVE_ADAPTION"/>
    <datapoint ise_id="3683" name="HmIP-RF-0000A1A49A0D0878:1.VALVE_STATE" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="4" type="VALVE_STATE"/>
    <datapoint ise_id="3684" name="HmIP-RF-0000A1A49A0D0878:1.WINDOW_STATE" operations="7" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="WINDOW_STATE"/>
  </channel>
</device>
```

## Important

Prior adding new hardware, ensure Domoticz accepts new hardware as devices will be created when adding the plugin.

Check: GUI > Setup > Settings > Hardware

Either mark accept or click Allow for 5 minutes

Goto GUI > Setup > Hardware

Select Type: HomematicIP Radiator Thermostat (HMIP-eTRV)

Give a name, i.e. Thermostat Dining

Set initially the parameter: Keep debug to true
Click button Add
Check the Domoticz log if the four devices are created and updated: Thermostat Dining - Setpoint,Thermostat Dining - Temperature,Thermostat Dining - Battery,Thermostat Dining - Valve
Check the device widgets on the GUI tabs.
If all OK, set the hardware parameter Debug to false
Optional steps (see also screenshots below):
<ul style="list-style-type: none"> <li>• Add the devices to a room plan, i.e Dining Room</li> <li>• Add a device timer to ensure the thermostat is turned Off to avoid heating overnight</li> <li>• Add a switch device to turn the thermostat off</li> </ul>
<i>Notes</i>
If adding device timers for the setpoint, set the thermostat to “Manu mode” in the Homematic WebUI (RaspberryMatic) else if “Auto mode” is set, the selected Week profile is used.

## Add new Hardware (GUI > Setup > Hardware)

The screenshot shows the 'Add new Hardware' configuration dialog in Domoticz. The configuration fields are as follows:

- Enabled:**
- Name:** Thermostat Dining
- Type:** homematicIP Radiator Thermostat (HmIP-eTRV)
- Data Timeout:** Disabled

Below these fields, there is a note: "Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Do not enable this option for devices that do not receive data!"

The device details section includes:

- homematicIP Radiator Thermostat (HmIP-eTRV) v1.2.0
- A list of supported features:
  - Set the setpoint (degrees C).
  - Get the actual temperature (degrees C).
  - Get the low battery state (true or false). Threshold is set in the HomeMatic WebUI.
  - Get the valve position (0 - 100%).
  - Supported are the devices HmIP-eTRV-B, HmIP-eTRV-2

The Domoticz Devices (Type,SubType) section lists:

- Setpoint (Thermostat,Setpoint)
- Temperature (Temp,LaCrosse TX3)
- Battery (General,Alert)
- Valve (General,Percentage)

The Hardware Configuration section contains:

- Address (CCU IP address, default: 192.168.1.225)
- IDs (obtained via XML-API script <http://ccu-ip-address/addons/xmlapi/statelist.cgi>):
  - Device ID HmIP-eTRV-B or HmIP-eTRV-2 (default: 1541)
  - Datapoint IDs(#4): SET\_POINT\_TEMPERATURE, ACTUAL\_TEMPERATURE, LOW\_BAT, LEVEL as comma separated list in this order (defaults: 1584,1567,1549,1576)
  - Note: After configuration update, the setpoint is 0. Click the setpoint to set the value.

The bottom of the dialog shows the following settings:

- CCU IP: 192.168.1.225
- Device ID: 3637
- Datapoint IDs: 3680,3663,3645,3672
- Check Interval (sec): 60
- Debug: True

A blue 'Add' button is located at the bottom right of the dialog.

## Devices, Roomplan & Dashboard for selected room

The screenshot displays three main sections of the Domoticz interface:

- Devices "Dining Room":** A table listing devices in the Dining Room, including Thermostat Dining - Valve, Thermostat Dining - Battery, Thermostat Dining - Temperature, and Thermostat Dining - Setpoint.
- Roomplan "Dining Room":** A table listing room plan entries for the Dining Room.
- Dashboard "Dining Room":** A graphical interface showing the current temperature (21.9°C), setpoint (22.0), and battery level (22%) for the Thermostat Dining device. It includes up and down arrows for adjusting the setpoint and a 'Set' button.

A blue arrow points from the 'Change setpoint' text in the Roomplan section to the up/down arrows on the Dashboard.

## Enhancement Push Off Button

### Purpose

To define a switch to handle the thermostat device state change action to Off.

### How

Define a VirtualSensor device with properties:

- Name, i.e. “MakeLab Thermostat - Off”
- Sensor Type Switch.

After adding the device, select the device widget and change the properties:

- Switch Type:  
Push Off Button
- Off Action:  
[http://ccu-ip/addons/xmlapi/statechange.cgi?ise\\_id=1584&new\\_value=0](http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=0)

The datapoint SET\_POINT\_TEMPERATURE of the device is required, i.e. 1584.

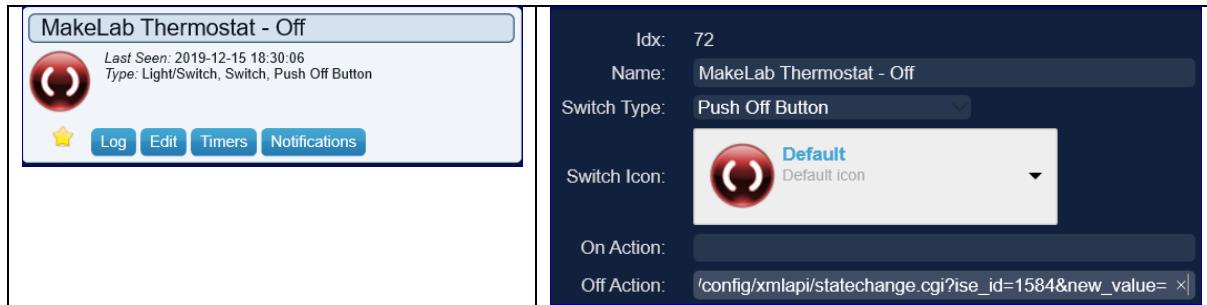
Obtain this datapoint id (ise\_id), by requesting the state of the thermostat device with

id=1541. The HTTP XML response holds various datapoints, select:

SET\_POINT\_TEMPERATURE.

```
http://raspmatic-ip/addons/xmlapi/state.cgi?device_id=1541
```

```
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" timestamp="1576829294" valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
```



### Domoticz Log

Performed test by pushing the icon on the widget, which send the HTTP XML-API request to the RaspberryMatic system to set the setpoint of the MakeLab thermostatstat to 0.

```
2019-12-15 18:30:06.767 (VirtualDevices) Light/Switch (MakeLab Thermostat - Off)
2019-12-15 18:30:06.758 Status: User: Admin initiated a switch command (72/MakeLab Thermostat - Off/Off)
```

## Timers

Each radiator thermostat has one or more timers to control the temperature.  
The timers are defined in the Domoticz Database table SetpointTimers.

Examples SQL SELECT statement to obtain the setpoint timers from the Domoticz Production database. The device names are in German.

```
SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID;
```

## Output Tool SQLiteSpy

Name	ID	Active	DeviceRowID	Date	Time	Type	Temperature	TimerPlan	Days	Month	MDay	Occurrence
Hwg Bad Sollwert	1	1	207	2019-12-15	06:00	2	20.0	0	128	0	0	0
Hwg Bad Sollwert	2	1	207	2019-12-15	11:00	2	19.0	0	128	0	0	0
Hwg Bad Sollwert	3	1	207	2019-12-14	23:00	2	17.0	0	128	0	0	0
Hwg EZ Sollwert	4	1	204	2019-12-14	06:00	2	20.0	0	128	0	0	0
Hwg EZ Sollwert	5	1	204	2019-12-14	22:00	2	17.0	0	128	0	0	0
Hwg WZ Sollwert	6	1	201	2019-12-15	06:15	2	21.0	0	128	0	0	0
Hwg WZ Sollwert	7	1	201	2019-12-14	22:00	2	0.0	0	128	0	0	0
Hwg Dusche Sollwert	8	1	198	2019-12-14	21:30	2	21.0	0	128	0	0	0
Hwg Dusche Sollwert	9	1	198	2019-12-14	23:00	2	0.0	0	128	0	0	0
Hwg MakeLab Sollwert	10	1	195	2019-12-14	19:00	2	0.0	0	128	0	0	0

## Examples SQLite Shell

The header (.header on) and mode CSV (.mode csv) are set.

More information read chapter [SQLite Shell](#).

## Full Overview

```
sqlite> SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID ORDER BY d.name;
```

```
Name,ID,Active,DeviceRowID,Date,Time,Type,Temperature,TimerPlan,Days,Month,MDay,Occurrence
"Hwg Bad Sollwert",1,1,207,2019-12-17,06:00,2,21.0,0,128,0,0,0
"Hwg Bad Sollwert",2,1,207,2019-12-15,11:00,2,19.0,0,128,0,0,0
"Hwg Bad Sollwert",3,1,207,2019-12-17,19:00,2,21.0,0,128,0,0,0
"Hwg Bad Sollwert",11,1,207,2019-12-17,23:00,2,17.0,0,128,0,0,0
"Hwg Dusche Sollwert",8,1,198,2019-12-14,21:30,2,21.0,0,128,0,0,0
"Hwg Dusche Sollwert",9,1,198,2019-12-14,23:00,2,0.0,0,128,0,0,0
"Hwg EZ Sollwert",12,1,210,2019-12-21,22:00,2,17.0,0,128,0,0,0
...and more ...
```

## Brief Overview

```
sqlite> SELECT d.Name, t.Time, t.Temperature FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID ORDER BY d.name,t.Time;
```

```
Name,Time,Temperature
"Hwg Bad Sollwert",06:00,21.0
"Hwg Bad Sollwert",11:00,19.0
... and more
```

## Example Device Timer Definition

Name: Hgz Bad Sollwert					
2019-12-15 19:27:34 ⏪ ▲08:31 ▼16:01					
Show 25	entries	Date	Time	Command	Days
Yes	On Time		06:00	Temperature, 20	Everyday
Yes	On Time		11:00	Temperature, 19	Everyday
Yes	On Time		23:00	Temperature, 17	Everyday

## Timerplans

The timerplan default (ID=0) is assigned.

In the GUI > Setup > Settings > Other: the timer plan can be set.



The selected timer plan ID, can be found in the Domoticz database, table Preferences , field Key with content ActiveTimerPlan.

# Raspberry Pi Monitor

## Purpose

To measure & show specific Raspberry Pi system information and trigger alarm if a value is above a certain threshold.

Monitored are: Memory Usage, Hard Disc Space, Temperature.

## Solution

Use the Domoticz Hardware Motherboard, which will create several devices. The devices can be used to trigger events.

### Hardware Motherboard

Idx	Hardware	ID	Unit	Name	Type	SubType	Value
1	Motherboard	0000044C	1	RPi Memory Usage	General	Percentage	20.93%
2	Motherboard	0000044E	1	HDD /boot	General	Percentage	1.5%
3	Motherboard	0000044F	1	RPi HDD	General	Percentage	32.32%
4	Motherboard	0001	1	RPiTemp	Temp	LaCrosse TX3	44.0 C
5	Motherboard	0000044D	1	CPU_Usage	General	Percentage	2.84%

**Enabled:**

**Name:** Motherboard

**Type:** Motherboard sensors

**Data Timeout:** Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
**Do not enable this option for devices that do not receive data!**

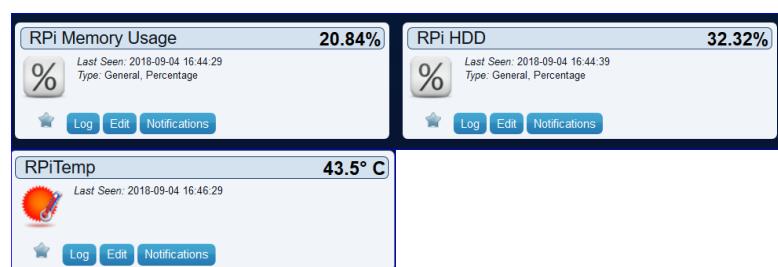
### Devices derived from Hardware Motherboard

Idx	Hardware	ID	Unit	Name	Type	SubType	Value
1	Motherboard	0000044C	1	RPi Memory Usage	General	Percentage	20.93%
2	Motherboard	0000044E	1	HDD /boot	General	Percentage	1.5%
3	Motherboard	0000044F	1	RPi HDD	General	Percentage	32.32%
4	Motherboard	0001	1	RPiTemp	Temp	LaCrosse TX3	44.0 C
5	Motherboard	0000044D	1	CPU_Usage	General	Percentage	2.84%

#### Note

Not all devices are added, (only) RPi Memory Usage, RPi HDD, RPi Temp.

### Widgets (Tabs Utility & Temperature)



# Monitoring

To monitor selective devices and notify if above threshold.

The notification is done using the Virtual Device Alert Message (IDX=55).



An email notification is sent!

## User Variables Thresholds

The thresholds are defined as user variables and used by the dzVents script. (Setup > More Options > User variables).

Idx	Variable name	Variable type	Current value
5	TH_RPI_MEMORYUSAGE	Integer	80
4	TH_RPI_HDDUSAGE	Integer	70
3	TH_RPI_TEMPERATURE	Integer	41

## Automation Script

Event name: rpi\_monitor

```
--[[[rpi_monitor.lua
Monitor the Raspberry Pi and notify, via alert message, in case threshold reached or exceeded.
Project: AtHome
Interpreter: dzVents, Device
See: AtHome.pdf
Author: Robert W.B. Linn
Version: 20180910
]]]

-- Module msgbox: /home/pi/domoticz/scripts/dzVents/scripts
local msgbox = require('msgbox')

-- Idx of the devices
local IDX_RPI_MEMORYUSAGE = 1
local IDX_RPI_HDDUSAGE = 3
local IDX_RPI_TEMPERATURE = 4

-- Thresholds are set via user variables, i.e. TH_RPI_MEMORYUSAGE

-- Check if the device state exceeds threshold and update control message
local function checkthreshold(domoticz, device, threshold)
    if (tonumber(device.state) > threshold) then
        local message = device.name .. ' above threshold ' .. threshold .. ' (' .. device.state .. ')'
        msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
    end
end

return {
    on = {
        -- Devices idx to monitor
        devices = {
            IDX_RPI_MEMORYUSAGE,
            IDX_RPI_HDDUSAGE,
```

```
        IDX_RPI_TEMPERATURE
    },
execute = function(domoticz, device)
    -- RPi Memory Usage
    if (device.idx == IDX_RPI_MEMORYUSAGE) then
        checkthreshold(domoticz,device,domoticz.variables('TH_RPI_MEMORYUSAGE').value)
    end
    -- RPi HDD Usage
    if (device.idx == IDX_RPI_HDDUSAGE) then
        checkthreshold(domoticz,device,domoticz.variables('TH_RPI_HDDUSAGE').value)
    end
    -- RPi Temperature
    if (device.idx == IDX_RPI_TEMPERATURE) then
        checkthreshold(domoticz,device,domoticz.variables('TH_RPI_TEMPERATURE').value)
    end
end
}
```

# RaspberryMatic Duty-Cycle Monitor

## Purpose

To monitor the RaspberryMatic DutyCycle CCU using the System Variable “DutyCycle” and send an Alert message in case value exceeds threshold.

*Note*

The threshold value is set to 4% for testing and to 10% for operation.

### Screenshot Domoticz Dashboard

Alert message with DutyCycle above threshold (“Grenzwert”) 4%.



### Domoticz Log Entry with DutyCycle above threshold

```
2020-10-19 10:11:00.767 Error: dzVents: Error: (3.0.2) RaspMatic DutyCycle 9%, Grenzwert 4% (19-10 10:11)
```

### Domoticz User Variable with threshold 4%

Idx	Variable name	Variable type	Current value
23	TH_RASPBERRYMATIC_DUTYCYCLE	Integer	4

### RaspberryMatic GUI Home Page with DutyCycle

### RaspberryMatic System Variable DutyCycle

Name	Description	Last modified	Status
DutyCycle	DutyCycle CCU	10.09.2020 10:35:00	1.00 %

## Screenshot Domoticz Device Widget and Day Log (extract)



## Solution

An Automation Event dzVents Lua script requests from the RaspberryMatic system (Pull, HTTP XML-API GET Request) in regular intervals the value of the System Variable DutyCycle and updates the Domoticz percentage device every 10 min. The script checks if the DutyCycle value is above a threshold, set by a UserVariable and updates an Alert sensor to level 4 (Red).

Before explaining the solution, some information on handling HTTP XML-API requests.

## HTTP XML-API Requests

Required is the value of the RaspberryMatic system variable DutyCycle. To get the DutyCycle value, the ise\_id of the system variable is required. The ise\_id is obtained from the list of RaspberryMatic system variables. The ise\_id of the system variable DutyCycle is 1390 (see below HTTP Response).

### HTTP XML-API GET Request for all System Variables

```
http://ccu-ip/addons/xmlapi/sysvarlist.cgi
```

### HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<systemVariables>
  <systemVariable value_name_1="${sysVarAlarmZone1Triggered}"
  value_name_0="${sysVarAlarmZone1NotTriggered}" timestamp="0" visible="true" logged="false" subtype="6"
  type="2" unit="" max="" min="" ise_id="1235" value_list="" value="" variable="4"
  name="${sysVarAlarmZone1}"/>
  <systemVariable value_name_1="${sysVarPresencePresent}" value_name_0="${sysVarPresenceNotPresent}"
  timestamp="1587921018" visible="true" logged="false" subtype="2" type="2" unit="" max="" min=""
  ise_id="950" value_list="" value="true" variable="1" name="${sysVarPresence}"/>
  <systemVariable value_name_1="" value_name_0="" timestamp="1587970740" visible="true" logged="false"
  subtype="0" type="4" unit "%" max="100" min="-1" ise_id="1390" value_list="" value="2.000000"
  variable="2.000000" name="DutyCycle"/>
  <systemVariable value_name_1="${sysVarAlarmZone1Triggered}"
  value_name_0="${sysVarAlarmZone1NotTriggered}" timestamp="0" visible="true" logged="false" subtype="6"
  type="2" unit="" max="" min="" ise_id="1416" value_list="" value="" variable="4" name="WatchDog-
  Alarm"/>
</systemVariables>
```

### GET Single System Variable

Example DutyCycle with ise\_id 1390 (see previous HTTP XML-API Response).

## HTTP Request

```
http://ccu-ip/addons/xmlapi/sysvar.cgi?ise_id=1390
```

## HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<systemVariables>
  <systemVariable value_name_1="" value_name_0="" timestamp="1587970920" subtype="0" type="4"
unit "%" max="100" min="-1" ise_id="1390" value_text="" value_list="" value="3.000000"
variable="3.000000" name="DutyCycle"/>
</systemVariables>
```

## Domoticz Configuration

### Device

The RaspberryMatic system variable DutyCycle is configured as a Percentage device in Domoticz - Created as a virtual sensor for hardware Dummy named VirtualSensor.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
301	VirtualSensors	00082301	1	RaspberryMatic DutyCycle	General	Percentage	1%

### User Variable

A user variable is defined to set the threshold (in pct) of the DutyCycle.  
If the actual value of the DutyCycle exceeds the threshold, a log entry or an alert is sent.

Idx	Variable name	Variable type	Current value
23	TH_RASPBERRYMATIC_DUTYCYCLE	Integer	10

## Automation Script

The dzVents script requests, via HTTP XML-API, in regular intervals (i.e. 10 min), the status of the RaspberryMatic system variable DutyCycle.

The HTTP XML response is parsed to get the DutyCycle value from the XML tree and updates the Domoticz Device RaspberryMatic Duty Cycle.

Source: raspmatic\_dutycycle\_monitor.dzvents

```
-- url http xml-api request system variable. The ise_id is set in the function openURL
local URL_RASPMATIC = 'http://raspmatic-ip/addons/xmlapi/sysvar.cgi?ise_id=';
-- define the unique (across all dzVents) callback
local RES_RASPMATIC = "RES_RASPMATIC_DUTYCYCLE_MONITOR";
-- set the datapoint ise_id of the "DutyCycle" with ise_id 1390
local DATAPOINT_ISE_ID = 1390
-- domoticz dutycycle device
local IDX_RASPMATIC_DUTYCYCLE = 301
-- set the alert message if dutycycle value is above threshold as set by uservariable
TH_RASPBERRYMATIC_DUTYCYCLE
-- (idx,name,type,value): 23, TH_RASPBERRYMATIC_DUTYCYCLE, Integer, 10
local IDX_UV_TH_RASPBERRYMATIC_DUTYCYCLE = 23
-- Timer interval 'every 10 minutes' - for tests use: 'every minute'
local TIMER_RULE = 'every 10 minutes'

return {
on = {
    timer = { TIMER_RULE },
    httpResponses = { RES_RASPMATIC, }
},
execute = function(domoticz, item)
    if (item.isTimer) then
        domoticz.openURL({url=URL_RASPMATIC .. DATAPOINT_ISE_ID,method='GET',callback=RES_RASPMATIC,})
    end
    if (item.isHTTPResponse) then
        if (item.statusCode == 200) then
            if (item.callback == RES_RASPMATIC) then
                -- get the key value from the http response
                -- <systemVariable value_name_1="" value_name_0="" timestamp="1587970920" subtype="0" type="4"
unit "%" max="100" min="-1" ise_id="1390" value_text="" value_list="" value="3.000000"
variable="3.000000" name="DutyCycle"/>
                local value = tonumber(domoticz_applyXPath(item.data,
'//systemVariable[@ise_id="'.DATAPOINT_ISE_ID..']/@value'))
                domoticz.devices(IDX_RASPMATIC_DUTYCYCLE).updatePercentage(value);
                local threshold = domoticz.variables(IDX_UV_TH_RASPBERRYMATIC_DUTYCYCLE).value
                local message = ""
                -- Check the actual value against the threshold
                if value > threshold then
                    message = string.format("RaspMatic DutyCycle %.0f%%, Grenzwert %d%% (%s)",
value, threshold, domoticz.helpers.isnowshort(domoticz))
                    domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
                else
                    message = string.format("RaspMatic DutyCycle %.0f%%, Grenzwert %d%% (%s)",
value, threshold, domoticz.helpers.isnowshort(domoticz))
                end
                end
            else
                domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
}
```

# Remote Control (HmIP-RC8)

## Purpose

To remote control selected functions using a handheld Remote-Control device.

## Solution

This solution makes use of the Homematic IP Remote Control - 8 buttons (HmIP-RC8).

	<p>The HmIP-RC8 device is integrated in the RaspberryMatic system running a CCU3 (see explore <a href="#">RaspberryMatic</a>).</p>
---	--

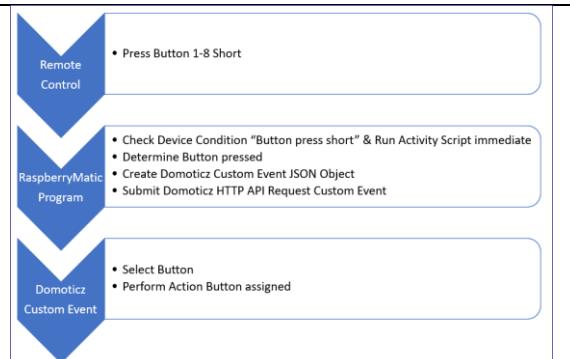
There are two options worked out on using the Remote-Control.

### Option Custom Event

The Remote-Control device submits, via a RaspberryMatic program, a HTTP API request to Domoticz with Custom Event data holding the button number (and datapoint id of the button - not used).

An Automation Event dzVents with trigger customEvents acts according button number pressed.

There are no Domoticz devices required.

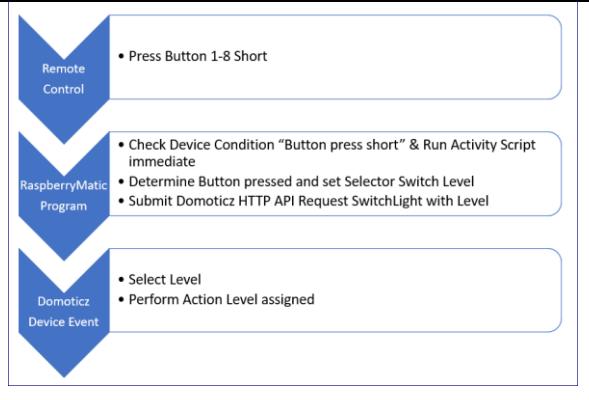


## Option Selector Switch

The Remote-Control device submits, via a RaspberryMatic program, a HTTP API request to Domoticz with Selector Switch Level depending button number.

An Automation Event dzVents with trigger device, acts according level selected.

A Domoticz Selector Switch device is required (Hardware Dummy, Virtual Sensor).



The first option is rather straight forward, and the button actions can only be used via the Remote-Control device.

The second option enables to use the button actions also from the Domoticz GUI (i.e. widget or dashboard) or integrate in Custom Web Pages or UI.

The first option is currently used.

# RaspberryMatic Configuration

## Teach-In

The Remote-Control device is “teach-in” to RaspberryMatic via the Homematic WebUI (standard approach to add devices).

Type	Description	Picture	Serial number	Interface category	Transmission mode	Name	Function	Room	Functional text	Action	Done			
HmIP-RC8	Homematic IP Remote Control 8-channel		000B1BE98D94DE	HmIP-RF	Secured	HmIP-RC8 000B1BE98D94DE			<input type="button" value="Test"/>	<input type="button" value="Delete"/>	<input checked="" type="checkbox"/> operable	<input checked="" type="checkbox"/> visible	<input type="checkbox"/> logged	<input type="button" value="Done"/>

**Renamed from HmIP-RC8 000B1BE98D94DE to Remote Control**

Channel	Room	Function	Last modified
HmIP-RC8 000B1BE98D94DE:1			16.10.2020 12:16:30
Push button			
8 Push buttons			

## Get Datapoints

The Remote-Control device datapoints are required to determine which button is pressed to submit an action for Domoticz.

The list of datapoints are gathered by submitting a XMLAPI request to RaspberryMatic.

### Steps

1. From the RaspberryMatic system, get the list of devices as XML document tree.
2. Select the device name="Remote Control".
3. Get the device properties from which the ise\_id of the datapoints type="PRESS\_SHORT" are used in a RaspberryMatic script to determine which button has been pressed.

Submit URL to the RaspberryMatic system to get the list of devices:

```
http://raspmatic-ip/addons/xmlapi/statelist.cgi
```

From the response, select the name “Remote Control” to get the device id (i.e. 8397)

```
<device name="Remote Control" ise_id="8397" unreach="false" config_pending="false">
```

Get the Remote-Control device properties – either from previous XML tree or submit

```
http://raspmatic-ip/addons/xmlapi/state.cgi?device_id=8397
```

Below response XML document tree, hard to read, is just to show the content.  
Required are the ise\_id of the datapoints type “PRESS\_SHORT”:

```
<datapoint name="HmIP-RF.000B1BE98D94DE:1.PRESS_SHORT" type="PRESS_SHORT" ise_id="8423" value="" valuetype="2" valueunit="" timestamp="0"/>
```

```
<state>
  <device name="Remote Control" ise_id="8397" unreach="false" config_pending="false">
    <><channel name="Remote Control:0" ise_id="8398">
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.CONFIG_PENDING" type="CONFIG_PENDING" ise_id="8399" value="false" valuetype="2" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.DUTY_CYCLE" type="DUTY_CYCLE" ise_id="8403" value="false" valuetype="2" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.LOW_BAT" type="LOW_BAT" ise_id="8405" value="false" valuetype="2" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.OPERATING_VOLTAGE" type="OPERATING_VOLTAGE" ise_id="8409" value="3.00000" valuetype="4" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.OPERATING_VOLTAGE_STATUS" type="OPERATING_VOLTAGE_STATUS" ise_id="8410" value="188" valuetype="16" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.RSSI_DEVICE" type="RSSI_DEVICE" ise_id="8411" value="188" valuetype="8" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.RSSI_PEER" type="RSSI_PEER" ise_id="8412" value="0" valuetype="8" valueunit="" timestamp="0"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.UNREACH" type="UNREACH" ise_id="8413" value="false" valuetype="2" valueunit="" timestamp="1602932195"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:0.UPDATE_PENDING" type="UPDATE_PENDING" ise_id="8417" value="false" valuetype="2" valueunit="" timestamp="1602932712"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:1" ise_id="8421">
      <datapoint name="HmIP-RF.000B1BE98D94DE:1.PRESS_LONG" type="PRESS_LONG" ise_id="8422" value="false" valuetype="2" valueunit="" timestamp="1602843015"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:1.PRESS_SHORT" type="PRESS_SHORT" ise_id="8423" value="false" valuetype="2" valueunit="" timestamp="1602930321"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:2" ise_id="8424">
      <datapoint name="HmIP-RF.000B1BE98D94DE:2.PRESS_LONG" type="PRESS_LONG" ise_id="8425" value="false" valuetype="2" valueunit="" timestamp="1602843006"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:2.PRESS_SHORT" type="PRESS_SHORT" ise_id="8426" value="false" valuetype="2" valueunit="" timestamp="1602930328"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:3" ise_id="8427">
      <datapoint name="HmIP-RF.000B1BE98D94DE:3.PRESS_LONG" type="PRESS_LONG" ise_id="8428" value="false" valuetype="2" valueunit="" timestamp="1602845598"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:3.PRESS_SHORT" type="PRESS_SHORT" ise_id="8429" value="false" valuetype="2" valueunit="" timestamp="1602930871"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:4" ise_id="8430">
      <datapoint name="HmIP-RF.000B1BE98D94DE:4.PRESS_LONG" type="PRESS_LONG" ise_id="8431" value="false" valuetype="2" valueunit="" timestamp="1602844303"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:4.PRESS_SHORT" type="PRESS_SHORT" ise_id="8432" value="false" valuetype="2" valueunit="" timestamp="1602930854"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:5" ise_id="8433">
      <datapoint name="HmIP-RF.000B1BE98D94DE:5.PRESS_LONG" type="PRESS_LONG" ise_id="8434" value="0" valuetype="2" valueunit="" timestamp="0"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:5.PRESS_SHORT" type="PRESS_SHORT" ise_id="8435" value="false" valuetype="2" valueunit="" timestamp="1602848911"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:6" ise_id="8436">
      <datapoint name="HmIP-RF.000B1BE98D94DE:6.PRESS_LONG" type="PRESS_LONG" ise_id="8437" value="0" valuetype="2" valueunit="" timestamp="0"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:6.PRESS_SHORT" type="PRESS_SHORT" ise_id="8438" value="false" valuetype="2" valueunit="" timestamp="1602849123"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:7" ise_id="8439">
      <datapoint name="HmIP-RF.000B1BE98D94DE:7.PRESS_LONG" type="PRESS_LONG" ise_id="8440" value="0" valuetype="2" valueunit="" timestamp="0"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:7.PRESS_SHORT" type="PRESS_SHORT" ise_id="8441" value="false" valuetype="2" valueunit="" timestamp="1602930862"/>
    </channel>
    <channel name="HmIP-RC8 000B1BE98D94DE:8" ise_id="8442">
      <datapoint name="HmIP-RF.000B1BE98D94DE:8.PRESS_LONG" type="PRESS_LONG" ise_id="8443" value="0" valuetype="2" valueunit="" timestamp="0"/>
      <datapoint name="HmIP-RF.000B1BE98D94DE:8.PRESS_SHORT" type="PRESS_SHORT" ise_id="8444" value="false" valuetype="2" valueunit="" timestamp="1602930865"/>
    </channel>
  </device>
</state>
```

## Solution Custom Event

### RaspberryMatic Program & Script

The screenshot shows the RaspberryMatic program with the condition based on the Remote-Control device button short pressed. Immediate action is taken by executing the script.

The script assigns the datapoint id of the button short to the button. A JSON object is created with id and buttonnr, used for the data parameter of the HTTP API custom event request to Domoticz.

Source Script: remote\_control\_custom\_event.script.

The screenshot displays the RaspberryMatic software interface. At the top, there's a navigation bar with tabs like 'Home page', 'Status and control', 'Programs and connections', and 'Programming'. Below the navigation bar is a table listing a single item: 'Remote Control' with the description 'Remote Control specific functions'. This row has a 'Condition (if...)' column containing a complex logic expression involving multiple device selection and OR conditions. The 'Activity (then..., or else...)' column contains the command 'Script: ... immediately run' and the 'Action' column is set to 'Intrinsic'. Below the table is a large code editor window. The code is a script in a language that uses comments starting with '!' followed by English text. It defines variables like 'string Action = "8"'; 'string urlBase = "http://domoticz-ip:port/json.htm";', and 'string customEvent = "raspematic\_remote\_control";'. It then loops through various button IDs (8, 2, 3, 4, 5, 6, 7, 8) to map them to button numbers (1-8). Finally, it constructs a JSON object with 'data = {"id":datapointID,"buttonnr":buttonNr}'; and sends an HTTP POST request to the URL 'urlRequest = urlBase+type=command#param=customevent#event="#customEvent#data="#data#";'. The entire script is enclosed in a red box. A yellow box highlights the first five lines of the script, which are the steps: 1.Get Datapoint ID, 2.Set Button 1-8, 3.Create JSON Object, 4.Submit Domoticz HTTP API Request Custom Event, 5.Custom Event Action Button selected.

```

! Function: Remote Control
! if a button of the remote control is pressed, send custom event with button nr to Domoticz to action according.
! via cuxd an http api request for a customevent is submitted to domoticz.
! custom event data parameter must be in json format.

string Action = "8";
! Domoticz system url base for the http api request
string urlBase = "http://domoticz-ip:port/json.htm"; ! Production
string customEvent = "raspematic_remote_control";
string datapointID = 0; ! 8429
string buttonNr = 0; ! 1 - 8

! Get the object datapoint of the remote control
object objDatapoint = dom.GetObject ("$src");
if (objDatapoint) {
    datapointID = objDatapoint.ID(); ! 8423

    ! Map the raspematic remote control datapoint to the button nr
    ! BUTTON SHORT ON BUTTON LONG
    if (datapointID == "8423") { buttonNr = 1; } ! 8422
    if (datapointID == "8426") { buttonNr = 2; } ! 8425
    if (datapointID == "8429") { buttonNr = 3; } ! 8428
    if (datapointID == "8432") { buttonNr = 4; } ! 8431
    if (datapointID == "8435") { buttonNr = 5; } ! 8434
    if (datapointID == "8438") { buttonNr = 6; } ! 8437
    if (datapointID == "8441") { buttonNr = 7; } ! 8440
    if (datapointID == "8444") { buttonNr = 8; } ! 8443

    ! Custom event data parameter must be in json format
    string data = ("{"id":'datapointID',"buttonnr":'buttonNr'}");
    ! Build the Domoticz http rest request url to update the device using customevent
    string urlRequest = urlBase+type=command#param=customevent#event="#customEvent#data="#data#";
    ! Run the command without a return result
    CmdRes = dom.GetObject("CUD.CUX2881001:1:CUD_EXEC").State("wget -q -O - #urlRequest");
}

```

## Domoticz Automation Event

Source Script: remote\_control\_custom\_event.dzvents.

**Domoticz Custom Event triggered by the RaspberryMatic Script**

```
--[[  
remote_control.dzvents  
Trigger: customEvent  
To handle remote control buttons from the HomematicIP HmIP-RC8 device connected to a RaspberryMatic system.  
A RaspMatic script creates a custom event with data {[{"id":8444, "buttonnr":8}  
An action is assigned to every button - see below.  
]]--  
local CUSTOMEVENTNAME = 'raspmatic_remote_control'  
return {  
    on = {  
        customEvents = { CUSTOMEVENTNAME, }  
    },  
    logging = {  
        level = domoticz.LOG_INFO, marker = 'REMOTECONTROL',  
    },  
    execute = function(domoticz, item)  
        local data = item.data  
        local buttonNr = data.buttonnr  
        if buttonNr == 1 then  
            -- action  
        end  
        if buttonNr == 2 then  
            -- action  
        end  
        -- more buttons ...  
    end  
}
```

## Solution Selector Switch

### Domoticz Device Configuration

Domoticz GUI > Setup > Devices

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
41	VirtualSensors	00014079	1	Remote Control	Light/Switch	Selector Switch	Off

Domoticz GUI > Switches

**Remote Control** 2

Last Seen: 2020-10-17 13:28:59  
Type: Light/Switch, Selector Switch, Selector



1 2 3 4 5 6 7 8

★ Log Edit Timers Notifications

Device has 9 levels.  
Level 0 Off is not used (hidden).

Level 10 -80 are have level names 1-8 according Remote-Control device buttons.

The Level property is used in the Automation Event to set the action for the button pressed.

Id: 41  
Name: Remote Control  
Switch Type: Selector

Switch Icon: Default

On Delay: 0 (Seconds) 0 : Disabled  
Off Delay: 0 (Seconds) 0 : Disabled  
Protected:   
Selector Style:  Button set  Select menu  
Hide Off Level

Selector Levels:

Level	Level name	Order
0	Off	2 0
10	1	2 1
20	2	2 2
30	3	2 3
40	4	2 4
50	5	2 5
60	6	2 6
70	7	2 7
80	8	2 8

Selector actions:

Level	Action	Order
0	0	2 0
10	1	2 1
20	2	2 2
30	3	2 3
40	4	2 4
50	5	2 5
60	6	2 6
70	7	2 7
80	8	2 8

Description:

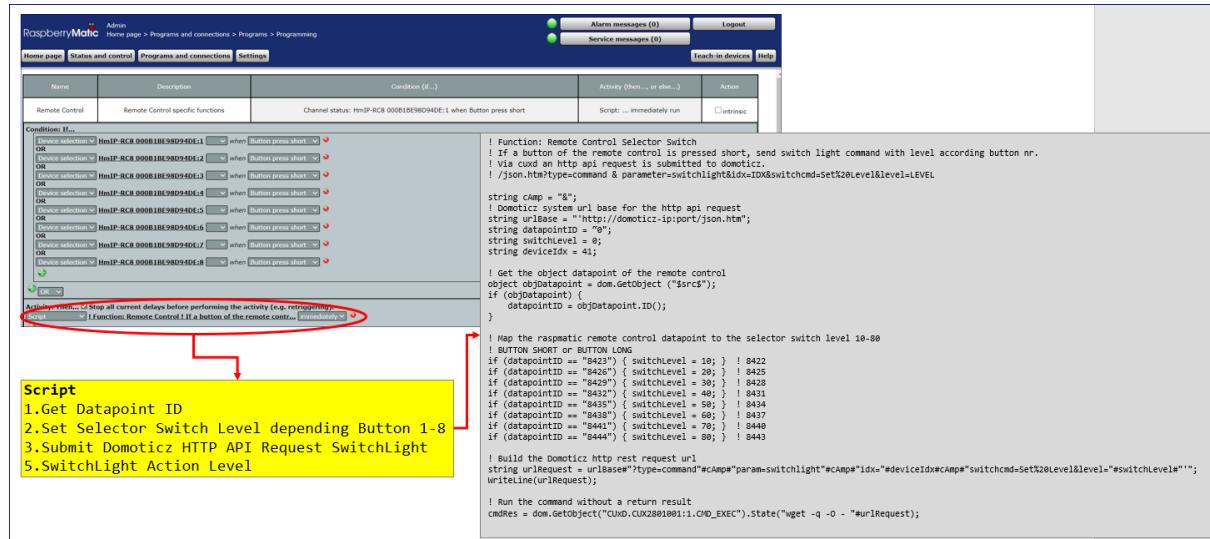
Save Delete Replace

## RaspberryMatic Program & Script

The screenshot shows the RaspberryMatic program with the condition based on the Remote-Control device button short pressed. Immediate action is taken by executing the script.

The script assigns the datapoint id of the button short to the switch level 10-80 for buttons 1-8. The HTTP API switch light request with Selector Switch Idx and level are submitted to Domoticz.

Source Script: remote\_control\_selector\_switch.script.



## Domoticz Automation Event

Source Script: remote\_control\_selector\_switch.dzvents.

### Domoticz Device Event triggered by the RaspberryMatic Script

```
--[[  
    remote_control_selector_switch.dzvents  
    To take action based on a selector switch devices trigger.  
    The selector switch level is set by a RaspMatic script triggered by a remote control device pressing short.  
    The RaspMatic script submits a HTTP API request with level 10-80 for button 1-8  
    http://domoticz-ip:port/json.htm?type=command&param=switchlight&idx=IDX&switchcmd=Set%20Level&level=LEVEL  
20201017 rwbl  
]]--  
  
-- Idx f the device*s(  
local IDX_SELECTOR_SWITCH_RC = 41  
  
return {  
    on = {  
        devices = {IDX_SELECTOR_SWITCH_RC}  
    },  
    execute = function(domoticz, device)  
        -- Button 1  
        if device.level == 10 then  
            domoticz.log(("Remote Control Button %d action"):format(device.level / 10))  
        end  
        -- Button 2  
        if device.level == 20 then  
            domoticz.log(("Remote Control Button %d action"):format(device.level / 10))  
        end  
    end  
}
```

# River Elbe Tide (WSV Service)

## Purpose

To monitor the level (absolute above sea-level in cm [Abs NHN]) and tide (Ebb | Flood) of the river “Elbe” at location “Schulau” near Hamburg (Germany).

## Solution

An Automation Event dzVents Lua requests every 15 minutes river “Elbe” information via HTTP request ([example](#)) to a German webservice PEGELONLINE.wsv.de.

The HTTP JSON response is parsed to get the river level current measurement "Relative PNP" and the reference value "PNP" which are used to calculate the "Abs NHN".

The tide Ebb or Flood is also obtained from the HTTP response and used in the Domoticz device name as indicator.

### Example HTTP JSON Response containing the River Data

```
{
  "uuid": "f3c6ee73-5561-4068-96ec-364016e7d9ef",
  "number": "5950090",
  "shortname": "SCHULAU",
  "longname": "SCHULAU",
  "km": 641.0,
  "agency": "WSA HAMBURG",
  "longitude": 9.702887479041944,
  "latitude": 53.567910528723715,
  "water": {
    "shortname": "ELBE",
    "longname": "ELBE"
  },
  "timeseries": [
    {
      "shortname": "W",
      "longname": "WASSERSTAND ROHDATEN",
      "unit": "cm",
      "equidistance": 1,
      "currentMeasurement": {
        "timestamp": "2020-09-09T09:46:00+02:00",
        "value": 650.0,
        "trend": -1,
        "stateMnwMhw": "unknown",
        "stateNswHsw": "unknown"
      },
      "gaugeZero": {
        "unit": "m. ü. NHN",
        "value": -5.034,
        "validFrom": "2019-11-01"
      }
    }
  ]
}
```

From the JSON data, following keys are used to calculate the river level and get the tide.  
This is an extract of the Automation event dzVents Lua script below:

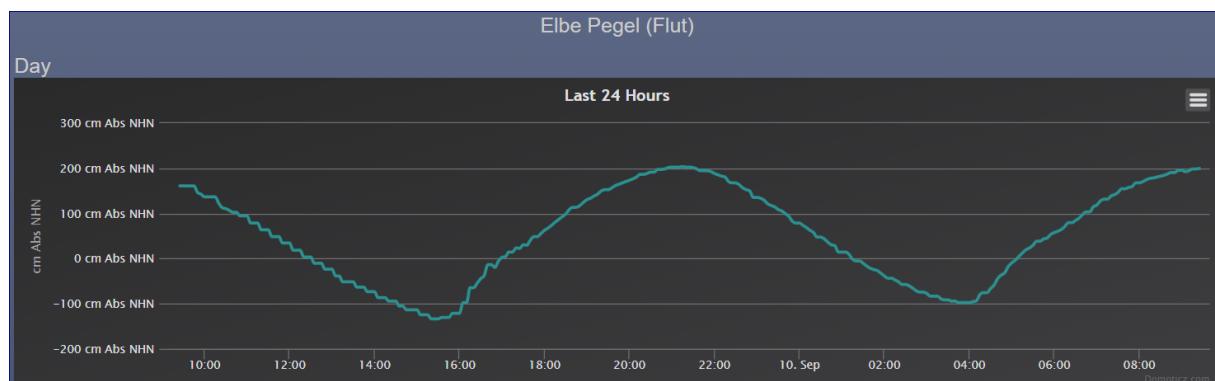
```
-- Get the first entry of the timeseries. NOTE: starts with 1 and not 0!
timeseries = item.json.timeseries[1]
local riverlevelrelpnp = timeseries.currentMeasurement.value
local riverlevelpnp = timeseries.gaugeZero.value;
local riverlevelabsnhn = riverlevelrelpnp + (riverlevelpnp * 100);
-- Tide
local trend = timeseries.currentMeasurement.trend
local tide = ""
if trend == 1 then tide = "Flut" end
if trend == 0 then tide = "Keine Änderung" end
if trend == -1 then tide = "Ebbe" end
```

## Domoticz Configuration

A single custom sensor is created which name and Data are updated by the Automation event.

Idx	Hardware	ID	Unit	Name	Type	Sub Type	Data
302	VirtualSensors	00082302	1	Elbe Pegel (Ebbe)	General	Custom Sensor	106.6 cm Abs NHN

Example of a nice tide graph.



## Automation Script

The dzVents Lua script event is time triggered every 15 minutes.

Event name: river\_elbe\_level.dzvents

```
-- river_elbe_level.dzvents
-- Get the river elbe level Abs NHN every 5 minutes.
-- Request URL: see below
-- 20200824 rwbl

-- Domoticz devices
-- Idx, Hardware, Name, Type, SubType, Data
-- 302, VirtualSensors, Elbe Pegel, General, Custom Sensor, 161 cm Abs NHN
local IDX_RIVER_LEVEL = 302

-- URL HTTP request from German webservice PEGELOWORLD.ws.de
local REQUEST_URL = 'https://www.pegeloworld.ws.de/webservices/rest-
api/v2/stations/SCHULAU.json?includeTimeseries=true&includeCurrentMeasurement=true';

-- callback of the url request - must be unique across all dzevents
local REQUEST_RES = 'RIVERELBELEVEL';

return {
  on = {
    timer = {
      'every 5 minutes'
      -- 'every minute' -- for tests
    },
  },
}
```

```

httpResponses = {
    REQUEST_RES
}
},
execute = function(domoticz, item)
    -- Timer requesting river level data
    if (item.isTimer) then
        domoticz.openURL({ url = REQUEST_URL, method = 'GET', callback = REQUEST_RES, })
    end
    -- HTTP Response. Important to check on the callback because for the name also update an HTTP
request is used
    if (item.isHTTPResponse) then
        -- domoticz.log(item.data)
        if (item.ok and item.callback == REQUEST_RES) then
            if (item.isJSON) then
                -- Get the first entry of the timeseries. NOTE: starts with 1 and not 0!
                timeseries = item.json.timeseries[1]
                local riverlevelrelpnp = timeseries.currentMeasurement.value
                local riverlevelpnp = timeseries.gaugeZero.value;
                local riverlevelabsnhn = riverlevelrelpnp + (riverlevelpnp * 100);
                domoticz.devices(IDX_RIVER_LEVEL).updateCustomSensor(riverlevelabsnhn);
                -- Tide
                local trend = timeseries.currentMeasurement.trend
                local tide = ""
                if trend == 1 then tide = "Flut" end
                if trend == 0 then tide = "Keine Änderung" end
                if trend == -1 then tide = "Ebbe" end
                -- Amend the tide information to the device name, i.e. "Elbe Pegel (Flut)"
                local newname = domoticz.utils.urlEncode(string.format("Elbe Pegel (%s)", tide))
                local urlRequest =
string.format("http://127.0.0.1:8080/json.htm?type=setused&idx=%d&used=true&name=%s",
                    IDX_RIVER_LEVEL, newname)
                -- Set the new name via url without response
                domoticz.openURL({url = urlRequest})
                domoticz.log(string.format("Abs NHN in cm: %.2f (%s)", riverlevelabsnhn, newname),
domoticz.LOG_INFO)
            end
        else
            domoticz.log(string.format('Problem handling the request: %s',UV_URL), domoticz.LOG_ERROR)
            domoticz.log(item, domoticz.LOG_ERROR)
        end
    end
end
}
}

```

## Domoticz Log

```

2020-09-09 10:31:00.406 Status: dzVents: Info: ----- Start internal script: river_elbe_level:,
trigger: "every 15 minutes"
2020-09-09 10:31:00.406 Status: dzVents: Info: ----- Finished river_elbe_level

2020-09-09 10:31:00.774 Status: dzVents: Info: ----- Start internal script: river_elbe_level:
HTTPResponse: "RIVERELBELEVEL"
2020-09-09 10:31:00.791 Status: dzVents: Info: Abs NHN in cm: 106.60 (Elbe+Pegel+%28Ebbe%29)
2020-09-09 10:31:00.791 Status: dzVents: Info: ----- Finished river_elbe_level
2020-09-09 10:31:00.791 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua

```

## Outlook

A notification warning, via email or other, could be given if a very high is reached – threshold for example at 200 cm. The therhold can be defined as user variable.

# Roomplans

## Purpose

To define roomplans containing all devices. The roomplans are used to control the rooms via the GUI, but also to organize the devices in a proper way.

To access a device, a roomplan should be selected.

The GUI Dashboard contains (only) a few devices for a quick lookup.

## Define Roomplan

The roomplans are defined via the GUI > Setup > More options > Plans.

Example list of roomplan and for a room with devices.

Idx	Name
1	\$Hidden Devices
13	Bad
14	Dusche
11	Esszimmer (EZ)
18	Flur
16	Haus
2	Information
17	Keller
15	MakeLab
19	Schlafzimmer (SZ)
4	Termine
12	Wohnzimmer (WZ)
8	Hardware Monitor
5	Volumio

Idx	Name
12	Wohnzimmer (WZ)
Showing 1 to 1 of 1 entries (filtered)	
<a href="#">Edit</a>	<a href="#">Delete</a>
<b>Devices</b> (Select Plan first to Edit...)	
Show 10 entries	
Idx	Name
112	Hue WZ TV
185	Hue WZ Stehlampe
113	Hue WZ Rob
40	WZ Temperatur
201	Hdg WZ-1 Sollwert
202	Hdg WZ-1 Temperatur
203	Hdg WZ-1 Batterie
204	Hdg WZ-2 Sollwert
205	Hdg WZ-2 Temperatur
206	Hdg WZ-2 Batterie
12	Markise WZ
Showing 1 to 11 of 11 entries	

### Notes

The definition of the roomplans is not completed yet.

Still thinking about how to organize in a simple way, do not have too many rooms and the naming convention of the roomplan (which is used for defining the device names = see next). This also relates for defining floorplans.

## Define Device Names

Define naming convention and organise device according roomplans.

### Rules

- Device naming convention: Hardware Roomplan Device.
- A roomplan gets an abbreviation, max 3 characters, if the roomplan name is longer than 7 characters.
- Domoticz production system: all device names translated to german.
- Domoticz development system: all device names in english.
- Define roomplans and associate devices.
- All devices must be associated with a roomplan.

### Examples\_(Hardware, Roomplan, Device)

- Hue Light located in the living room near TV.  
Roomplan: Wohnzimmer (WZ).  
Device name: Hue WZ TV
- Thermostat dining room with three devices setpoint,temperature, battery.  
Roomplan: Esszammer (EZ).  
Device name: Hzg EZ Sollwert, Hzg EZ Temperatur, Hzg EZ Batterie

## Overview Roomplans

There is no complete overview of the roomplans defined with devices associated.

An option to get the list of roomplans and devices, is using the SQLite command-line shell (see [SQLite Shell](#)) to get the information from the Domoticz database by direct accessing the tables:

- **Database**  
/home/pi/domoticz/domoticz.db
- **Tables**
  - DeviceStatus – get the device name for a given ID (=Idx)
  - Plans – get the roomplans
  - DeviceToPlansMap – get the roomplans with associated devices

The SQLite command-line shell is opened with

```
cd /home/pi/domoticz
sqlite3
```

## Plans

### Get the roomplans

The output is in order of appearance.

```
sqlite> .header on
sqlite> .mode column
sqlite> select * from plans order by "Order";
ID      Order     Name          FloorplanID Area
-----  -----  -----
1       1        $Hidden Devices  0
13      4        Bad           0
14      5        Dusche         0
11      7        Esszimmer (EZ)  0
16      8        Haus          0
2       9        Information    1
17      10       Keller         0
15      11       MakeLab        0
4       12       Termine        0
12      13       Wohnzimmer (WZ)  0
8       17       Hardware Monito  0
5       18       Volumio        3             123,118
```



### Note

Screenshot of the GUI with the dropdown list showing the roomplans.

### Get the roomplans

List the plans with their associated devices. The output format is CSV.

If the device name contains blanks, the name is put between "", i.e. Device Name"

```
cd /home/pi/domoticz
sqlite3

sqlite> .open domoticz.db
sqlite> .header on
sqlite> .mode csv
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan,PlanID,Device,Idx
Bad,13,"Hdg Bad Batterie",209
Bad,13,"Hdg Bad Sollwert",207
Bad,13,"Hdg Bad Temperatur",208
Dusche,14,"Hdg Dusche Batterie",200
Dusche,14,"Hdg Dusche Sollwert",198
Dusche,14,"Hdg Dusche Temperatur",199
"Esszimmer (EZ)",11,"Hdg EZ Batterie",206
"Esszimmer (EZ)",11,"Hdg EZ Sollwert",204
"Esszimmer (EZ)",11,"Hdg EZ Temperatur",205
"Esszimmer (EZ)",11,"Hue EZ",111
... and more ...
```

# RFXCOM RFxTrx433E

## Purpose

The [RFXCOM RFxTrx433E](#) Transceiver is a 433Mhz transmitter and receiver (transceiver) to control or respond to 433Mhz devices (sensors).  
More Product information [here](#) (including successor products).

## Prepare RFxTrx433E

Install the utility programs RFXflash and RFXmngr on the Development Device.

*Note*

For upgrading the firmware, the RFX Flash Programmer is required. Always download the latest version prior flashing.

Flash latest firmware (RFxTrx433\_Ext2\_Firmware) as described in [RFxTrx User Guide](#).

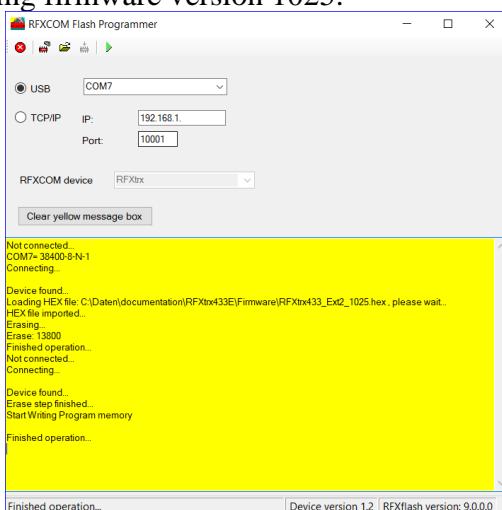
*Note*

Use the same procedure for updating the firmware, i.e.

### Steps to install/upgrade

- Shutdown the Domoticz system (i.e. Raspberry Pi)
- Connect the RFXCOM device to USB port of development device (used COM7 on a notebook)
- Download and unzip the latest firmware
- Download and start the RFXCOM Flash Programmer
- Select the latest hex file
- Write to the device
- Connect the RFXCOM device to the Domoticz system
- Reboot the Domoticz system
- Open Domoticz in a browser and check the log (Setup > Log) and if the hardware device (Setup > Hardware) RFxTrx433e shows the correct version (in this case Version: Ext2/1025)
- In case issues, restart Domoticz from terminal:  
sudo service domoticz.sh restart

Example screenshot flashing firmware version 1025.



Open the RFXmngr, connect to the RFXtrx433E, obtain status information and log incoming messages for devices found.

### Example RFXmngr Log

```
Get Status
-----
Packettype      = Interface Message
subtype        = Interface Response
Sequence nbr   = 1
response on cmnd = Get Status
Transceiver type = 433.92MHz
Firmware version = 1022
Firmware Type   = Ext2
Transmit power   = 10dBm
Hardware version = 1.2
...
```

Depending devices found, incoming **messages** are logged, i.e. for a TFA TS34C (Temperature & Humidity sensor)

```
30.12.2017 13:46:01
Packettype      = TEMP_HUM
subtype        = TH7 - Cresta, TFA TS34C
                  channel 1
Sequence nbr   = 102
ID             = 280E decimal:10254
Temperature     = 17,3 °C
Humidity       = 54
Status          = Comfortable
Signal level    = 5  -80dBm
Battery         = OK
```

#### Note

The RFXmngr is not only used to log connected devices but also to manually configure connected devices, like the Somfy RTS Devices.

## Domoticz Configuration

The hardware device is added / updated via **Settings > Hardware**.

4	RFXtrx433e	Yes	RFXCOM - RFXtrx433 USB 433.92MHz Transceiver Version: Ext2/1022 <a href="#">Set Mode</a>	/dev/ttyUSB-RFX433E-A	Disabled
---	------------	-----	--	-----------------------	----------

*Note*

When updating the firmware, ensure to use the version Ext2/NNNN.

The firmware can be downloaded [here](#).

Enabled:

Name: RFXtrx433e

Type: RFXCOM - RFXtrx433 USB 433.92MHz Transceiver

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
**Do not enable this option for devices that do not receive data!**

Serial Port: /dev/ttyUSB-RFX433E-A

After adding the device, restart Domoticz (Settings > More Options > Restart) and check the Domoticz Log (Settings > Log), i.e.

```
2018-08-30 11:27:51.659 Status: RFXCOM: Using serial port: /dev/ttyUSB-RFX433E-A
2018-08-30 11:27:52.311 subtype = Interface Response
2018-08-30 11:27:52.311 Sequence nbr = 2
2018-08-30 11:27:52.311 response on cmnd = Get Status
2018-08-30 11:27:52.311 Transceiver type = 433.92MHz
2018-08-30 11:27:52.311 Firmware version = 1022
2018-08-30 11:27:52.311 Firmware type = Ext2
2018-08-30 11:27:52.311 Hardware version = 1.2
2018-08-30 11:27:52.311 Undec off
2018-08-30 11:27:52.311 X10 enabled
2018-08-30 11:27:52.311 ARC enabled
2018-08-30 11:27:52.311 AC enabled
2018-08-30 11:27:52.311 HomeEasy EU enabled
2018-08-30 11:27:52.311 Meiantech/Atlantic enabled
2018-08-30 11:27:52.311 Oregon Scientific enabled
2018-08-30 11:27:52.311 ATI/Cartellectronic enabled
2018-08-30 11:27:52.311 Visonic enabled
2018-08-30 11:27:52.311 Mertik enabled
2018-08-30 11:27:52.311 AD enabled
2018-08-30 11:27:52.311 Hideki enabled
2018-08-30 11:27:52.311 La Crosse enabled
2018-08-30 11:27:52.311 Legrand enabled
2018-08-30 11:27:52.311 MSG4Reserved5 enabled
2018-08-30 11:27:52.311 BlindsT0 enabled
2018-08-30 11:27:52.311 BlindsT1 enabled
2018-08-30 11:27:52.311 AE enabled
2018-08-30 11:27:52.311 RUBiCSOn enabled
2018-08-30 11:27:52.311 FineOffset enabled
2018-08-30 11:27:52.311 Lighting4 enabled
2018-08-30 11:27:52.311 Conrad RSL enabled
2018-08-30 11:27:52.311 ByronSX enabled
2018-08-30 11:27:52.311 IMAGINTRONIX enabled
2018-08-30 11:27:52.311 KEELOQ enabled
2018-08-30 11:27:52.311 Home Confort enabled
2018-08-30 11:27:52.402 (RFXtrx433e) Temp + Humidity (Garage)
```

## Troubleshooting

If Domoticz recognizes a new device and the device is added to the device, but the data is not correct or not updated by Domoticz, then check the device data using the RFXmngr.

### Example

A door contact switch PB-67R is recognized by Domoticz as Type: Lighting 5, SubType: Kangtai / Cotech, Data: Off.

If the switch changes state, Domoticz does not update.

Monitoring using the RFXmngr, shows:

```
Packettype = Lighting5
ERROR: Unknown Sub type for Packet type=14: 11
Signal level = 7
```

It confirms that this device is not (status 20190521) supported by RFXtrx433E.

## Packet Analysis

Out of curiosity tried to analyse the content of a packet. The reason behind, is considering using an Arduino Microcontroller with an CC1101 RF Transciever to display data received. [The CC1101 is a low-cost sub-1 GHz transceiver designed for very low-power wireless applications, integrated with a configurable baseband modem supporting various modulation formats and configurable data rate up to 600 kbps. TX power(Max)(dBm): 12, Frequency bands(MHz): 300-348; 387-464; 779-928, Data rate(Max)(kbps): 500]

Looking up the Domoticz source, file “main\RFXtrx.h”, which defines the structure of the various packettypes.

### Example packettype TEMP\_HUM

Data from the RFXCOM Tool “rfxmgr”.

```
29/07/2020 01:38:11:769= 0A520722280E00CD390169
Packettype      = TEMP_HUM
subtype        = TH7 - Cresta, TFA TS34C, channel 1
Sequence nbr   = 34 = HEX 22      (SN)
ID             = 280E decimal:10254 (ID)
Temperature    = 20.5 °C = 205 HEX CD      (TT)
Humidity       = 57 = HEX 39      (HH)
Status          = Comfortable     (SS)
Signal level   = 6   -72dBm = (L)
Battery         = OK (B)
```

PACKET CONTENT  
PL PT ST SN ID1ID2TS TT HH HS L B  
0A 52 07 22 28 0E 00 CD 39 01 6 9

```
struct {
    BYTE packetlength;
    BYTE packettype;
    BYTE subtype;
    BYTE seqnbr;
    BYTE id1;
    BYTE id2;
#ifndef IS_BIG_ENDIAN
    BYTE tempsign : 1;
    BYTE temperatureh : 7;
    BYTE temperaturel;
    BYTE humidity;
    BYTE humidity_status;
    BYTE rssi : 4;
    BYTE battery_level : 4;
#else
    BYTE temperatureh : 7;
    BYTE tempsign : 1;
    BYTE temperaturel;
    BYTE humidity;
    BYTE humidity_status;
    BYTE battery_level : 4;
    BYTE rssi : 4;
#endif
} TEMP_HUM;
```

# Soil Moisture Monitor (Tinkerforge, Plugin)

## Purpose

To measure the Soil Moisture value of a plant, display in Domoticz and near the plant on a 4-Digit LED Display and indicate moisture state on a LED indicator.

## Solution

A Domoticz Python plugin "Soil Moisture Monitor" with a Soil Moisture Monitor device obtaining the Moisture value from a Tinkerforge Moisture Bricklet.

The Tinkerforge Moisture Bricklet is connected to a Tinkerforge Master Brick with WiFi extension.

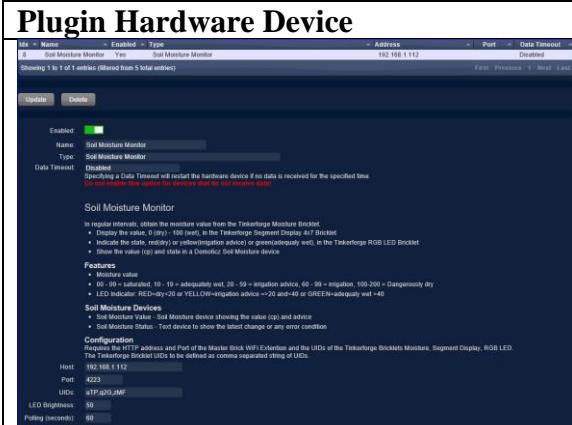
The Moisture value is converted to a range 0 (dry) – 100 (saturated) and displayed in a Tinkerforge Segment Display 4x7.

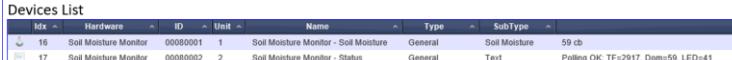
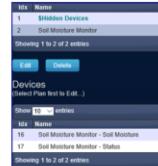
In addition, a Tinkerforge RGB LED bricklet indicates the state RED (dry) or YELLOW (irrigation advice) or GREEN (saturated).

A prototype device is built and tested on a mini palm tree in author's MakeLAB. It is planned to create a weatherproof case, to monitor the Moisture value in garden areas.

More information or get the latest version, go [here](#) (GitHub).

See also [Python Plugin Development](#).

Plugin Hardware Device		Monitoring Mini Palm Tree
		

Domoticz – Hardware & Soil Moisture Device	
	
	

# Stock Quotes (Service, Alpha Vantage)

## Purpose

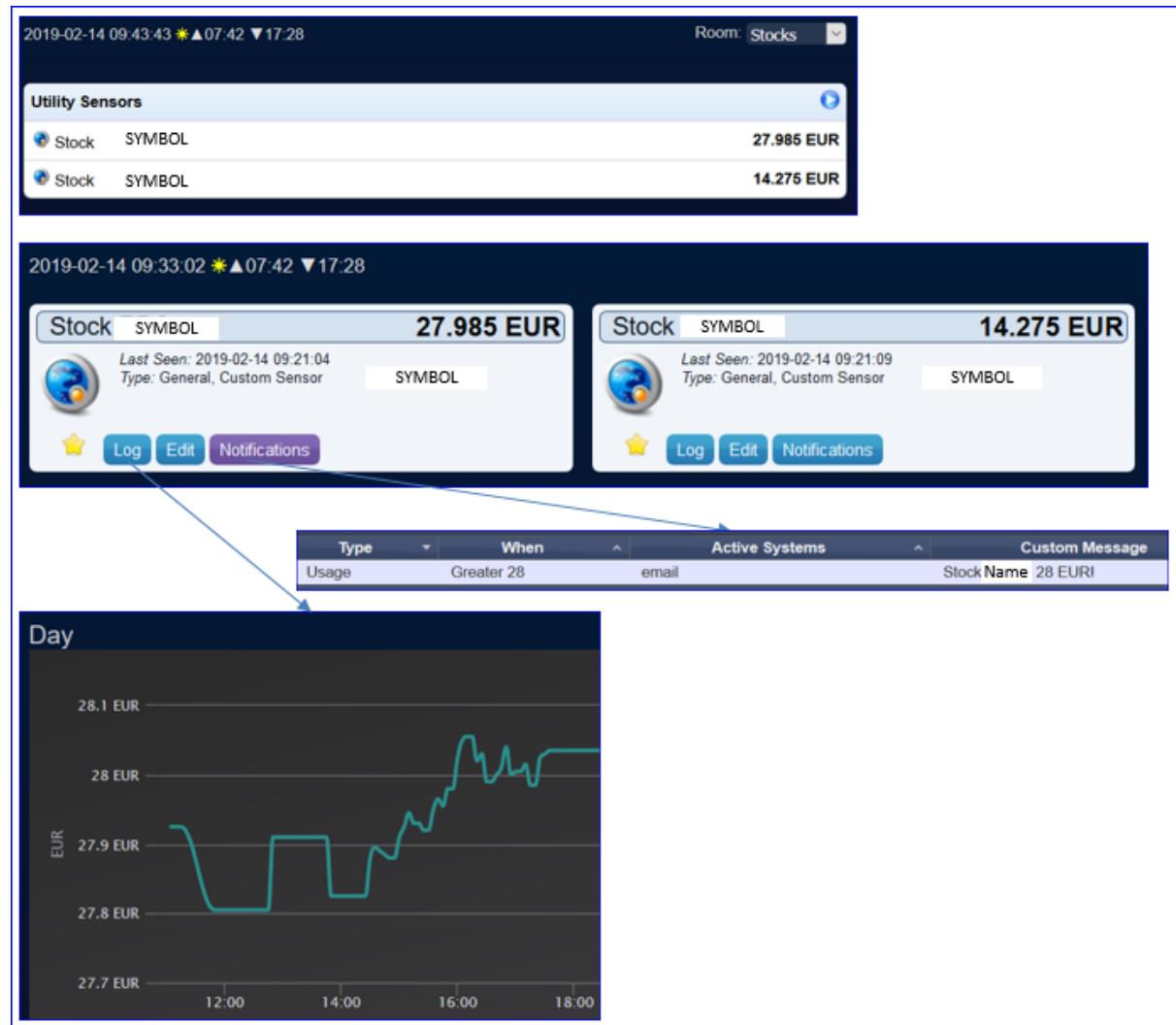
To monitor stock quotes (values):

- Request every 30 minutes stock quotes from [Alpha Vantage](#)  
An Alpha Vantage API key is required to place requests (get [here](#)).
- Display quotes in a dedicated Domoticz dashboard (Room Stocks).
- View trends in charts day, month, year.
- Notify via alert message and (optional) email on thresholds.

### Note

Initially a Node-RED solution has been developed and running for a while. Whilst getting on with scripting, a dzVents solution has been put in place which replaces the Node-RED flows.

## Screenshots Stock Monitor with Dashboard Room Stocks



# Domoticz Configuration

## Devices

Created Virtual Sensors to monitor a Stockprice.

Idx	Name	Type	SubType	Function
152	Stock STOCKA	General	Custom Sensor	Monitor the stock price for STOCKA
153	Stock STOCKB	General	Custom Sensor	Monitor the stock price for STOCKB
	Add more stocks ...			

### Note

The custom sensors data are stored in the Domoticz database tables

- Percentage (DeviceRowID [the device Idx], Percentage [the value], Date) and
- Percentage Calendar (DeviceRowID, Percentage\_Min, Max,\_Value for per day)

SQL Select Statement Example with few Results:

Select * from Percentage_Calendar where DeviceRowID=153;				
DeviceRowID	Percentage_Min	Percentage_Max	Percentage_Avg	Date
153	14.145	14.235	14.1728	2019-02-13
153	14.07	14.275	14.1551	2019-02-14
153	13.975	14.28	14.1697	2019-02-15
153	14.26	14.26	14.26	2019-02-16

## Roomplan

Created a roomplan *Stocks* with the previous defined devices

(Domoticz GUI Setup > More Options > Plans > Roomplan).

The roomplan is used as the Domoticz Dashboard (see earlier screenshot) to monitor the stocks.

### Screenshot Roomplan number 6, named Stocks with 2 devices

Idx	Name	Symbol
152	Stock	SYMBOL
153	Stock	SYMBOL

Add more devices as required.

# Solution Automation Script

The dzVents Lua script event sends a HTTP API request for several symbols to the Alpha Vantage service.

The HTTP response, per symbol, is a JSON string, which is parsed to obtain the symbol and the price. Then the Domoticz stock device value is updated with the price.

## Example HTTP API request with JSON response

```
https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=YOURSYMBOL&apikey=YOURAPIKEY
```

```
{  
    "Global Quote": {  
        "01. symbol": "YOURSYMBOL",  
        "02. open": "25.7500",  
        "03. high": "25.7700",  
        "04. low": "25.7350",  
        "05. price": "25.7700",  
        "06. volume": "5250",  
        "07. latest trading day": "2019-12-05",  
        "08. previous close": "25.6200",  
        "09. change": "0.1500",  
        "10. change percent": "0.5855%"  
    }  
}
```

## Automation Script

The stocks are defined in a JSON string. Each stock has its own JSON definition.

The Lua package JSON.lua (stored in /home/pi/domoticz/scripts/lua/) is required – many thanks to the author.

The JSON key:value pairs for a symbol assigned to a variable:

```
local STOCKA = JSON:decode('{"symbol":"YOURSYMBOL", "idx":YOURDEVICEIDX,  
"response":"SQMRESYOURSYMBOL"}')
```

Ensure the response key contains a unique value. In the script, sequential characters are used starting with A ("SQMRESA").

In the HTTP response, the response string is checked to then select & update the Domoticz device value with the price.

Prior updating the device, a calculation like converting local currency to EUR can be performed.

### Event name: stock\_quotes

```

local SQM_APIKEY    = '*****' -- Your API Key
local SQM_URL        = 'https://www.alphavantage.co/query?function=GLOBAL_QUOTE&apikey=' .. SQM_APIKEY
.. '&symbol='
-- JSON response keys used - case sensitive (see comments above)
local KEYGLOBALQUOTE   = 'Global Quote'
local KEYSYMBOL        = '01. symbol'
local KEYPRICE         = '05. price'
-- Stocks with Symbol, device idx, unique HTTP response
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()
local STOCKA = JSON:decode('{"symbol":"YOURSYMBOL","idx":152,"response":"SQMRESA"}')
-- local STOCKE = JSON:decode('{"symbol":"YOURSYMBOL","idx":NNN,"response":"SQMRESE"}')

return {
on = {
  timer = { 'every 30 minutes' },
  httpResponses = { STOCKA.response, --STOCKE.response, }
},
execute = function(domoticz, item)
  if (item.isTimer) then
    domoticz.openURL({url=SQM_URL .. STOCKA.symbol,method='GET',callback = STOCKA.response,})
    --domoticz.openURL({url=SQM_URL .. STOCKE.symbol,method='GET',callback = STOCKE.response,})
  end

  if (item.isHTTPResponse) then
    -- Checking statuscode, callback. Item is type json (item.isJSON)
    if (item.statusCode == 200) then
      local symbol = item.json[KEYGLOBALQUOTE][KEYSYMBOL]
      local price = item.json[KEYGLOBALQUOTE][KEYPRICE]
      domoticz.log(symbol .. ' = ' .. price)
      if (item.callback == STOCKA.response) then
        domoticz.devices(STOCKA.idx).updateCustomSensor(price)
      end
      --if (item.callback == STOCKE.response) then
      --  domoticz.devices(STOCKE.idx).updateCustomSensor(price)
      --end
      return
    else
      domoticz.log('[ERROR]' .. item.statusText, domoticz.LOG_ERROR)
      return
    end
  end
end
end
}
}

```

### Domoticz Log

Single stock update.

```

2019-12-05 14:15:03.461 Status: dzVents: Info: Handling httpResponse-events for: "SQMRESA"
2019-12-05 14:15:03.461 Status: dzVents: Info: ----- Start internal script: stock_quote_monitor:
HTTPResponse: "SQMRESA"
2019-12-05 14:15:03.466 Status: dzVents: Info: SQM Status Code = 200, Name=SQMRESA
2019-12-05 14:15:03.466 Status: dzVents: Info: SQM Updating SQMRESA
2019-12-05 14:15:03.466 Status: dzVents: Info: YOURSYMBOL = 16.7900
2019-12-05 14:15:03.482 Status: dzVents: Info: ----- Finished stock_quote_monitor

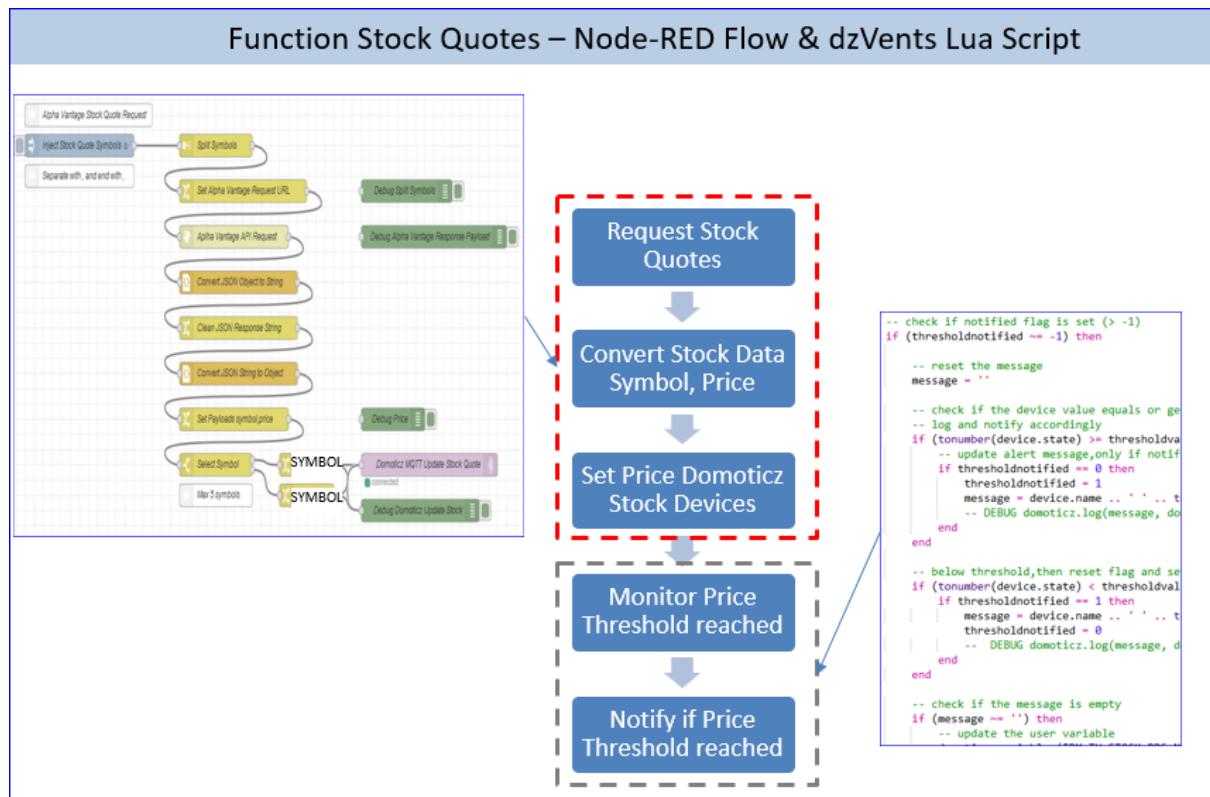
```

## Solution Node-RED

NOT USED – replaced by the previous described solution “Automation Script”.

Outlined is a Node-RED solution for two XETRA stock quotes.

The stock quote information is requested, every 30 minutes, via an Alpha Vantage API request (URL), which returns a JSON string with price information etc. The symbol and price are extracted and send to the respective Domoticz device.



### Example API Request & Response

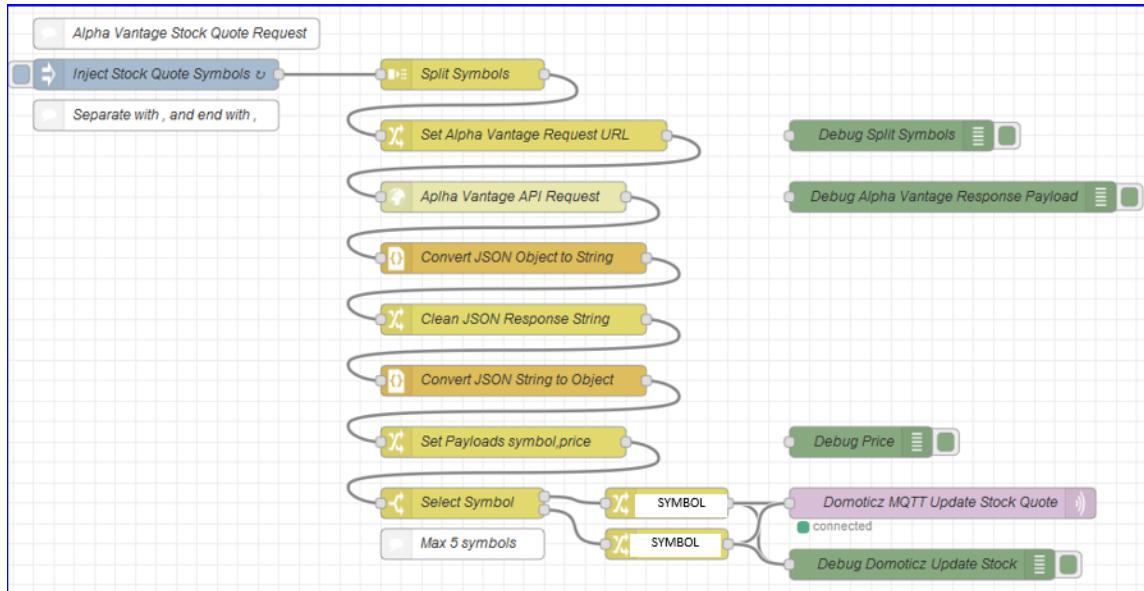
[https://www.alphavantage.co/query?function=GLOBAL\\_QUOTE&symbol=STOCKA&apikey=YOURAPIKEY](https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=STOCKA&apikey=YOURAPIKEY)

### JSON Response

```
{
  "Global Quote": {
    "01. symbol": "STOCKA",
    "02. open": "0.0000",
    "03. high": "0.0000",
    "04. low": "0.0000",
    "05. price": "16.0550",
    "06. volume": "0",
    "07. latest trading day": "2019-02-12",
    "08. previous close": "15.9000",
    "09. change": "0.1550",
    "10. change percent": "0.9748%"
  }
}
```

## Node-RED Flow

The flow is used to define the Stock Quote Symbols, obtain the data from Alpha Vantage, extract the price, and update the Domoticz devices.



### Inject Node

The stocks to monitor are defined in the Inject Node, with their symbol as a comma separated string (last symbol MUST end with a comma).

Example for two stocks:

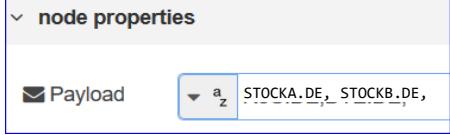
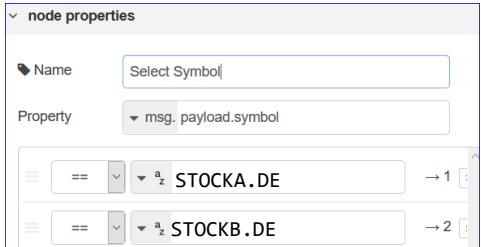
```
STOCKA.DE, STOCKB.DE,
```

### Select Symbol Node

The Select Symbol switch node splits the symbols, which require each a change node to define the Domoticz MQTT update command.

## Add, change, or remove symbols

Change following nodes in the flow:

Inject Node	Property Payload String
	 <pre> &lt;node properties&gt;   &lt;Payload&gt; STOCKA.DE, STOCKB.DE,   &lt;/Payload&gt; </pre>
Select Symbol Node	Properties
	 <pre> &lt;node properties&gt;   &lt;Name&gt; Select Symbol   &lt;Property&gt; msg.payload.symbol   &lt;Condition&gt; == STOCKA.DE   &lt;Condition&gt; == STOCKB.DE   &lt;/Condition&gt; </pre>
Change Node	Properties for each of the Change Nodes for a Stock
	 <pre> &lt;node properties&gt;   &lt;Name&gt; STOCKA.DE   &lt;Rules&gt;     &lt;Rule&gt; Set msg.payload to {"command": "udevice", "idx": 152, "nvalue": 0, "svalue": payload.price}   &lt;/Rules&gt; </pre>

## Domoticz Log

The devices are updated.

```

2019-02-19 09:41:52.053 MQTT: Topic: domoticz/in, Message:
{"command":"udevice","idx":152,"nvalue":0,"svalue":"27.6300"}
2019-02-19 09:41:52.259 MQTT: Topic: domoticz/in, Message:
{"command":"udevice","idx":153,"nvalue":0,"svalue":"14.5100"}

```

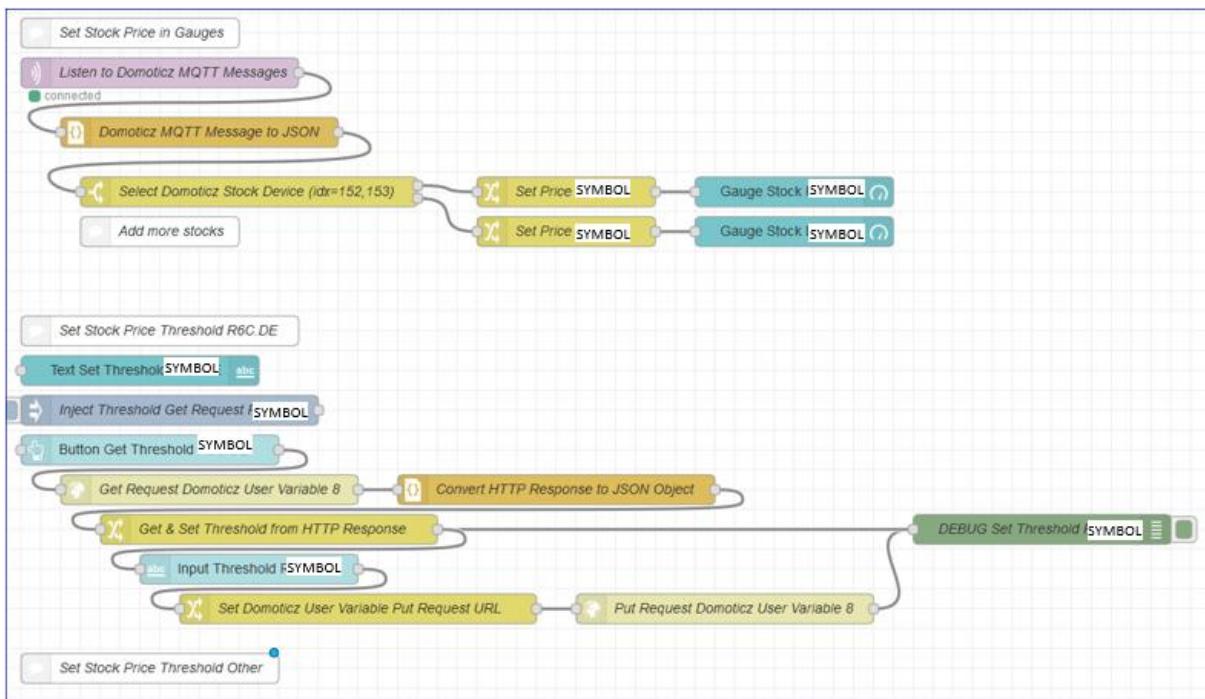
## Node-RED Dashboard UI

A Node-RED Dashboard with UI nodes can be used to monitor price, thresholds and setting thresholds.

The dashboard displays two stock price gauges for EUR. The first gauge shows a value of 27.74 with a threshold set at 28.01. The second gauge shows a value of 14.38. A 'Set Threshold' input field is present below the first gauge, and a 'REFRESH' button is at the bottom.

**Two stocks, one with just a gauge to show the value, the other with the option to change the threshold.**

This solution can be enhanced, to f.e. set the threshold message or show simple trend in a graph.



## Notifications

Optional feature to notify incase a stock price is above or below a threshold.

### Option Widget

Notify by using the device configuration to set the threshold for a stock and notify via email.

Type	When	Active Systems	Custom Message
Usage	Greater 28	email	Stock Name 28 EUR!

To change the threshold, modify the rule.

### Option dzVents

Notify by using User Variables and a dzVents script to monitor the threshold. If a threshold is reached or is below the threshold, send a notification and update the alert message.

### User Variables

The User Variables are defined in the Domoticz GUI (Setup >More Options > User variables).

Idx	Variable name	Variable type	Current value
9	TH_STOCK_SYMBOL_NOTIFIED	Integer	1
8	TH_STOCK_SYMBOL	Float	27

Name uses syntax: PREFIX\_TYPE\_SYMBOL\_NAME

The TH\_STOCK\_SYMBOL sets the value of the threshold for the stock.

The TH\_STOCK\_SYMBOL\_NOTIFIED is used to determine if notification should be used (0), if not notified (0) or if notified (1).

The values 0 and 1 are to avoid sending out new messages when the stock value is checked against its threshold. These values are set via the dzVents script "stock\_monitor".

Changing the value of the threshold can be done via Domoticz GUI changing the User Variable or as an alternative via Node-RED Dashboard UI Input Node.

## Automation Script

Event name: stock\_monitor

```
-- External modules:
local utils = require('utils')
local msgbox = require('msgbox')
-- Stock NAME - idx device, idx uservariable threshold, idx uservariable thresholdnotified
local IDX_STOCK_NAME = 152
local IDX_TH_STOCK_NAME = 8
local IDX_TH_STOCK_NAME_NOTIFIED = 9
-- temp var for the threshold value
local thresholdvalue
-- flag to check if notified, to avoid notifying for every change above threshold
local thresholdnotified = 0
-- message
local message

return {
  on = {
    devices = {
      IDX_STOCK_NAME
    }
  },
  execute = function(domoticz, device)
    domoticz.log('Device '..device.name..' was changed to '..device.state, domoticz.LOG_INFO)

    -- select the device to obtain the thresholdvalue
    if device.idx == IDX_STOCK_NAME then
      thresholdvalue = domoticz.variables(IDX_TH_STOCK_NAME).value
      thresholdnotified = domoticz.variables(IDX_TH_STOCK_NAME_NOTIFIED).value
      domoticz.log('Device '..device.name..': '..tostring(thresholdvalue)..', '..
      tostring(thresholdnotified), domoticz.LOG_INFO)
    end;
    -- add more devices
    -- check if notified flag is set (> -1)
    if (thresholdnotified ~= -1) then
      -- reset the message
      message = ''
      -- check if the device value equals or geater threshold (user_variable)
      -- log and notify accordingly
      if (tonumber(device.state) >= thresholdvalue) then
        -- update alert message,only if notifiedflag = 0 to avoid duplication
        if thresholdnotified == 0 then
          thresholdnotified = 1
          message = device.name .. ' ' .. tonumber(device.state) .. ' reached threshold ' ..
          tostring(thresholdvalue)
          -- DEBUG domoticz.log(message, domoticz.LOG_INFO)
        end
      end
      -- below threshold,then reset flag and set message
      if (tonumber(device.state) < thresholdvalue) then
        if thresholdnotified == 1 then
          message = device.name .. ' ' .. tonumber(device.state) .. ' below threshold ' ..
          tostring(thresholdvalue)
          thresholdnotified = 0
          -- DEBUG domoticz.log(message, domoticz.LOG_INFO)
        end
      end
      -- check if the message is empty
      if (message ~= '') then
        -- update the user variable
        domoticz.variables(IDX_TH_STOCK_RDS_NOTIFIED).set(thresholdnotified)
        -- write to log
        domoticz.log(message, domoticz.LOG_INFO)
        -- set the alert message
        msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_ORANGE, message)
        -- and notification
        -- domoticz.notify(message)
      end
    end
  end
}
```

# Enhancements

## Total Stock Value

Beside showing the current stock value, it is also interesting to show, with trend, the total value of shares in stock.

### Solution

#### User Variable

Define User Variable holding the total number of shares in stock

Name: DEF\_STOCK\_NAME\_COUNT

Type: Integer

Set the value

Note the idx: 12

#### Create Custom Sensor

Name: Stock Name Total

Axis Label: EUR

Note the idx: 156

#### dzVents Script

Enhance the dzVents script stock\_monitor with two local vars for the idx of the totalvalue device and number of shares user variable

```
local IDX_STOCK_NAME = 152          -- price
local IDX_TH_STOCK_NAME = 8          -- threshold
local IDX_STOCK_NAME_TOTAL = 156     -- total value all shares
local IDX_DEF_STOCK_NAME_COUNT = 12  -- number of shares

...
-- select the device to obtain the thresholdvalue
if device.idx == IDX_STOCK_NAME then
    -- update total value custom sensor
    stockvalue = math.floor(domoticz.variables(IDX_DEF_STOCK_NAME_COUNT).value * tonumber(device.state))
    domoticz.log('Device ' .. device.name .. ' Stock Value: ' .. tostring(stockvalue), domoticz.LOG_INFO)
    domoticz.devices(IDX_STOCK_NAME_TOTAL).updateCustomSensor(stockvalue)
end
-- add more devices
...
```

## Hints

### Search Stock Quote Symbol

Examples search for the Powercell stock quote symbol.

```
https://www.alphavantage.co/query?function=SYMBOL_SEARCH&keywords=POWERCELL&apikey=YOURAPIKEY
```

#### Result

```
{
  "bestMatches": [
    {
      "1. symbol": "PCELF",
      "2. name": "PowerCell Sweden AB (publ)",
      "3. type": "Equity",
      "4. region": "United States",
      "5. marketOpen": "09:30",
      "6. marketClose": "16:00",
      "7. timezone": "UTC-05",
      "8. currency": "USD",
      "9. matchScore": "0.5714"
    },
    {
      "1. symbol": "27W.FRK",
      "2. name": "PowerCell Sweden AB (publ)",
      "3. type": "Equity",
      "4. region": "Frankfurt",
      "5. marketOpen": "08:00",
      "6. marketClose": "20:00",
      "7. timezone": "UTC+02",
      "8. currency": "EUR",
      "9. matchScore": "0.5143"
    }
  ]
}
```

# Temperature & Humidity (TFA TS34C)

## Purpose

To measure the temperature (°C) & humidity (%RH) in various rooms.

## Device

Device (433MHz): Temperature TFA Dostmann / Wertheim TS34C.

The devices are connected to the RFXCOM RFXtrx433E.

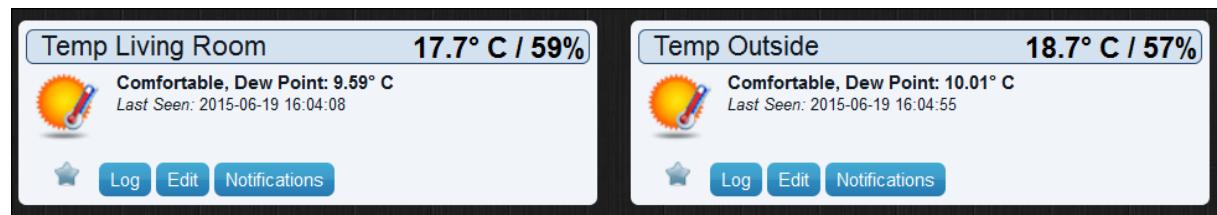
## Setup

1. Open the device battery flip and set the channel.  
Start with 1.
2. Take out the battery blocker
3. The device will be recognized automatically and listed under the devices list
4. Select the green arrow to give the device the name and the name will be shown in the list (see below)

## Device List Entries (IDX=14,11)

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Tail	Last Seen
14	RFXCom	450E	2	Temp Outside	Temp + Humidity	Cresta, TFA TS34C	19.0 C, 55 %	4 100	2015-06-19 15:55:10
11	RFXCom	2A0E	1	Temp Living Room	Temp + Humidity	Cresta, TFA TS34C	17.6 C, 59 %	6 100	2015-06-19 15:54:06
6	GPIO Port	18		LED GPIO18	Lighting 1	Impuls	Off	7 -	2015-06-19 15:20:16
12	RFXCom	1F1F1F	1	Blind Living Room	RFY	RFY	Stopped	7 -	2015-06-19 15:18:55
13	RFXCom	1E1E1E	2	Blind Bed Room	RFY	RFY	Stopped	7 -	2015-06-19 15:06:34

## Temperature & Humidity Widgets



## Replace Batteries

The battery level is shown in the device attributes.

Two examples (different devices) with battery level 100% and 0%.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Battery
356	RFXtrx433e	270E	1	Keller Großraum	Temp + Humidity	Cresta, TFA TS34C	19.3 C, 59 %	5

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Battery
28	RFXtrx433e	9D16	0	Windmesser	Wind	TFA	236.00;SW;3;0;12.4;12.4	4

If the battery level is low, then replace the two batteries.

If these are replaced, Domoticz recognizes the current device as a new device (with a new ID line 270E) and the old device (with ID 2A0E) is not updated anymore.

This means all historical data are lost.

To keep the historical month & year data, use the command tool [sqlite3](#).

### Example

Idx Old Device = 11; Idx New Device = 356.

#### Steps

1. Open CLI
2. Stop Domoticz and set domoticz folder
3. Start sqlite3 and open the domoticz database
4. Run sql statement to update the devicerowid from 11 to 356
5. Run sql query to check if records for devicerowid 356
6. Start Domoticz
7. Open the GUI and check the log of device with idx 356
8. Delete old device with IDX=11

```
$ cd domoticz
$ sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open domoticz.db
sqlite> UPDATE Temperature SET DeviceRowID = 356 WHERE DeviceRowID = 11;
sqlite> UPDATE Temperature_Calendar SET DeviceRowID = 356 WHERE DeviceRowID = 11;
sqlite> SELECT * FROM Temperature_Calendar WHERE DeviceRowID = 356;
356|17.5|20.0|18.63|0.0|0.0|61|0|8.98|0.0|0.0|0.0|2015-06-19
356|17.6|20.7|18.67|0.0|0.0|62|0|9.71|0.0|0.0|0.0|2015-06-20
... and more
sqlite> .exit
```

# Time Control

## Purpose

- To track & control the time spent in hours, per activity block (with start- & end-time) and total/day, on a generic task.
- To set a threshold and notify if the time spent reached daily limit.

### Note

The author wanted to get a grip on Make activities time spent with a limit of 2.5 to 4 hours/day.

## Screenshots devices, dashboard, and a simple graph

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
106	VirtualDevices	00082106	1	Time Control Status	General	Text	18:21 Update: Block=0.32, Day=0.60, Limit=2.50 hrs
102	VirtualDevices	000140B6	1	Time Control Today	Light/Switch	Switch	On
101	VirtualDevices	000140B5	1	Time Control	Light/Switch	Switch	On

Time Control running (On) with status update.

Time Control stopped (Off) with status showing all todays activity blocks with summary. Triggered by switch Time Control Today.

Idx	Variable name	Variable type	Current value
2	TH_TIMECONTROL	Float	2.5

## Solution

This solution uses a

- **Switch device** ("Time Control") - start & stop time tracking
- **Text device** ("Time Control Status") - provide information on time spent per activity block and day
- **Switch device** ("Time Control Today") - enables to show todays activity blocks in the text device "Time Control Status".
- **User Variable** ("TH\_TIMECONTROL") – set the threshold of max hours/day (float) which is checked in the dzVents Lua Automation Script "time\_control.lua", function "checkLimit".
- **dzVents Lua Automation Script**: time\_control.lua – start, update, stop with status update.
- **dzVents Lua Automation Script**: time\_control\_today.lua – display daily summary.
- **SQL-Command-File** – select & report the daily activity blocks with summary.

The time between tracking start & end is defined as a block. A day has one or more blocks (or empty if no action).

The switch device bar chart log provides a nice overview on the time spent.

A dzVents Lua Automation Script runs every minute and updates the text device with the time spent on the current block, day total and daily limit in hours.

The dailay limit can be set via a user variable. If not defined or value=0, the daily limit is not used.

## Domoticz Configuration

### Devices

- Type: Light/Switch, SubType:Switch, SwitchType: On/Off, Name: Time Control, Idx: 101, Add to dashboard.
- Type: General, SubType:Text, Name: Time Control Status, Idx: 106, Add to dashboard.
- Type: Light/Switch, SubType:Switch, SwitchType: On/Off, Name: Time Control Today, Idx: 102, Add to dashboard.

### User Variable

- Variable name: TH\_TIMECONTROL, Variable type: Float, Variable value: 2.5

### Dashboard & Log

The three devices are added to the Domoticz dashboard or to a dedicated roomplan.

The switch "Time Control" is used to start & stop time tracking and the text device provides information when the actitity starts, updates, or stops. The update interval is every minute.

Example information:

```
10:52 Stopped: Block=1.43, Day=1.43, Limit=2.50 hrs
11:03 Started: Block=0.00, Day=1.43, Limit=2.50 hrs
11:05 Update: Block=0.03, Day=1.47, Limit=2.50 hrs
```

*Note*

There are 2 digits used to display the values. This can be changed in the dzVents Lua script "time\_control.lua".

To view the time spent, select GUI > Switches > Widget "Time Control" and click Log.

- "Today" shows, a bar chart, state On or Off (Y-axis) and time On or Off (X-axis).
- "Text log" lists the state (Data) with time (Date) and the User.

## Automation Events

There are two dzVents Lua automation events:

- **Time Control**: runs every minute and updates the text device "Timer Control Status". At midnight, the daily information is resetted.  
(time\_control.lua)
- **Time Control Today**: runs when triggered by switch "Time Control Today" and updates the text device "Timer Control Status" with daily summary. The data is selected using sqlite command-line with an external file "time-control-today.sql" (located in folder "/home/pi/domoticz/scripts/dzVents/scripts/" containing the SQL statements).  
(time\_control\_today.lua)

### Source: time\_control.lua

```
-- Idx of the devices used
IDXTRACK = 101          -- switch on/off
IDXSTATUS = 106          -- text status
IDXTHTIMECONTROL = 2     -- user var threshold time control (TH_TIMECONTROL)

-- The value to increment on every timer update: minutes/60 to get an hour.
-- Examples: timer every minute: 1/60, every 5 minutes = 5/60
INCREMENTVALUE = (1/60)

function roundNumber(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

function roundsValue(svalue)
    return roundNumber(tonumber(svalue),2)
end

function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

function isnowhhmm(domoticz)
    local timearray = splitstring(domoticz.time.rawTime, ':')
    return timearray[1] .. ':' .. timearray[2]
end

function checkLimit(domoticz)
    local msg
    -- get the latest value of the user var because it might have changed
    domoticz.data.daylimit = domoticz.variables(IDXTHTIMECONTROL).value
    if (domoticz.data.daylimit == 0) or
       (domoticz.variables(IDXTHTIMECONTROL).value == nil) then
        return
    end
    -- check if dayvalue beyond daily limit and if not already notified
```

```

if (domoticz.data.dayvalue > domoticz.data.daylimit) and
(domoticz.data.notified == 0) then
    msg = string.format('%s: Day=%.4f, Daily threshold reached %.2f hrs',
        isnowhhmm(domoticz),
        domoticz.data.dayvalue,
        domoticz.data.daylimit)
    domoticz.notify('Time Control', msg, domoticz.PRIORITY_LOW)
    domoticz.log(msg, domoticz.LOG_INFO)
    domoticz.data.notified = 1
    return
end
-- check daylimit increased, i.e made a change whilst already notified. if so, reset notified
if (domoticz.data.dayvalue < domoticz.data.daylimit) and
(domoticz.data.notified == 1) then
    domoticz.data.notified = 0
    msg = string.format('%s: Day=%.4f, Daily threshold reset %.2f hrs',
        isnowhhmm(domoticz),
        domoticz.data.dayvalue,
        domoticz.data.daylimit)
    domoticz.log(msg, domoticz.LOG_INFO)
end
end

function updateStatus(domoticz, action)
    checkLimit(domoticz)
    local msg = string.format('%s %s: Block=%.2f, Day=%.2f, Limit=%.2f hrs',
        isnowhhmm(domoticz),
        action,
        domoticz.data.blockvalue,
        domoticz.data.dayvalue,
        domoticz.data.daylimit)
    domoticz.devices(IDXSTATUS).updateText(msg)
    domoticz.log(msg, domoticz.LOG_INFO)
end

function updateData(domoticz)
    domoticz.data.blockvalue = domoticz.data.blockvalue + INCREMENTVALUE
    domoticz.data.dayvalue = domoticz.data.dayvalue + INCREMENTVALUE
    updateStatus(domoticz, 'Update')
end

function setTrack(domoticz)
    if (domoticz.devices(IDXTRACK).state == 'Off') then
        updateStatus(domoticz, 'Stopped')
    else
        domoticz.data.blockvalue = 0
        updateStatus(domoticz, 'Started')
    end
end

function resetDay(domoticz)
    domoticz.data.blockvalue = 0
    updateStatus(domoticz, 'Day Reset')
    domoticz.data.dayvalue = 0
    domoticz.data.notified = 0
end

return {
    -- handle timer and switch device change on or off
    on = {
        timer = { 'every minute', 'at 00:15' },
        devices = { IDXTRACK }
    },
    data = {
        blockvalue = { initial = 0 },
        dayvalue = { initial = 0 },
        daylimit = { initial = 0 },
        notified = { initial = 0 }
    },
    execute = function(domoticz, item)
        if (item.isTimer) and
            (domoticz.time.matchesRule('every minute')) and
            (domoticz.devices(IDXTRACK).state == 'On') then

```

```

        updateData(domoticz)
        return
    end
    if (item.isTimer) and
        (domoticz.time.matchesRule('at 00:15')) then
        resetDay(domoticz)
        return
    end
    if (item.isDevice) and
        (item.idx == IDXTRACK) then
        setTrack(domoticz)
        return
    end
end
}

```

### Source: time\_control\_today.lua

```

IDXSWITCH = 102
IDXSTATUS = 106
SQLCMD = "sqlite3 /home/pi/domoticz/domoticz.db '.read /home/pi/domoticz/scripts/dzVents/scripts/time-control-today.sql'

function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

local function osExecute(cmd)
    local fileHandle      = assert(io.popen(cmd, 'r'))
    local commandOutput   = assert(fileHandle:read('*a'))
    local returnTable     = {fileHandle:close()}
    -- rc[3] contains returnCode
    return commandOutput,returnTable[3]
end

return {
    on = {devices = { IDXSWITCH },},
    execute = function(domoticz, item)
        if (item.isDevice) then
            if (domoticz.devices(IDXSWITCH).state == "On") then
                output,rt = osExecute(SQLCMD)
                -- local handle = os.execute(sqlcmd)
                rawdate = domoticz.time.rawDate .. " "
                -- add the date to the output
                output = string.format("%s\n%s",rawdate,output)
                domoticz.log(output, domoticz.LOG_INFO)
                -- update the status device
                domoticz.devices(IDXSTATUS).updateText(output)
                -- Returncode: 0=OK
                domoticz.log(tostring(rt), domoticz.LOG_INFO)
                -- convert the output to a table. Use #outputtable to get the number of entries.
                outputtable = splitstring(output, "\n")
                domoticz.log(outputtable, domoticz.LOG_INFO)
            end
        end
    end
}

```

### Source: time\_control\_today.sql

```

.mode list
.separator " - "
.headers on
-- List the blocks
WITH rows AS
( SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM [lightinglog] WHERE [devicerowid] = 101 AND [date] >= date('now'))
SELECT strftime('%H:%M', startdata.date) AS Start, strftime('%H:%M', enddata.date) AS End,
       ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
  FROM rows startdata
 JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1

```

```

WHERE startdata.nValue = 1;

-- Show the total
WITH rows AS
  (SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
   FROM [lightinglog] WHERE [devicerowid] = 101 AND [date] >= date('now'))
SELECT
  ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),3 ) AS Total
FROM rows startdata
JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1
WHERE startdata.nValue = 1;

```

## Various

### Cleanup (Interactive)

Delete all time control device entries from the tables LightingLog usig SQLite3 command line.

```

cd /home/pi/domoticz
sudo service domoticz.sh stop
sqlite3
sqlite> .open domoticz.db
sqlite> delete from [LightingLog] where [DeviceRowID] IN (101,102,106);
sqlite> select * from [Percentage] where [DeviceRowID] IN (101,102,106);
sqlite> .quit

```

### Clear Device Log (Lua)

To clear a device log via dzVents Lua Script, use

```

IDXSTATUS = 106      -- text status
domoticz.openURL('http://localhost:8080/json.htm?type=command&param=clearlightlog&idx=' .. IDXSTATUS)

```

### Clear Device Time Control Status (HTTP API Request)

To clear the text, set the property svalue empty.

```

http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=106&nvalue=0&svalue=
{"status" : "OK","title" : "Update Device"}

```

# Timers

## Purpose

To perform, in regular intervals, a task or tasks.

Timers can be defined as an Automation Event (all devices) or as a Device Timer (which is only possible for the device type Switches).

### *Note*

Started off by defining timers for all devices by using dzVents.

In the meantime, more and more all kind of Switches have been added which perform “On Time” tasks.

## Rule

Dediced as a rule to use

- **Device Timer** type “On Time” – for a single action, i.e. switch a device On/Off, change setpoint.
- **Automation Events** dzVents - for complex tasks with multiple actions.

## Device Timers

The device timers are device specific for Switches only.

These are created via the GUI > Device widget > option Timers.

Enables to perform simple tasks i.e. On | Off, change setpoint value etc.

### **Types supported (Type,SubType,SwitchType)**

(not the full list, these are the ones used for this workbook):

- Thermostat,SetPoint
- Light/Switch,Switch,Dimmer
- Light/Switch,Switch,Push On Button
- Light/Switch,Switch,Push Off Button
- RFY,RFY,Blinds
- Lighting 1, ELRO AB400,On/Off
- Lighting 2, AC,On/Off
- Lighting 4, PT2262,Contact
- Color Switch,WW,Dimmer
- More...

## Setpoint Timers

Each radiator thermostat device (type: Thermostat, SubType: SetPoint) has one or more timers to control the temperature.

### Setpoint Timers Example

#### Device

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
207	Bath Thermostat	00160001	1	Hdg Bad Sollwert	Thermostat	SetPoint	20.0

#### Timers

Name: Hdg Bad Sollwert				
2019-12-15 19:27:34 ⏪ 08:31 ⏴ 16:01				
Show 25	entries	Search:		
Active	Type	Date	Time	Command Days
Yes	On Time		06:00	Temperature, 20 Everyday
Yes	On Time		11:00	Temperature, 19 Everyday
Yes	On Time		23:00	Temperature, 17 Everyday

### Setpoint Timers Overview

To get an overview of the setpoint device timers defined, perform an SQL SELECT query on the Domoticz Database. The timers are defined in the table SetpointTimers. The device name is taken from field Name of table DeviceStatus

#### Example

SQL Select statement to obtain all setpoint timers from the Domoticz Production database. The device names are in German.

The tables used are DeviceStatus to get the device Name and SetPointTimers to get the timer settings for the device.

The field DeviceRowID is the idx of the device.

```
SELECT d.Name, t.*
FROM DeviceStatus d, SetpointTimers t
WHERE d.ID=t.DeviceRowID
```

Name	ID	Active	DeviceRowID	Date	Time	Type	Temperature	TimerPlan	Days	Month	MDay	Occurence
Hdg Bad Sollwert	1	1	207	2019-12-15	06:00	2	20.0	0	128	0	0	0
Hdg Bad Sollwert	2	1	207	2019-12-15	11:00	2	19.0	0	128	0	0	0
Hdg Bad Sollwert	3	1	207	2019-12-14	23:00	2	17.0	0	128	0	0	0
Hdg EZ Sollwert	4	1	204	2019-12-14	06:00	2	20.0	0	128	0	0	0
Hdg EZ Sollwert	5	1	204	2019-12-14	22:00	2	17.0	0	128	0	0	0
Hdg WZ Sollwert	6	1	201	2019-12-15	06:15	2	21.0	0	128	0	0	0
Hdg WZ Sollwert	7	1	201	2019-12-14	22:00	2	0.0	0	128	0	0	0
Hdg Dusche Sollwert	8	1	198	2019-12-14	21:30	2	21.0	0	128	0	0	0
Hdg Dusche Sollwert	9	1	198	2019-12-14	23:00	2	0.0	0	128	0	0	0
Hdg MakeLab Sollwert	10	1	195	2019-12-14	19:00	2	0.0	0	128	0	0	0

## Switch Timers

Example Switch Device (IDX=30). Type: Lighting 2, AC, On/Off



Active	Type	Date	Time	Randomness	Command	Days
No	On Time		00:00	No	On	Everyday
No	On Time		00:05	No	Off	Everyday

### SQL SELECT on table Timers

```
SELECT d.Name, t.*
FROM DeviceStatus d, Timers t
WHERE d.ID=t.DeviceRowID
```

Name	ID	Active	DeviceRowID	Date	Time	Type	Cmd	Level	Hue	UseRandomness	TimerPlan	Days	Month	MDay	Occurence	Color
TimerSwitch	1	0	30	2018-09-04	00:00	2	0	100	0	0	0	128	0	0	0	
TimerSwitch	2	0	30	2018-09-04	00:05	2	1	100	0	0	0	128	0	0	0	

*Note*

The timers are not active in this case.

<TODO>Clarify why field Date has content although the timer definitions do not set a Date but Time only.

<TODO>Add more device timers. Replace Automation scripts which use only one device.

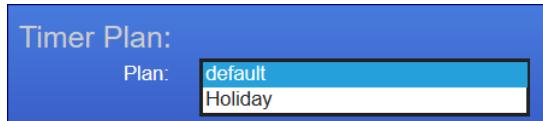
<TODO>Explore if there are any other tables where times are defined.

## Timerplans

In the Domoticz settings, several timer plans can be defined.

Initially there is one timer plan: default (with ID=0). This plan is assigned to the timers defined. See previous SQL SELECT query result, field TimerPlan, which has value 0.

In the GUI > Setup > Settings > Other: the timer plan can be set.



The selected timer plan ID, can be found in the Domoticz database, table Preferences , field Key = ActiveTimerPlan, nValue = 0, sValue = "" (empty).

```
SELECT *  
FROM Preferences  
WHERE Key="ActiveTimerPlan"
```

Key	nValue	sValue
ActiveTimerPlan	0	

A screenshot of a MySQL command-line interface. It shows a single query: "SELECT \* FROM Preferences WHERE Key='ActiveTimerPlan'". Below the query is the result of the execution, which is a table with three columns: Key, nValue, and sValue. There is one row with the values: ActiveTimerPlan, 0, and an empty string.

The timer plans are defined in GUI > Setup > More Options > Plans > Timerplan.

Idx	Name
0	default
1	Holiday

Showing 1 to 2 of 2 entries

First | Previous | 1 | Next | Last

Edit | Duplicate | Delete

A screenshot of the Domoticz software interface showing a table titled "Plans". The table has two rows: one for "default" (Idx 0) and one for "Holiday" (Idx 1). The table includes columns for "Idx" and "Name". At the bottom of the table, there are buttons for "Edit", "Duplicate", and "Delete". Above the table, there are navigation links: "First", "Previous", "1", "Next", and "Last". Below the table, it says "Showing 1 to 2 of 2 entries".

## Automation Events

The events are created with the Domoticz Event Editor using dzVents scripts. The usage of dzVents scripts enables to create complex tasks for all devices.

Below some examples. More Automation events with timers can be found in functions like Coffee Machine Monitor, Days-To-Go, Stock Quotes, Waste Calendar.

### Example Daylight Info

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
121	VirtualSensors	00082121	1	Tageslicht (Std)	General	Text	10-09 06:46 - 19:49 (13:03)



```
--[[[
    daylight_info.dzvents
    Display the day-month sunrise and sunset time plus daylength (hours) in a virtual text sensor:
    10-09 06:32 - 20:08 (13:36)
    Updated once a day.
    20200909 rwbl
]]]

-- Domoticz devices
-- Idx, Hardware, Name, Type, SubType, Data
-- 121, VirtualSensors, Tageslicht (Std), General, Text, 10-09 06:31 - 20:13 (13:42)
local IDX_DAYLIGHTINFO = 121

return {
    on = {
        timer = {
            'at 01:00'
            -- 'every minute' -- TESTS
        }
    },
    execute = function(domoticz, timer)
        -- Calculate the daylength
        local daylength = domoticz.time.sunsetInMinutes - domoticz.time.sunriseInMinutes
        -- Define the text msg: HH:MM -- HH:MM (HH:MM)
        local msg = string.format("%s %s - %s (%s)",
            domoticz.helpers.isnowdatedddmm(domoticz),
            domoticz.helpers.minutestoclock(domoticz.time.sunriseInMinutes),
            domoticz.helpers.minutestoclock(domoticz.time.sunsetInMinutes),
            domoticz.helpers.minutestoclock(daylength))
        -- Update the device
        domoticz.devices(IDX_DAYLIGHTINFO).updateText(msg)
        domoticz.log(msg)
    end
}
```

## Example Hue Lights Timer

```
--[[  
hue_lights_timer.lua  
Turn Hue devices ON & OFF by given time: ON at sunset, OFF at 23:00. Update the alert message.  
IMPORTANT: Ensure the timer and the timer trigger are the same.  
]]--  
  
-- Idx of the devices  
local IDX_HUE_LIVINGSHELF = 111  
local IDX_HUE_LIVINGTV = 112  
local IDX_HUE_DINING = 185  
local IDX_HUE_MAINDOOR = 110  
  
-- Update alert message with alert level green.  
local function setalertmsg(domoticz, state)  
    local message= 'Hue WZ ' .. state .. ' ' .. domoticz.helpers.isnowhhmm(domoticz)  
    domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)  
end  
  
local function hueswitchon(domoticz)  
    domoticz.devices(IDX_HUE_LIVINGSHELF).switchOn()  
    domoticz.devices(IDX_HUE_LIVINGTV).switchOn()  
    domoticz.devices(IDX_HUE_DINING).switchOn()  
    domoticz.devices(IDX_HUE_MAINDOOR).switchOn()  
    setalertmsg(domoticz, 'on')  
end  
  
local function hueswitchoff(domoticz)  
    domoticz.devices(IDX_HUE_LIVINGSHELF).switchOff()  
    domoticz.devices(IDX_HUE_LIVINGTV).switchOff()  
    domoticz.devices(IDX_HUE_DINING).switchOff()  
    domoticz.devices(IDX_HUE_MAINDOOR).switchOff()  
    setalertmsg(domoticz, 'off')  
end  
  
return {  
    on = {  
        timer = {  
            '60 minutes before sunset', '30 minutes before sunset',  
            'at 23:00', 'at 07:00', 'at 08:00'  
        }  
    },  
    execute = function(domoticz, timer)  
        if (timer.trigger == '30 minutes before sunset') then hueswitchon(domoticz) end  
        if (timer.trigger == 'at 23:00') then hueswitchoff(domoticz) end  
    end  
}
```

## Example Hue Group Timer

```
--[[  
hue_control.lua  
Timer: Switch the hue lights of wz group on and off at specific times  
]]--  
local IDX_HUE_WZ_GROUP = 332  
local TIMER_RULE_WZ_GROUP_ON = '5 minutes after sunset'  
local TIMER_RULE_WZ_GROUP_OFF = 'at 23:00'  
  
return {  
    on = { timer = { TIMER_RULE_WZ_GROUP_ON, TIMER_RULE_WZ_GROUP_OFF }, },  
    execute = function(domoticz, item)  
        if (item.isTimer) then  
            if item.trigger == TIMER_RULE_WZ_GROUP_ON then domoticz.devices(IDX_HUE_WZ_GROUP).switchOn() end  
            if item.trigger == TIMER_RULE_WZ_GROUP_OFF then domoticz.devices(IDX_HUE_WZ_GROUP).switchOff() end  
        end  
    end  
}
```

## Example Radiator Thermostat

A complex test script to obtain the setpoint temperature and actual temperature from a HomematicIP radiator thermostat connected to a RaspberryMatic system (see [RaspberryMatic](#)). A text device is updated with the setpoint and temperature.

```
-- Domoticz device text (named MakeLab Thermostat Info)
local IDX_THERMOSTAT_MAKELAB_INFO = 175;
-- Homematic defines
local ID_DEVICE = 1541;
-- url of the CCU to obtain device information
local URL_CCU = 'http://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
-- callback of the url request - must be unique across all dzevents
local RES_CCU = 'thermostatmakelabinfo';
-- helper to round a number to n decimals
local DECIMALS = 1;

return {
  on = { timer = { 'every minute'}, httpResponses = { RES_CCU } },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({ url = URL_CCU, method = 'GET', callback = RES_CCU })
    end
    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        -- SETPOINT °C
        local setpointvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
        -- TEMPERATURE °C
        local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')
        -- log
        domoticz.log('Setpoint: ' .. setpointvalue)
        domoticz.log('Temperature: ' .. temperaturevalue)
        local msg = 'SP: ' .. tostring(domoticz.helpers.roundnumber(setpointvalue, DECIMALS)) .. ', T: ' ..
.. tostring(domoticz.helpers.roundnumber(temperaturevalue, DECIMALS))
        -- update the thermostat info device
        domoticz.devices(IDX_THERMOSTAT_MAKELAB_INFO).updateText(msg);
      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## Utility Dashboard (Grafana, Flowcharting)

## Purpose

To monitor the utilities Heater, Power, Water with connected equipment, in a Supervisory Control And Data Acquisition (SCADA) type way via a Human Machine Interface (HMI) Dashboard – the *Home Utility Unit*.

To use the Home Utility Unit as a prototype for other SCADA type solutions.

## Notes

This solution is rather technical, defined as an Industrial process, based on the authors engineering background.

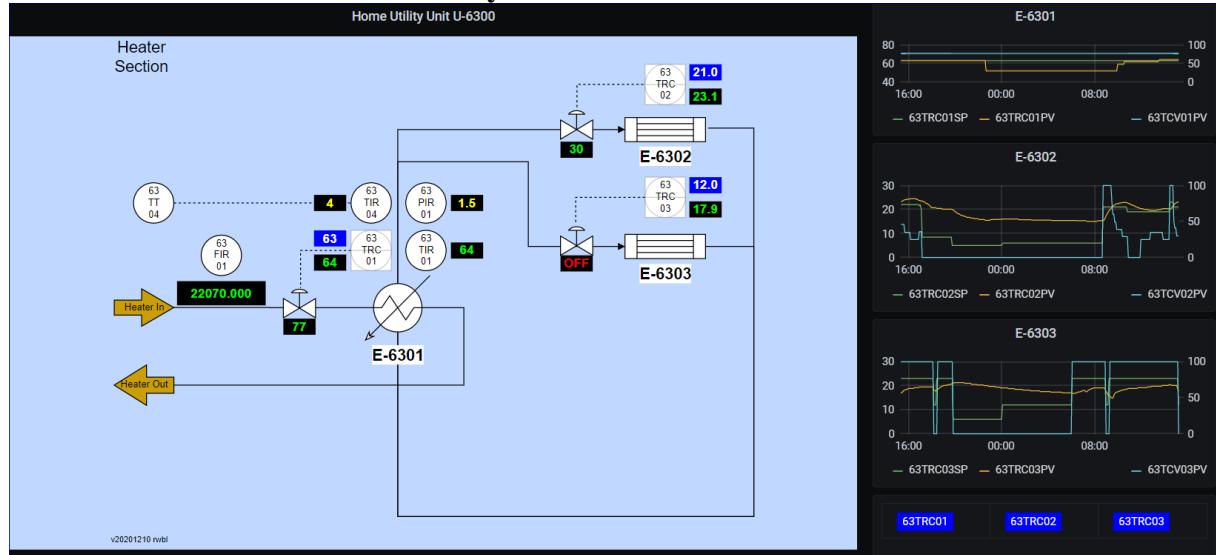
This is a first prototype; not all sections & equipment are included. These are planned for the next version i.e., sections Power & Water, additional radiators, alerts.

If not familiar with Grafana and the plugins used, recommend reading the [Grafana](#) & [FlowCharting](#) documentation first.

Additional information and sources in the archive (zip): Utility-Dashboard.zip.

## Dashboard

Screenshot Dashboard Home Utility Unit Heater Section with Live Data in Kiosk Mode



The dashboard has 4 panels with on the left the **Process Flow Diagram** (PFD) of the **Heater Section** with equipment and instrumentation.

The dashboard right side shows three 24 hour trend graph panels for the Heat-Exchangers and a panel with three buttons to set the setpoint of a Heat-Exchanger.

The dashboard data is refreshed every minute.

The heater (E-6301), temperature controlled (63TRC01), supplies the two radiators (E-6302, E-6303), which are thermostat controlled (63TRC02, 63TRC03).

The heater has indicators showing the supply usage (63FIR01) and the outside temperature (63TIR04) to control the heater temperature setpoint (63TRC01), radiator supply temperature (63TIR01) & pressure (63PIR01).

The heater cannot communicate (old model with no internet connection) with Domoticz, therefor the data (supply usage, temperature & pressure) is entered manually via a Domoticz custom page.

The setpoint is calculated from the outside temperature, based on the heater settings (taken from the datasheet) and send automatically to Domoticz.

Each radiator is temperature controlled with indicators setpoint, temperature and valve position. The data is controlled automatically.

The blue indicators show the setpoint and the green indicators the actual value with thresholds in yellow or red.

The trend graphs show the temperature setpoints, the actual temperature, and the valve position of the temperature control.

The button panel enables to change a setpoint via a simple dialog.

The Home Utility Unit consists out of following sections and devices:

### **Heater Section**

- **Heater (Heat Exchanger)**  
Heating Unit manually controlled (setpoint) & monitored (temperature, pressure, throughput).
- **Radiators (Heat Exchanger)**  
Devices are automatic controlled (setpoint) & monitored (temperature, valve position/level) via homematicIP thermostats using a RaspberryMatic system.

### **Power Section**

- **Electric Usage House**  
Automatic controlled, supplied by a Smart Meter (Type EMH ED300L) connected to a Raspberry Pi running volkszeahler system.
- **Electric Usage Rooms**  
Automatic controlled, supplied by RaspberryMatic system.

### **Water Section**

- **Water Usage House**  
Manually controlled by reading water meter (analog device).

# Solution

## Hard- / Software

The solution makes use of following hard- and software

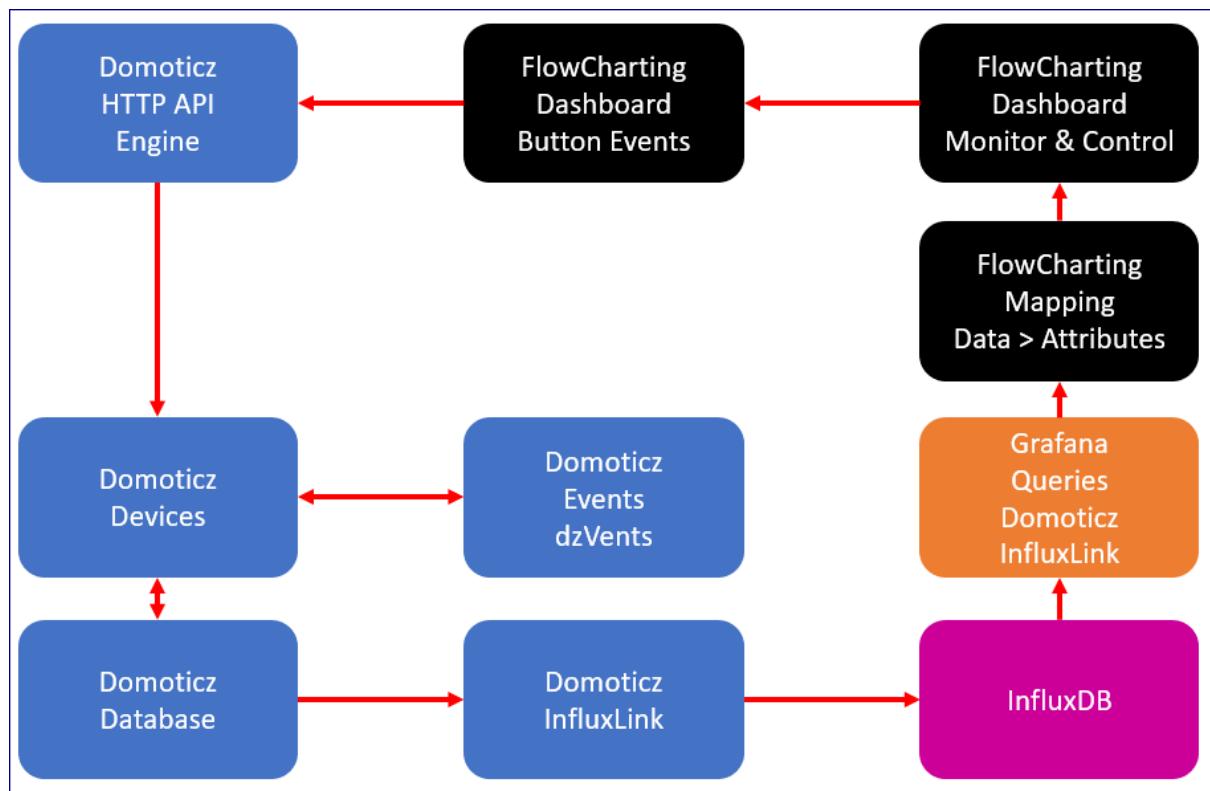
- Raspberry Pi 4 (2GB) with [Raspberry Pi OS](#)
- [Domoticz](#) 2020.2 build 12732 (BETA channel)
- [InfluxDB](#) 1.8.3
- [Grafana](#) v7.3.4
- Grafana [FlowCharting](#) plugin 0.9.0 ([GitHub](#))  
To display complex diagrams using the online graphing library [draw.io](#).

## Information Flow

The Home Utility Unit information flow has been designed first, followed by the Process Flow Diagram indicating the flow of the unit process and equipment (devices).

The instruments and control loops have been included as well.

The Process Flow Diagram acts as the Dashboard to monitor and control the unit.



The information flow provides an overview of the components and the data flows.

### In a nutshell

Starting with a Domoticz device, which data is to be shown on the dashboard. The device is added to the Domoticz Data Push InfluxDB, called InfluxLink – every change is pushed to the InfluxDB.

The InfluxDB stores the data as time series with time stamp, measurement, and tags. Grafana is designed for time series, so its needs data to be stored over a time frame.

The Grafana Flowcharting plugin has an element which shows the value of the Domoticz devices. This is realized by a Grafana query and a mapping of the data to the element.

Overall, it looks complicated, but in essence all fits nicely together – (just) need to be (very) strict in the design.

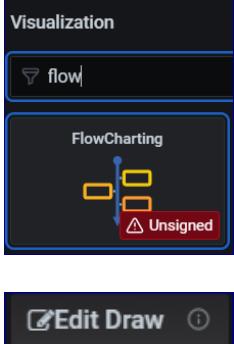
## Design Approach

1. Develop the Process Flow Diagram for the Unit with the Sections.
2. Create Domoticz devices according the Process Flow Diagram.
3. Create Domoticz Data Push InfluxDB devices.
4. Create control logic all devices, performed by Domoticz Automation Events dzVents.
5. Create Domoticz custom page for the Manual Input devices.
6. Start Domoticz Automation Events.
7. Create Grafana Flowcharting panel based on the Process Flow Diagram.
8. Create Grafana queries based on the Domoticz Data Push InfluxDB devices.
9. Create Grafana FlowCharting device mappings to set values and device triggers.
10. Create Grafana Trend Graph panels.
11. Create Grafana Text panel with buttons to change setpoints.
12. Open the Grafana dashboard in kiosk mode.
13. *Check / correct / enhance iteration steps whilst testing ...*

# Process Flow Diagram

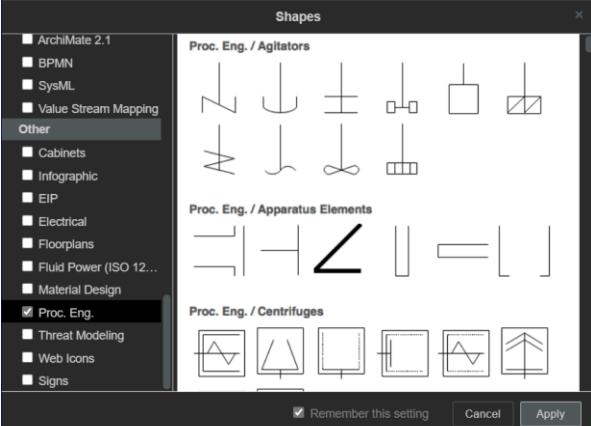
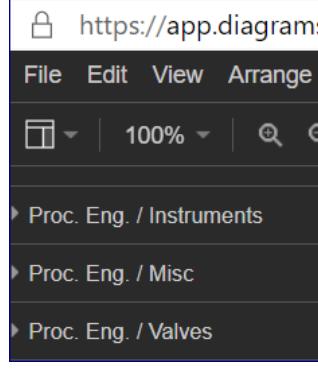
## Overview

In Grafana a new dashboard with a new panel is added.

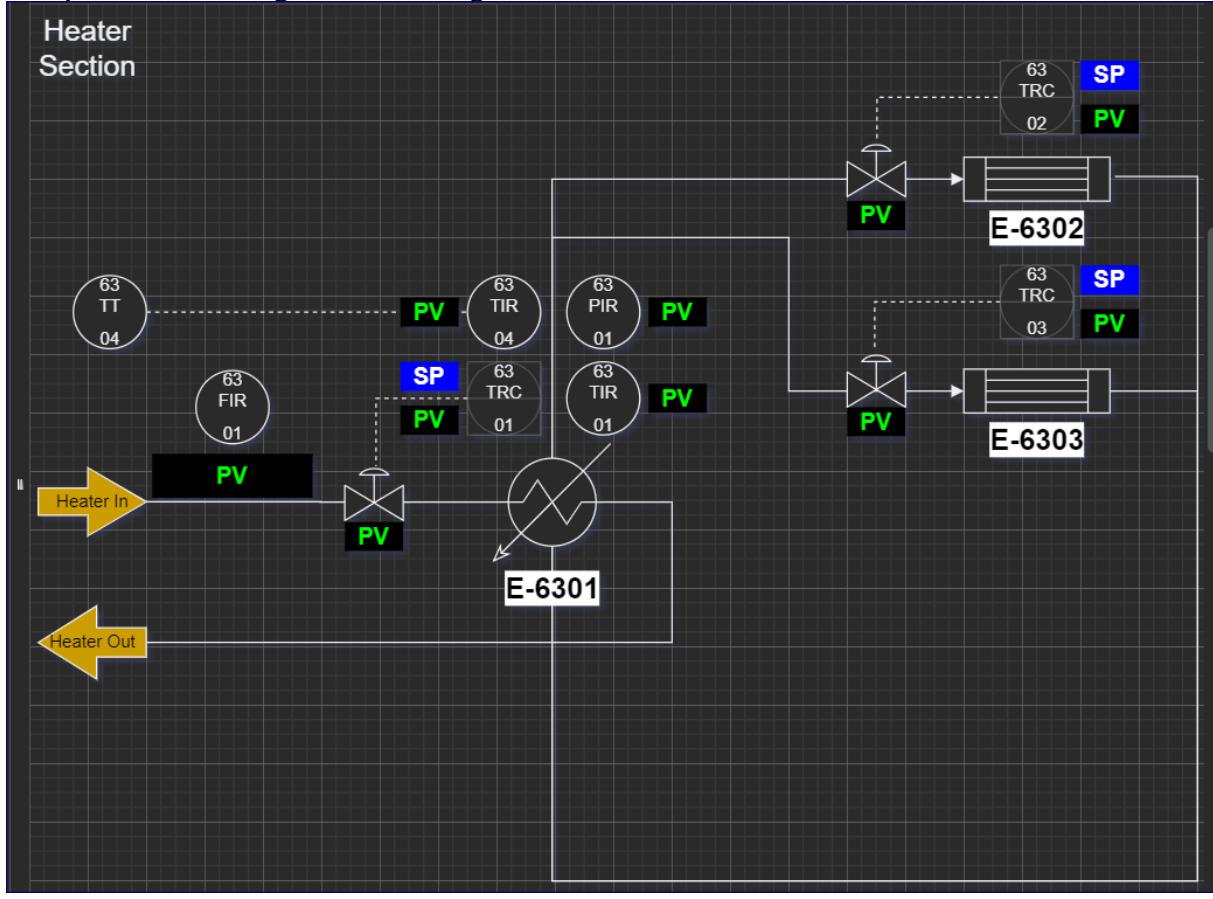
Visualization	Picture	Settings
The new panel visualization type is FlowCharting.	 <b>Edit Draw</b>	<p><b>Flowchart Edition</b></p> <p>Main</p> <p><b>Definition</b></p> <p>Download source: False Editor Url &amp; Theme: <a href="https://www.draw.io">https://www.draw.io</a>, Dark Source Type: XML</p> <p><b>Advanced</b></p> <p>Display: Scale Center: True Grid: True Bg Color: #C0D8FF</p> <p><b>Other options</b></p> <p>Lock: True Allow draw.io source: False Enable animation: False Tooltip: True</p>

The **Process Flow Diagram** is

- created with draw.io (via Option Panel > Edit Draw) and
- the Process Engineering shapes (selected from the Option “+ More Shapes” > select *Other Proc. Eng.*)

Select Proc. Eng. Shapes	Shapes added to the edit draw
	

The process flow diagram in drawing mode.



The Process Flow Diagram has shapes like valves, instrumentation, heat exchangers, text, arrows and more. These are mentioned next as objects.

**To get started:** draw the process flow diagram first on paper.

It is **important** to get the flow with its instrumentation numbering **right first time** to avoid rework when creating the shapes, devices, queries, mappings, automation events.

Better to think twice prior starting drawing & coding.

## Object Identification

### Description

The identification of the various objects is key to get data from Domoticz into the Process Flow Diagram.

#### Name

Each object name must be unique and if linked to a Domoticz device it must have the same name as in Domoticz defined.

#### Example

Domoticz device IDX = 51 with device name = 63TRC02PV.

The draw.io object has the same name: 63TRC02PV.

#### ID

Each object has an unique ID set by draw.io.

This ID is used in Grafana to map an object to data where required.

#### Example

ID: 60 for 63TRC02PV

#### Properties

Each of the instrumentation objects (draw as P&ID symbols) gets two additional properties IDX and Tag.

#### Property IDX

The Domoticz IDX to identify which Domoticz device is mapped to the object.

#### Property Tag

The P&ID identifier following P&ID standards ISA-5.1-2009 (R1992) ([Info](#)) with an own suffix.

#### IMPORTANT:

Each tag must be unique and has no spaces nor special characters.

Unit Identification Number Function	
<b>Unit</b>	Two digit for the unit, i.e., 63 for unit U-6300
<b>Identification</b>	Indicator according P&ID standard abbreviation
<b>Number</b>	Sequence number = Two digits with leading zero
<b>Function</b>	Two letter identifier as suffix of the symbol name: SP = Setpoint OR PV = Process Value This is required because in Domoticz, the device can be an indicator only (PV) or a control (SP). If a symbol has no indicator or control function, SP/PV suffix is left out.

#### Examples

63TT04 - Temperature Transmitter

63TRC02SP - Temperature Setpoint

63TRC02PV - Temperature Process Value

## Summary

### Example

Domoticz IDX	Domoticz Name	Draw.io ID Element	Draw.io Tag Property	Draw.io IDX Property	Draw.io Text
51	63TRC02SP	60	63TRC02SP	52	63TRC02SP
Grafana Query	Grafana Query	FlowCharting Mapping Rule	FlowCharting Mapping Rule		
WHERE ("name" = '63TRC02PV')	ALIAS BY 63TRC02PV	Rule Name: Rule_63TRC02PV	Apply to Metrics: 63TRC02PV		

For many devices used, it is best to create a table as the ID, IDX, Name are required for various design actions as seen from the table.

### Screenshot various definitions

Example device 63TRC02PV (= E-6302 Temperature Process Value)

The screenshot displays four panels side-by-side:

- Domoticz Device Definition:** A table showing device configurations. Row 51 has a blue circle around its 'Idx' value (51). Row 52 has a red circle around its 'Device name' (63TRC02SP). Row 53 has a red circle around its 'Device name' (63TCV02PV).
- Grafana Query:** A query configuration interface. The 'FROM' section shows '63TRC02PV' selected. The 'ALIAS BY' section also shows '63TRC02PV'.
- FlowCharting Draw.io:** A process diagram. A valve on the left is connected to a heat exchanger labeled 'E-6302'. The heat exchanger has two outputs: one goes to a 'SP' node (Setpoint) and another to a 'PV' node (Process Value). The 'SP' node is connected to a '63 TRC 02' node, which is then connected to a '63TRC02PV' node. The '63TRC02PV' node has a red circle around its 'ID: 51' and 'Tag: 63TRC02PV' labels.
- FlowCharting Mapping:** A mapping configuration. It lists a rule: '1 : Rule Rule\_63TRC02PV'. Below it, there's information about the 'Heat Exchanger E-6302' and 'Valve 63TCV01'.

## Styling

The styling for objects has been defined and applied consistent across the diagram.

### Visualisation

#### Flowcharting

Panel Title: U-6300

Display: Scale: true, Center: true, Grid: true, Bg Color:#C0D8FF

Other options: Lock: true, Allow draw.io source: false, Enable animation: true, Tooltip: true

## Draw.io

### Instrument

Style: Fill: Transparent

Text:Helvetica,12pt,center,color:white,position:center,writing direction: auto, wordwrap: true, formatted text: true

### PV (Process value of a measurement)

Style: Fill: black (#000000)

Text:Helvetica,16pt,center,color:green (#00FF00),position:center,writing direction: auto, wordwrap: true, formatted text: true

### SP (Setpoint of a device)

Style: Fill: blue (#0000FF)

Text:Helvetica,16pt,center,color:white (#FFFFFF),position:center,writing direction: auto, wordwrap: true, formatted text: true

### Equipment

Style: Fill: gray (#2A2A2A),line:white

Text:Helvetica,20pt,center,color:white (#FFFFFF),position:center,writing direction: auto, wordwrap: true, formatted text: true

### Arrow (In-/Output Direction)

Style: Fill: yellow (#CB9D06), line: yellow (#CB9D06)

Text:Helvetica,12pt,center,color:black (#000000),background: white (#FFFFFF),position:center,writing direction: auto, wordwrap: true, formatted text: true

### Information Text

Style: Fill: false, line: false

Text:Helvetica,10pt,center,color:white (#FFFFFF),position:center,writing direction: auto, wordwrap: true, formatted text: true

### Mappings Thresholds (Level 2 : Critical > ... > 0 : Ok)

The colors for the levels:

- Level 2: red (#FF0000)
- Level 1: yellow (#FFFF00)
- Level 0: green(#00FF00)

If there are more levels, use like #FF00FF or #00FFFF.

## Hints

- Be consistent in styling.
- For colors red, yellow, green use the highest RGBA colouring values (FF or 255) with A=1.
- For the colors use either HEX or RGB setting. For RGB:  
red = `rgb(255, 0, 0)`, yellow = `rgb(255, 255, 0)`, green = `rgb(0, 255, 0)`
- Drawing could be slow or with some delay depending internet connection speed.  
Be patient and do not draw or change settings too fast.
- For the mapping's threshold colors, the dashboard JSON Model source can be checked or changed as well:  
Open dashboard settings, JSON Model and look for "colors":  
`"colors": ["rgb(255, 0, 0)", "rgb(255, 255, 0)", "rgb(0, 255, 0)"],`

# Domoticz Configuration

The Utility Dashboard runs on a dedicated Domoticz system with virtual devices. The devices are getting data from various sources.

After designing the PFD, in Domoticz the required devices are created according the equipment and assigned to the InfluxLink (function Data Push InfluxDB).

## Devices

The name of the device must the same as shown in the PFD.

### Example

The Temperature of the Heat-Exchanger E-6302 is controlled by 63TRC02 which has three Domoticz devices:

- 63TRC02SP - Temperature Setpoint (Thermostat SetPoint)
- 63TRC02PV - Temperature Process Value (the actual temperature) (Temperature)
- 63TCV02PV – Hot water supply valve position (Percentage)

### Devices List

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
67	VirtualSensors	14093	1	63FIR01PV	P1 Smart Meter	Gas	22067.000
58	VirtualSensors	00082058	1	63PIR01PV	General	Pressure	1.6 Bar
66	VirtualSensors	00082066	1	63TCV01PV	General	Percentage	76.6667%
53	VirtualSensors	00082053	1	63TCV02PV	General	Percentage	38%
56	VirtualSensors	00082056	1	63TCV03PV	General	Percentage	100%
57	VirtualSensors	14089	1	63TIR01PV	Temp	LaCrosse TX3	58.0 C
63	VirtualSensors	1408F	1	63TIR04PV	Temp	LaCrosse TX3	2.3 C
65	VirtualSensors	14091	1	63TRC01PV	Temp	LaCrosse TX3	58.0 C
64	VirtualSensors	0014090	1	63TRC01SP	Thermostat	SetPoint	63.0
51	VirtualSensors	14083	1	63TRC02PV	Temp	LaCrosse TX3	22.7 C
52	VirtualSensors	0014084	1	63TRC02SP	Thermostat	SetPoint	21.0
54	VirtualSensors	14086	1	63TRC03PV	Temp	LaCrosse TX3	18.9 C
55	VirtualSensors	0014087	1	63TRC03SP	Thermostat	SetPoint	23.0
61	VirtualSensors	82061	1	Heating System Alerts	General	Alert	02:00: Device history logs cleared

## Data Push InfluxDB

Grafana uses the InfluxDB as data source.

The InfluxDB runs on the same system as Grafana and Domoticz.

Domoticz has a function **InfluxLink** to send measurements to Domoticz.

Each of the required measurements are defined in the Domoticz InfluxLink configuration.

Idx	Device name	Value to send	Target type	Enabled
67	63FIR01PV	Gas usage	Direct	Yes
58	63PIR01PV	Pressure	Direct	Yes
66	63TCV01PV	Percentage	Direct	Yes
53	63TCV02PV	Percentage	Direct	Yes
56	63TCV03PV	Percentage	Direct	Yes
57	63TIR01PV	Temperature	Direct	Yes
63	63TIR04PV	Temperature	Direct	Yes
65	63TRC01PV	Temperature	Direct	Yes
64	63TRC01SP	Temperature	Direct	Yes
51	63TRC02PV	Temperature	Direct	Yes
52	63TRC02SP	Temperature	Direct	Yes
54	63TRC03PV	Temperature	Direct	Yes
55	63TRC03SP	Temperature	Direct	Yes

### Target Type

The Target type for all devices is set as Direct.

Why?

Grafana is designed for time series, so it needs data to be stored over a time frame.

That means, for a query, if selecting the last("value") with group by "time interval", no records are found if there is no data written within the time interval.

This happens for example when a setpoint or manual entered values are not updated over a longer time.

```
SELECT last("value") FROM "Temperature" WHERE ("name" = 'TRC-6201-PV') AND $timeFilter GROUP BY time($__interval) fill(previous)
```

To ensure that a Grafana query finds data in the InfluxDB, it is required to force data updates from Domoticz by writing direct. *Therefor the InfluxLink target type is set to Direct.*

The disadvantage is writing data to the InfluxDB by Domoticz with time trigger i.e., every minute = populating the database with unused measurements.

### WORKAROUND

- Set Grafana current time range to "Last 24 hours" (in the dashboard top right, also set the current time range as dashboard default).
- Set all queries to SELECT field("value") last("value").
- Set in a Domoticz dzVents a time trigger "every 12 hours" to set the values for all devices even if these have not changed.
- Set in Domoticz InfluxLink the target type to "Direct" for all devices.

### TODO

Checkout if there is a possibility in Grafana to query for the last value without time filter and group by time.

## Data Format

Domoticz sends the measurements in format to the InfluxDB (last table row is an example):

Measurement	Tag idx	Tag name	value
Name of the measurement depending Domoticz device type	IDX of the Domoticz device	Name of the Domoticz device according PFD	Value of the measurement
Temperature	idx=55	name=63TRC03SP	value=23

In the InfluxDB, the measurements are stored with UNIX timestamp (default precision ns).

### Domoticz Log

```
2020-12-12 10:30:01.310 Sending SetPoint to device....
2020-12-12 10:30:01.313 (VirtualSensors) Thermostat (63TRC03SP)
2020-12-12 10:30:01.616 InfluxLink: value Temperature, idx=55, name=63TRC03SP value=23
2020-12-12 10:30:01.616 InfluxLink: value Temperature, idx=54, name=63TRC03PV value=16.3
2020-12-12 10:30:01.616 InfluxLink: value Percentage, idx=56, name=63TCV03PV value=100
```

## InfluxDB CLI

With the InfluxDB Query Language InfluxQL, data can be selected or manipulated.

The InfluxDB is called using the command *influx* from the Raspberry Pi CLI.

- The database used is: *domoticz*.
- The default value of the precision for the CLI is nanoseconds.
- Define tags in double quotes (“tag”) and values in single quotes ('value').

### InfluxDB InfluxQL Examples

Select query on Domoticz device 63TRC03PV

```
$ influx
Connected to http://localhost:8086 version 1.8.3
InfluxDB shell version: 1.8.3
> use domoticz
Using database domoticz
> SELECT * from Temperature WHERE "name" = '63TRC03PV'
name: Temperature
time          idx name      value
----          --- --  -----
...
160776510100000000 54 63TRC03PV 16.3
160776540100000000 54 63TRC03PV 16.3
...
```

The UNIX time 1607765401 (with the nanoseconds) converts to 2020-12-12T09:30:01+00:00.

```
SELECT * from Temperature WHERE "name" = '63TRC03PV' and "time" > 160776570100000000
name: Temperature
time          idx name      value
----          --- --  -----
160776600100000000 54 63TRC03PV 17.3
160776630100000000 54 63TRC03PV 17.6
160776660100000000 54 63TRC03PV 18
```

## Use precision RFC3339

```
precision rfc3339
> SELECT * from Temperature WHERE "name" = '63TRC03PV' and "time" > 1607765701000000000
name: Temperature
time          idx name      value
----          --- -- -
2020-12-12T09:40:01Z 54 63TRC03PV 17.3
2020-12-12T09:45:01Z 54 63TRC03PV 17.6
2020-12-12T09:50:01Z 54 63TRC03PV 18
```

Using InfluxQL is useful to check data or to clean-up measurements.

Some more examples.

### Show Series

Checkout the measurement series pushed from Domoticz to the InfluxDB.

```
> show series
key
---
Gas-usage, idx=62, name=63FIR01PV
Gas-usage, idx=67, name=63FIR01PV
Percentage, idx=53, name=63TCV02PV
Percentage, idx=56, name=63TCV03PV
Percentage, idx=66, name=63TCV01PV
Pressure, idx=58, name=63PIR01PV
Temperature, idx=51, name=63TRC01PV
Temperature, idx=51, name=63TRC02PV
Temperature, idx=52, name=63TRC01SP
Temperature, idx=52, name=63TRC02SP
Temperature, idx=54, name=63TRC02PV
Temperature, idx=54, name=63TRC03PV
Temperature, idx=55, name=63TRC02SP
Temperature, idx=55, name=63TRC03SP
Temperature, idx=57, name=63TIR01PV
Temperature, idx=63, name=63TIR03PV
Temperature, idx=63, name=63TIR04PV
Temperature, idx=64, name=63TRC01SP
Temperature, idx=65, name=63TRC01PV
```

### Drop Series

If a device is renamed, the series can be dropped.

In the previous series list, the Domoticz device 63FIR01PV has two series, beside some other devices also with duplicate idx – this was caused by renaming devices = *therefore ensure to design right from the beginning to avoid rework like this.*

```
key
---
Gas-usage, idx=62, name=63FIR01PV
Gas-usage, idx=67, name=63FIR01PV
```

In the Domoticz Data Push InfluxDB list there is only one device defined with IDX=67.

Idx	Device name	Value to send	Target type	Enabled
67	63FIR01PV	Gas usage	Direct	Yes

The series for the device with idx=62 can be dropped.

```
> DROP SERIES FROM "Gas-usage" WHERE "idx" = '62'
> show series
key
```

```
---  
Gas-usage, idx=67, name=63FIR01PV
```

The cleaned up series list.

```
> show series  
key  
---  
Gas-usage, idx=67, name=63FIR01PV  
Percentage, idx=53, name=63TCV02PV  
Percentage, idx=56, name=63TCV03PV  
Percentage, idx=66, name=63TCV01PV  
Pressure, idx=58, name=63PIR01PV  
Temperature, idx=51, name=63TRC02PV  
Temperature, idx=52, name=63TRC02SP  
Temperature, idx=54, name=63TRC03PV  
Temperature, idx=55, name=63TRC03SP  
Temperature, idx=57, name=63TIR01PV  
Temperature, idx=63, name=63TIR04PV  
Temperature, idx=64, name=63TRC01SP  
Temperature, idx=65, name=63TRC01PV
```

There are in total 13 series which are used next in the Grafana Query definition and FlowCharting mappings.

#### *Next Steps*

The Domoticz device configuration is done.

Any automation events dzVents will be created later.

Let's get the data into Grafana Process Flow Diagram by defining the Grafana Queries and Flowchart Mappings.

## Grafana

Before explaining, recommend reading the chapter [Grafana & InfluxDB](#).

### New Panel

Open Grafana in a browser (<http://domoticzip:3000>) and login with username & password.

Create a new panel with *Visualization FlowCharting*.

### Create Queries

For **all the data** to be displayed in the PFD, select queries must be defined for all devices. For this solution , 13 queries have been defined so far (InfluxDB > InfluxQL > show series).

The query data source selected is **InfluxDB**.

Because data is sent from Domoticz to the InfluxDB via the InfluxLink, the measurements and tags are available and can be selected for each of the queries.

Query	Parameter	Value Example
FROM	Measurement	Percentage
WHERE	Tag Two tags are available: idx and name. The tag name is used as key because easier to read in the query for the devices used. Be aware of the lowercase of the key.	name=63TCV02PV
GROUP BY	time(\$__interval) The interval set by Grafana is used (top right on the screen) fill(previous) Link the time series values	Last 24 Hours
FORMAT AS	Time series The data from the time series is selected	Time series
ALIAS BY	Set the name of the device	63TCV02PV

### Screenshot Query Definition

For the Heat-Exchanger E-6302 valve position 63TRC02PV.

The screenshot shows the Grafana query editor interface. At the top, there are tabs for 'Query' (13) and 'Transform' (0). Below that, the data source is set to 'InfluxDB'. The 'Query options' section shows 'MD = auto = 1257' and 'Interval = 1m'. The main area displays a query configuration table:

FROM	default	Percentage	WHERE	name	=	63TCV02PV	+
SELECT	field (value)	last ()	+				
GROUP BY	time (\$__interval)	fill (previous)	+				
FORMAT AS	Time series						
ALIAS BY	63TCV02PV						

In Edit Mode:

```
SELECT last("value") FROM "Percentage" WHERE ("name" = '63TCV02PV') AND $timeFilter GROUP BY time($__interval) fill(previous)
```

and start to edit the drawing using draw.io (Panel > Flowchart > Button Edit Draw).

Add the shapes *Proc. Eng.* via More Shapes.

Draw the process by selecting and linking shapes.

## Flowchart Mapping

### Introduction

To visualize in the Process Flow Diagram, device values and other actions, like thresholds, **mapping rules** are defined. Recommend reading the [documentation](#).

For each of the devices, mapping rules are defined. So if there are 13 devices, there are 13 Grafana Queries and 13 Flowchart Mappings.

### Screenshot Example two Mappings

Heat-Exchanger E-6302 Thermostat 63TRC02	Heat-Exchanger E-6301 Valve 63TCV01
➤ 2 : Rule [Rule_63TRC02SP] Last state level : [0] Raw value : [21] Formatted value : [21.0] color : <span style="color: green;">■</span> Execute time : [0.00 ms]	▼ 12 : Rule [Rule_63TCV01PV] Last state level : [0] Raw value : [76.6667] Formatted value : [77] color : <span style="color: green;">■</span> Execute time : [0.00 ms]

### Standards

Used following standards for a mapping.

These depend on the kind of object used, like an instrument or equipment (i.e., valve) or information text.

<b>Options</b>	Rule name  Prefix: Rule_ Suffix: Object Name	Rule_63TRC02SP
	Apply to metrics	63TRC02SP
	Aggregation	Last (not null)
<b>Type</b>	Type	Number
	Unit	Short
	Decimals	NN 1
<b>Thresholds</b>	The number of thresholds depends on the object. <i>Default 3 Levels</i> 2 : Critical (RED) > 1: Mid (YELLOW) > 0 : Ok (GREEN) <i>For a valve 2 Levels Open/Close</i> 1 : Critical (RED, Base), 0 : Ok (GREEN, > 5)	
	Color: RED Number: Base Lvl: 2	Rgb(255,0,0) Base 2
	Color: YELLOW Number: NN Lvl: 1	Rgb(255,255,0) > 11 1
	Color: GREEN	Rgb(0,255,0)

	Number: NN Lvl: 0		> 18" 0
	Invert	False	
	Gradient	False	
	Icon state	True	
<b>Tooltips</b>			
	Display metrics	False	
<b>Color/Tooltip Mappings</b>			
	Identify by	Id	
	Regular expression	True	
	What When How		113 Always Label Font Color <i>Note</i> Settings for a valve. Depending threshold, the label color is set. This is done always.
<b>Label/Text Mappings</b>	Change the text of the object with value of the metrics.		
	Identify by	Id	
	Regular expression	True	
	What When How With	ID When Metric Displayed All Content Leave empty	113 When Metric Displayed All Content
<b>Link Mappings</b>			
	Identify by	Id	
	Regular expression	True	
	What When How		
<b>Event/Animation Mappings</b>			
	Identify by	Id	
	Regular expression	True	
	What When How With	ID Threshold Level 2,1,0 Label: Replace: Text New Text	67 2 Replace: Text OFF <i>Note</i> If the threshold level is 2 (Base) change the text of the object 63TRC02SP (blue color) to OFF.

## Trend Panels

For each of the heat-exchangers, a panel is created with a line chart showing 24 hour trend (1-DAY VIEW) for the setpoint, temperature and valve position.

**Screenshot Heat-Exchanger E-6302 (Room MakeLab)**



The setpoint 63TRC02SP and temperature 63TRC02PV are on the left y-axis (0 – 30)°C, whereas the valve position 63TCV02PV is on the right y-axis (0-100%).

## Panel Settings

- Visualization: Graph.
- Display setting Lines.
- Stacking and null value: Connected.
- Relative time range “last 24 hours”.
- Option Fill.
- Three queries used by the lines.
- For all other settings, the defaults are used.
- Panel has been positioned on the screen at the right side.

## Queries

There are 3 queries defined for each of the measurements: Setpoint (63TRC02SP), Temperature (63TRC02PV) and Valve Position (63TCV02):

```
SELECT mean("value") FROM "Temperature" WHERE ("name" = '63TRC02SP') AND $timeFilter GROUP BY time($__interval) fill(previous)
```

```
SELECT mean("value") FROM "Temperature" WHERE ("name" = '63TRC02PV') AND $timeFilter GROUP BY time($__interval) fill(previous)
```

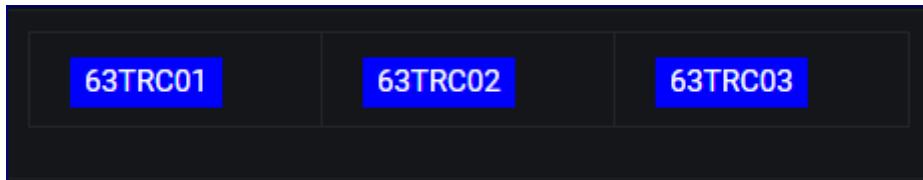
```
SELECT mean("value") FROM "Percentage" WHERE ("name" = '63TCV02PV') AND $timeFilter GROUP BY time($__interval) fill(previous)
```

## Button Panel

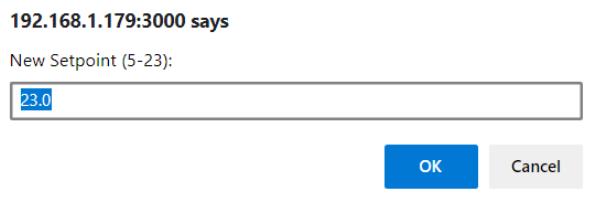
For the dashboard, have been looking for a way to change the setpoint of a heat-exchanger.

The best way would be to click on the blue SP field in the Process Flow Diagram ... but have not found a solution on how to do so.

Therefor worked out a solution with a Text panel, using the display option HTML containing buttons (“Button Panel”) to change the setpoint for each of the heat-exchangers.  
Clicking on a button, opens a simple prompt dialog to change the setpoint.



### Example New Setpoint Dialog



## Panel Settings

- Visualization: Text.
- Relative time range “last 24 hours”.
- Option Fill.
- Panel title & description are left empty to hide the title bar.
- Panel has been positioned on the screen bottom right.

## Enable HTML Scripting

For **security reasons**, the usage of HTML script tags is not recommended by Grafana ([Info](#)). Because the Utility Dashboard is running on a dedicated system for local use only, the option to use script tags, has been enabled.

This is done by making a change in the *grafana.ini* file located, on the Raspberry Pi, in folder */etc/grafana/*.

```
sudo /etc/grafana/grafana.ini
```

In section *panels*, set key *disable\_sanitze\_html* to *true*.

```
[panels]
# If set to true Grafana will allow script tags in text panels. Not recommended
disable_sanitze_html = true
```

After making the change, restart Grafana:

```
sudo service grafana-server restart
```

## HTML & JavaScript Code

The buttons are defined in a table with a single row.

The button onClick event calls the JavaScript function setSetPoint() with arguments Domoticz device IDX and Domoticz device Name.

The JavaScript function setSetPoint() performs following steps:

1. Submit an HTML API request to the Domoticz system to get the actual setpoint for the thermostat

```
http://domoticz-ip:port/json.htm?type=devices&rid=IDX
```

2. Open a simple input dialog (prompt) with the actual setpoint as default.

*Note*

The dialog is rather simple using the default prompt function. There is no error checking nor the option to select the value from a dropdown.

3. If the setpoint is changed and the dialog is closed with OK, an HTML API request to the Domoticz system is submitted to set the new setpoint.

```
http://domoticz-ip:port/json.htm?type=command&param=setsetpoint&idx=IDX&setpoint=NEWSETPOINT
```

## HTML Content Source

```
<!-- HTML -->
<!-- 63TRC01SP=64,63TRC02SP=52,63TRC03SP=55-->


|                                                                                                                                                              |                                                                                                                                                              |                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <button width="25%" style="font-size: 12px; background-color: #0000FF; border: none; color: white;" onclick="setSetPoint(64, '63TRC01SP');">63TRC01</a></td> | <button width="25%" style="font-size: 12px; background-color: #0000FF; border: none; color: white;" onclick="setSetPoint(52, '63TRC02SP');">63TRC02</a></td> | <button width="25%" style="font-size: 12px; background-color: #0000FF; border: none; color: white;" onclick="setSetPoint(55, '63TRC03SP');">63TRC03</a></td> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

```
requestUrl=urlRequest+"type=command&param=setsetpoint&idx="+deviceIdx+"&setpoint="+deviceSetpoint;
console.log("updateDevice - URL: " + requestUrl);
$.ajax({
  url: requestUrl,
  async: false,
  dataType: 'json',
  success: function (data) {
    if (typeof data != 'undefined') {
      if (data.status == "OK") {
        alert('Setpoint: ' + deviceSValueSelector + ' set to ' + deviceSetpoint);
      }
      else {
        alert('[ERROR]Change setpoint:' + deviceSValueSelector + ' set to ' + deviceSetpoint);
      }
    }
  },
  error: function () {
    alert('[ERROR]No Domoticz connection.');
  }
});
},
error: function () {
  console.log("[ERROR]No communication with the Domoticz system!");
  return false;
}
});
};
</script>
```

## Enhancements

Ideas for any next version of the Utility Dashboard.

### Set Setpoint User Dialog

Create an user dialog for setting a new setpoint with error checking and dropdown to select the new setpoint.

### Set Setpoint via Link

In the Process Flow Diagram, when clicking on a setpoint text field, open a browser window with a dialog to change the setpoint. The browser window could be a custom page with fields to change the setpoint or open the Domoticz GUI with the setpoint widget.

### Text Panel Alternative

Instead, a text panel with HTML & JavaScript, use a plugin with buttons.

Explore if there are plugins available, else develop own plugin.

## Domoticz Custom Page

A custom page is used to manual input measurements for devices that do not automatically update Domoticz.

This applies to data from the Heater E-6301.

The custom page has three input fields each with a button, to insert the measurement in the Domoticz database via HTTP API request.

### Screenshot Custom Page on Android Mobile Device

	<p>Each measurement has an input field with a</p> <ul style="list-style-type: none"> <li>• Heading – Equipment and measurement</li> <li>• Label – Instrument Name</li> <li>• Hint – Unit, min/max or other info</li> <li>• Set button – Sent the measurement value to Domoticz</li> </ul>
--	---

### Pseudo Code

1. Page load  
For each of the measurements get the actual value from the Domoticz database.
2. Manual Input the new value
3. Button Set Pressed  
The new value is inserted in the Domoticz database via HTTP API Request

### Source Code

Filename: Utility-Dashboard-Manual-Input.html (~domoticz/www/templates)

```
<!--
domoticz-homeautomation-workbook function Utility-Dashboard
Domoticz custom page:
Utility-Dashboard-Manual-Input.html (/home/pi/domoticz/www/templates)
Purpose:
To change the value of devices which do not have automatic updates.
63TIR01PV, idx=57, id=70, Heater temperature
/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP
IDX = id of your device (This number can be found in the devices tab in the column "IDX")
TEMP = Temperature
63PIR01PV, idx=58, id=72, Heater pressure
/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=BAR
IDX = id of your device (This number can be found in the devices tab in the column "IDX")
BAR = Pressure in Bar
63FIR01PV, idx=62, id=71, Heater Supply
/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=COUNTER
IDX = id of your device (This number can be found in the devices tab in the column "IDX")
COUNTER = Counter in liter

Any other settings, to be set via Domoticz GUI Setup > Settings (the standard way).
-->
```

```
-->
<style>
/* hint style text small */
.hint {
  font-size: smaller;
  color: white;
}

/* HR thin style class */
.hrthin {
  height:2px;
  border-width:0;
  color:gray;
  background-color:gray;
}
</style>






```

```

</tr>
<tr>
    <td align="right" style="width:50px"><label for="63FIR01PV"><span data-
i18n="63FIR01PV"></span></label></td>
    <td>
        <input id="63FIR01PV" type="number" min="0" style="width: 100px; padding: .2em;" class="text ui-
widget-content ui-corner-all" required />
        <a align="right" class="btnstyle3" onclick="updateDevice(67, '63FIR01PV', 0);" data-
i18n="Set"></a>
    </td>
</tr>
<tr>
    <td align="right" style="width:50px"></td>
    <td class="hint">(Liter, Min = 0, Enter without comma, i.e. 21998369)</td>
</tr>
</table>

<!-- Version info --&gt;
&lt;hr class="hrthin"&gt;
&lt;div id="pageversion" class="hint" align="right"&gt;&lt;/div&gt;

&lt;script type="text/javascript" charset="utf-8"&gt;
// Update the value of a device depending device type
// T &amp; P = json.htm?type=command&amp;param=udevice&amp;idx=IDX&amp;nvalue=0&amp;svalue=BAR
// updateDevice(57, '63TIR01PV', decimals);
function updateDevice(deviceIdx, deviceSValueSelector, valueDecimals) {
    var deviceNValue = 0;
    var deviceSValue = Number(document.getElementById(deviceSValueSelector).value);
    deviceSValue = deviceSValue.toFixed(valueDecimals);
    var requestUrl = "json.htm?type=command&amp;param=udevice&amp;idx=" + deviceIdx + "&amp;nvalue=" + deviceNValue
+ "&amp;svalue=" + deviceSValue;
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            if (typeof data != 'undefined') {
                if (data.status == "OK") {
                    ShowNotify($.t('Device ' + deviceSValueSelector + ' updated.'), 2500, true);
                }
                else {
                    ShowNotify($.t(data.message), 2500, true);
                }
            }
        },
        error: function () {
            ShowNotify($.t('Problem saving device data!'), 2500, true);
        }
    });
};

// Get the device properties
// Data = json.htm?type=devices&amp;rid=IDX
// Parameter:
// deviceIdx (Integer): 57
// deviceSValueSelector (String): 63TIR01PV
// deviceProperty (String): Temp, Pressure, Counter etc. !Check out the device properties
// getDevice(57, '63TIR01PV', 'Temp');
function getDevice(deviceIdx, deviceSValueSelector, deviceProperty) {
    var requestUrl = "json.htm?type=devices&amp;rid=" + deviceIdx;
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            if(typeof data.result == "undefined"){
                console.log("getDevice - no data found (status:" + data.status + ",title:" + data.title + ")");
                return false;
            }
            var propertyValue = data.result[0][deviceProperty];
            // Handle special property cases
            propertyValue = (deviceProperty == 'Counter') ? propertyValue * 1000 : propertyValue;
            // Update the device input with the property
            $('#'+deviceSValueSelector).val(propertyValue);
        },
    });
}
</pre>

```

```
error: function () {
    console.log("[ERROR] getDevice. Can not communicate to the Domoticz server!");
    return false;
}
});

// Sleep for ms
function sleep(milliseconds) {
    const date = Date.now();
    let currentDate = null;
    do {
        currentDate = Date.now();
    } while (currentDate - date < milliseconds);
}

function setVersion(pageVersion) {
    $('#pageversion').html(pageVersion);
};

// Load the document
$(document).ready(function(){
    setVersion("20201210 rwbl");
    // Set the input fields with actual data
    // Ensure to use the right case for the Domoticz device property
    getDevice(57, '63TIR01PV', 'Temp');
    sleep(50);
    getDevice(58, '63PIR01PV', 'Pressure');
    sleep(50);
    getDevice(67, '63FIR01PV', 'Counter');
    sleep(50);
});
});
```

# Automation Scripts

The usage of automated scripts dzVents depends on the devices and how these are getting their data.

For this solution, Domoticz & Grafana & InfluxDB are running on a dedicated system with no production data.

This means, automation events are required to get data but also to perform tasks like data checks, make calculations and other.

The next automation events are specific for this solution but think good to share the concepts.

## U6300-Heater

*Source: U6300-Heater.dzvents*

Push mandatory to the InfluxDB to ensure data is displayed - trigger once/day

This is done for the manual input devices: 63TI01PV (Heater Temperature), 63PI01PV (Heater Pressure), 63FI01PV (Heater Gas Counter)

The data is manually updated using the custom page Utility-Unit-Manual-Input.html (~domoticz/www/templates)

The outside temperature transmitted from 63TT04 is used to calculate the Heater temperature setpoint 63TRC01SP.

The calculation of 63TRC01SP is based on the Heater settings (taken from the Heater datasheet).

63TT04 gets the outside temperature from another Domoticz system, requested via HTTP API function Device Status (IDX=344).

For future solutions, its planned to use either RaspMatic device or an external service (for example met.io) to get the outside temperature.

## U6300-Heat-Exchanger

*Source: U6300-Heat-Exchanger.dzvents*

Obtain in regular intervals (trigger timer) the data for the heat exchangers and update the InfluxDB.

The heat exchangers are controlled by thermostats homematic-IP connected to a RaspberryMatic CCU3.

Decided for this prototype to PULL for data instead awaiting PUSH updates.

The solution is a process control prototype with Piping and Instrumentation Diagram (P&ID). identifier.

So far, two heat-exchangers are included for the rooms: E-6302 = MakeLab, E-6303 = Living-1

*Note*

More heat-exchangers planned for next version.

*Abbreviations*

SP=SetPoint, PV=Process Value, T=Temperature, R=Recording, C=Control, V=Valve, E=Heat-Exchanger

## Domoticz Roomplan

A room plan is defined for all devices of the Utility Dashboard.

The room plan has 13 devices used by the Process Flow Diagram plus an Domoticz Alert device. The Alert device contains messages from the Automation Events.

### Screenshot Room Plan Setup Utility Dashboard

The screenshot shows the Domoticz Room Plan Setup Utility Dashboard. It consists of two main sections: "Room plans" and "Devices".

**Room plans:**

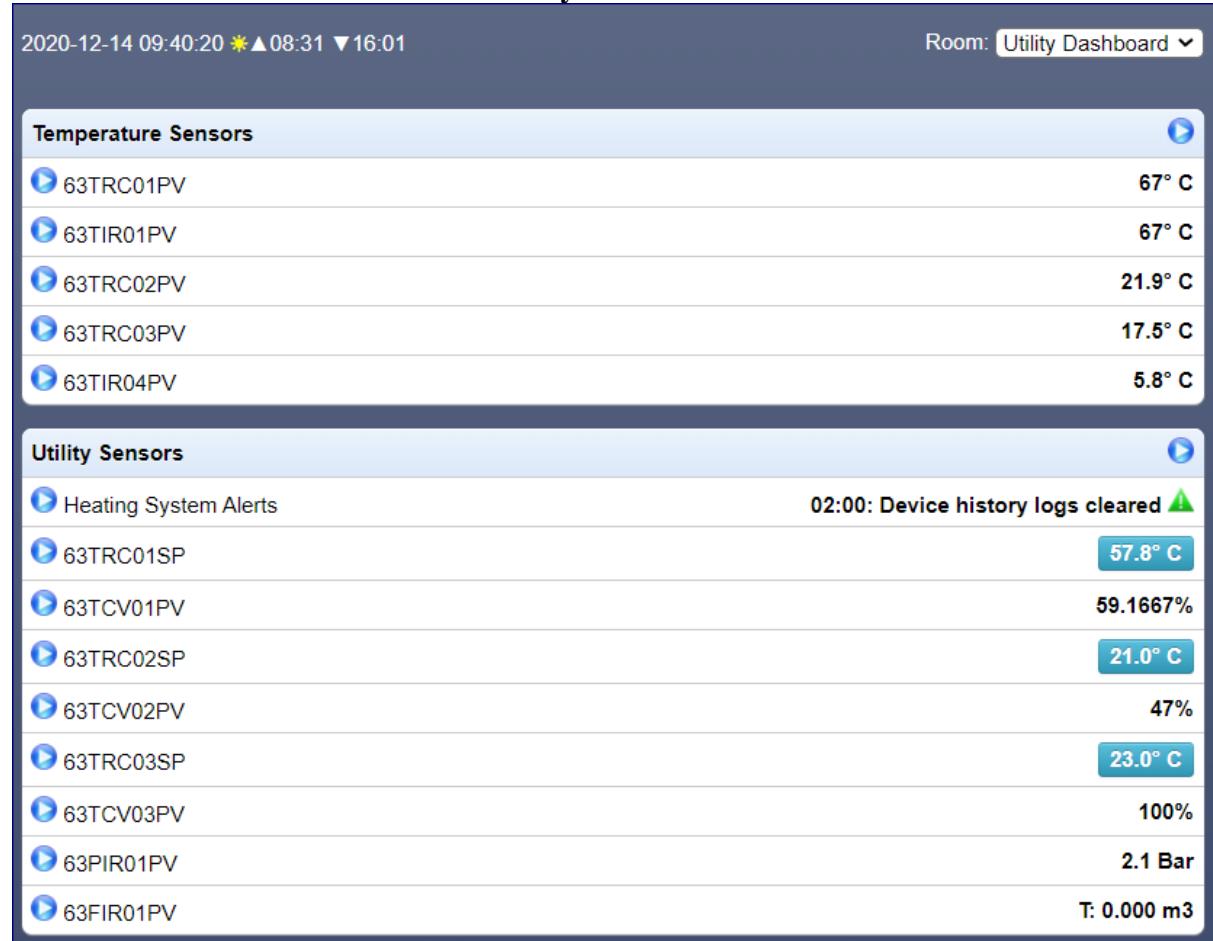
Idx	Name	Order
6	Utility Dashboard	(Up/Down arrows)

Show 25 entries | Search: utility | Add Plan | First | Previous | 1 | Next | Last | Showing 1 to 1 of 1 entries (filtered from 6 total entries) 1 row selected

**Devices:**

Idx	Name	Order
61	Heating System Alerts	(Up/Down arrows)
65	63TRC01PV	(Up/Down arrows)
64	63TRC01SP	(Up/Down arrows)
66	63TCV01PV	(Up/Down arrows)
57	63TIR01PV	(Up/Down arrows)
51	63TRC02PV	(Up/Down arrows)
52	63TRC02SP	(Up/Down arrows)
53	63TCV02PV	(Up/Down arrows)
54	63TRC03PV	(Up/Down arrows)
55	63TRC03SP	(Up/Down arrows)
56	63TCV03PV	(Up/Down arrows)
58	63PIR01PV	(Up/Down arrows)
63	63TIR04PV	(Up/Down arrows)
67	63FIR01PV	(Up/Down arrows)

Show 25 entries | Search: | Add | Clear | First | Previous | 1 | Next | Last | Showing 1 to 14 of 14 entries

**Screenshot Dashboard Room Plan Utility Dashboard**

# Weather Forecast (Service, Meteorologisk)

## Purpose

To view the 9-DAYS WEATHER FORECAST for a selected location in a chart. The data, updated hourly is temperature, air pressure and rain fall.

## Solution

### Overview

The full weather forecast data for a selected location is requested, via HTTP product Locationforecast, from the [Norwegian Meteorological Institute](#) (Meteorologisk) using the MET Norway Weather API.

The request is initiated every hour by a Domoticz Automation Script dzVents.

The script parses the HTTP API response and for each day, the data time, pressure, temperature is selected for the hours 0,6,12,8. The data is stored in a JSON file.

A custom page with a Highcharts combined column & line chart reads the JSON file to create the 9-DAYS FORECAST.

### Screenshot Custom Page Highcharts Combined Chart



The location, in this case Hamburg, is selected from a Domoticz Selector Switch to be able to view multiple locations.

The selector level name contains the location information as CSV string Name,Lat,Lon.

WXFX		Hamburg (DE),53.6375,9.7987
	Last Seen:	2020-11-28 10:32:19
Type:	Light/Switch, Selector Switch, Selector	
<input type="button" value="Hamburg (DE),53.6375,9.7987"/>		
	Hamburg (DE),53.6375,9.7987	
<input type="button" value="Arrecife (ES),28.9630,-13.5477"/>		
<input type="button" value="Domburg (NL),51.567,3.500"/>		

#### Note

The device name is shorthanded to get the location as full title.

## MET Weather API

Read the [documentation](#) (Version 3 at the time of writing).

The HTTP API locationforecast request contains the Lat & Lon parameter for the selected location.

### Example Full Request:

```
https://api.met.no/weatherapi/locationforecast/2.0/.json?lat=53.6375&lon=9.7987
```

The HTTP API request returns a JSON object covering the 9-DAYS FORECAST.

### Example Full Response (snippet):

```
{
  "type": "Feature",
  "geometry": {"type": "Point", "coordinates": [9.7987, 53.6375, 9]},
  "properties": {
    "meta": {"updated_at": "2020-11-24T17:52:07Z", "units": {"air_pressure_at_sea_level": "hPa", "air_temperature": "celsius", "air_temperature_max": "celsius", "air_temperature_min": "celsius", "cloud_area_fraction": "%", "cloud_area_fraction_high": "%", "cloud_area_fraction_low": "%", "cloud_area_fraction_medium": "%", "dew_point_temperature": "celsius", "fog_area_fraction": "%", "precipitation_amount": "mm", "precipitation_amount_max": "mm", "precipitation_amount_min": "mm", "probability_of_precipitation": "%", "probability_of_thunder": "%", "relative_humidity": "%", "ultraviolet_index_clear_sky": "1", "wind_from_direction": "degrees", "wind_speed": "m/s", "wind_speed_of_gust": "m/s"}},
    "timeseries": [{"time": "2020-11-24T18:00:00Z", "data": {"instant": {"details": {"air_pressure_at_sea_level": 1020.3, "air_temperature": 6.5, "cloud_area_fraction": 98.2, "cloud_area_fraction_high": 97.6, "cloud_area_fraction_low": 35.2, "cloud_area_fraction_medium": 0.0, "dew_point_temperature": 4.7, "f
...
}}
```

The key timeseries holds the forecast data for the 9-DAYS. Depending time of the HTTP request, the timeseries array has a size up-to 100 (or more).

From each timeseries entries, following keys are used:

- time = select the timestamp
- data.instant.details = select weather data for the given time

#### Note

There is also a compact HTTP API request:

```
https://api.met.no/weatherapi/locationforecast/2.0/compact?lat=53.6375&lon=9.7987
```

returning less data then the full request. This should be sufficient, if just interested in basic weather information, i.e. instant data:

```
air_temperature, air_pressure_at_sea_level, wind_speed, cloud_area_fraction, relative_humidity,
wind_from_direction
```

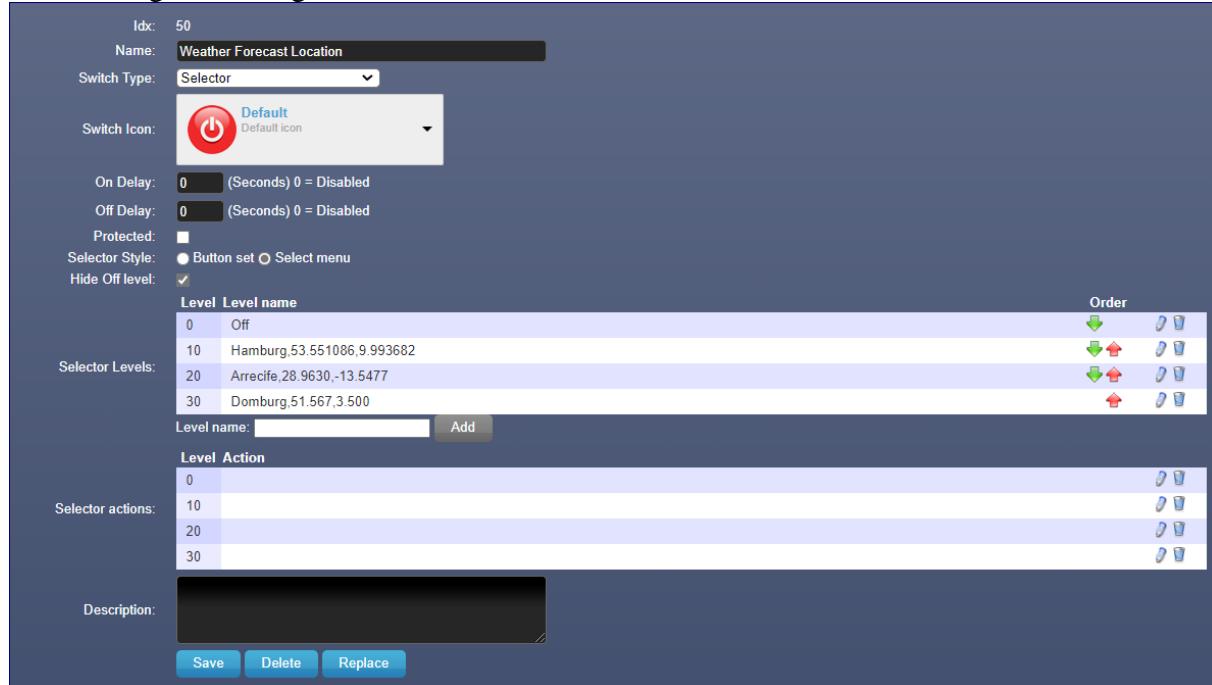
# Domoticz Configuration

## Devices

A selector switch is used to select a location.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
50	VirtualSensors	00014082	1	Weather Forecast Location	Light/Switch	Selector Switch	Set Level: 20 %

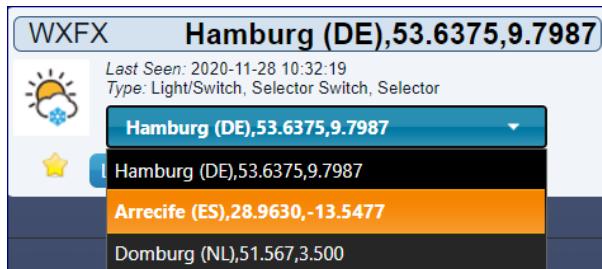
After creating the device using the Hardware Dummy (here named VirtualSensors), the device widget is configured:



- Selector levels contains CSV string: Location,Lat,Lon.  
The screenshot shows 3 selector levels as location definition.
- Selector Style: Select menu
- Hide Off level: Selected, i.e. is not used.
- There are no selector actions.  
An automation script dzVents handles the selector level changes.
- The Selector Switch Widget title shows the selected level Location,Lat,Lon. If the Selector Switch name is long the text of the name & selected level override. To avoid, give the Selector Switch a shorter name or use technical abbreviation, i.e. WXFX (see below screenshot).
- Optional: Add a custom icon reflecting the purpose Weather Forecast.  
See archive *weather-forecast-customicons.zip*.  
Add via GUI > Setup > More Options > Custom Icons > Choose File *weather-forecast-customicons.zip* > Upload > Edit the widget and select the new icon.

## Screenshot Selector Switch

Widget with short name WXF, custom icon, selected location, and list of the three selector levels.



## Automation Event

The dzVents Lua automation event *weather\_forecast.dzvents* handles device changes of the selector switch and the hourly update of the JSON file used by the custom page.

### Pseudo Code

```
Get location from the selector switch or f not available from the Domoticz Location Settings.
Submit for the selected location, a HTTP API Full Request to met.no.
Parse the HTTP API JSON Response to a Lua Table.
Create Lua Data Table.
Loop over the Lua Table
  Add selective keys to the Lua Data Table.
Export the Lua Data to a Data JSON File used by the Custom Page.
```

The data file for the Custom Page is stored, on the Raspberry Pi in:

```
~domoticz/www/templates/weather_forecast_data.json
or
/home/pi/domoticz/www/templates/weather_forecast_data.json
```

The data file format:

```
{
  "result" :
  [
    {
      "air_pressure_at_sea_level":1013.5,
      "air_temperature":17.8,
      "precipitation_amount":0.9,
      "probability_of_precipitation":0,
      "time":"26-11 12:00",
      "wind_speed":10.4
    },
    ...
    {
      "air_pressure_at_sea_level":1019.0,
      "air_temperature":19.1,
      "precipitation_amount":0,
      "probability_of_precipitation":0,
      "time":"05-12 18:00",
      "wind_speed":3.2
    }
  ],
  "status":"OK",
  "title":"Weather Forecast",
  "location":"Arrecife",
  "updated":"26-11 08:47"
}
```

The format is aligned with the Domoticz HTTP API Engine but has some enhancements like location & updated.

The weather data for the times 0,6,12,18 is stored in the key *result* as array.

## Custom Page

The Domoticz custom page to display the charts is

```
~domoticz/www/templates/weather-forecast.html
```

The Highcharts library is used to create the chart.

### Credits

The solution is based upon [this](#) example from Highcharts. Many thanks for providing

The source code of the custom page *weather-forecast.html* is not included in this document.

Lookup the src folder of the workbook repository.

Recommend also to read chapter [Custom Pages Charts](#).

## Concept Information

The custom page reads the JSON file and creates an array for each of the data series, i.e.

```
var dataCategories = [];
var dataAirPressureAtSeaLevel = [];
var dataAirTemperature = [];
var dataPrecipitationAmount = [];
var dataProbabilityOfPrecipitation = [];
var dataWindSpeed = [];
```

### Note

For future chart enhancements, the *dataProbabilityOfPrecipitation* and *dataWindSpeed* are also created but not used.

Next the Highcharts options are set and the data series assigned:

```
xAxis.categories: dataCategories
series: [
  {name: 'Rainfall', type: 'column', yAxis: 1, data: dataPrecipitationAmount, ...}
  {name: 'Sea-Level Pressure', type: 'spline', yAxis: 2, data: dataAirPressureAtSeaLevel, ...}
  {name: 'Temperature', type: 'spline', data: dataAirTemperature, ...}
]
```

The options are quite extensive. To learn, it's helpful to check with the Highcharts [documentation](#).

The chart is drawn using the chart options in the div container with id *chartcontainer* and Domoticz class *device-log-chart* (see Domoticz style.css in www/css folder)

```
<div class="chart noselect">
  <div class="chartarea">
    <!-- Container containing the chart -->
    <div id="chartcontainer" class="device-log-chart"></div>
  </div>
</div>
```

Extra functions used in the Highcharts Burger menu are Refresh (refreshChart) and About (showMessage).

For refresh, the page is reloaded fetching a clear copy to ensure the cache is cleared to get the latest location.

The chart can be opened in a browser directly by using the URL:

```
http://domoticz-ip:port/#/Custom/weather-forecast
```

## Installation

Consider using the Domoticz Settings Location or ability to select a location.

If location should be selectable:

- Create a Domoticz Virtual Sensor type Selector Switch.
- Define Selector Levels for the locations containing Location,Lat,Lon.
- Note the IDX of the Selector Switch.

Create a new event GUI > Setup > More Options > Events > + > dzVents > Minimal.

Clear the content of the event.

Copy & paste the content of the file *weather\_forecast.dzvents*.

Set the IDX of the local *IDX\_FORECAST\_LOCATOR*:

- Set -1 if Domoticz Settings Location else the IDX of the selector switch.

Set the event to On and save as *weather\_forecast*.

Check the Domoticz log and if the file *~domoticz/www/templates/weather\_forecast\_data.json* is created.

Copy the file *weather\_forecast* to folder *~domoticz/www/templates*.

Open browser with Domoticz.

Select GUI > Custom > Weather\_forecast

If all went well, the chart for the location is shown.

## Hints

### **Timer**

For tests, change timer rule from ‘every hour’ to ‘every minute’ and check the Domoticz log.

```
local TIMERRULE = 'every minute'
```

### **File Ownership**

The JSON file ownership is root. If want to change to user pi, then uncomment in function *writeFile*:

```
domoticz.utils.osCommand(OSCHOWM .. f)
```

This requires Domoticz version higher 2020.2 (April 26th, 2020).

### **Browser Page Refresh**

If the Custom Page is not updated after selecting another location, clear the browser cache or try select Refresh from the Custom Page Burger Menu.

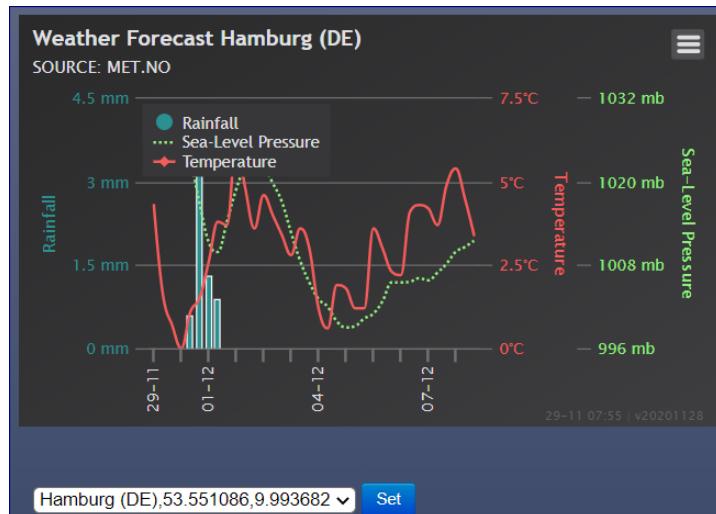
## Enhancements

Idea's to enhance this solution.

### Custom Page Location Selector

Add an HTML select element to build a drop-down list with location options.

Change the location via a button.



Source: weather-forecast-2.html

The source code is well documented, but more information on how the location selection works.

### HTML Selector Location

The HTML *SELECTOR* element is populated from the Selector Switch Level Names.

The Selector Switch attribute *LevelNames* holds a base64 decoded string containing CSV string (by |). The *LevelNames* string is obtained via HTTP API request.

### Example Device Status IDX=50

```
http://domoticz-ip:port/json.htm?type=devices&rid=50
```

### HTTP Response (snippet)

```
{
  "app_version" : "2020.2 (build 12705)",
  "result" :
  [
    {
      "Level" : 10,
      "LevelActions" : "fHx8",
      "LevelInt" : 10,
      "LevelNames" :
        "T2ZmfEhhbWJ1cmcgKERFKSw1My41NTEw0DYsOS450TM20DJ8QXJyZwNpZmUgKEVTKSwy0C45NjMwLC0xMy41NDc3fERvbWJ1cmcgKE
        5MKSw1MS41NjcsMy41MDA=",
      "LevelOffHidden" : true,
      "Name" : "WXFX",
      "idx" : "50"
    }
  ],
  "status" : "OK",
  "title" : "Devices"
}
```

In JavaScript, the *LevelNames* string is decoded using the function *atob(base64 string)*

```
Off|Hamburg,53.551086,9.993682|Arrecife,28.9630,-13.5477|Domburg,51.567,3.500
```

and then split into an array:

```
var locationLevels = atob(data.result[0].LevelNames).split('|');
```

The 4 items of the array *locationLevels* are used to define the options for the selector element. The device attribute *Level* is used to set the selected value of the selector element.

```
<option value=' + level + ' + valueSelected + '>' + locationLevels[i] + '</option>';
```

After pressing the button *Set*, the selected level is set for the Selector Switch device with IDX=50:

```
var requestUrl = "json.htm?type=command&param=switchlight&idx=" + deviceIDX +
"&switchcmd=Set%20Level&level=" + levelValue;
```

The device IDX is defined in HTML DIV element *idx* with attribute *data-value*.

```
<!-- Define the IDX of the Domoticz location selector switch. Default 50. -->
<div id="idx" data-value="50" hidden></div>
```

# Web App Site Control

## Purpose

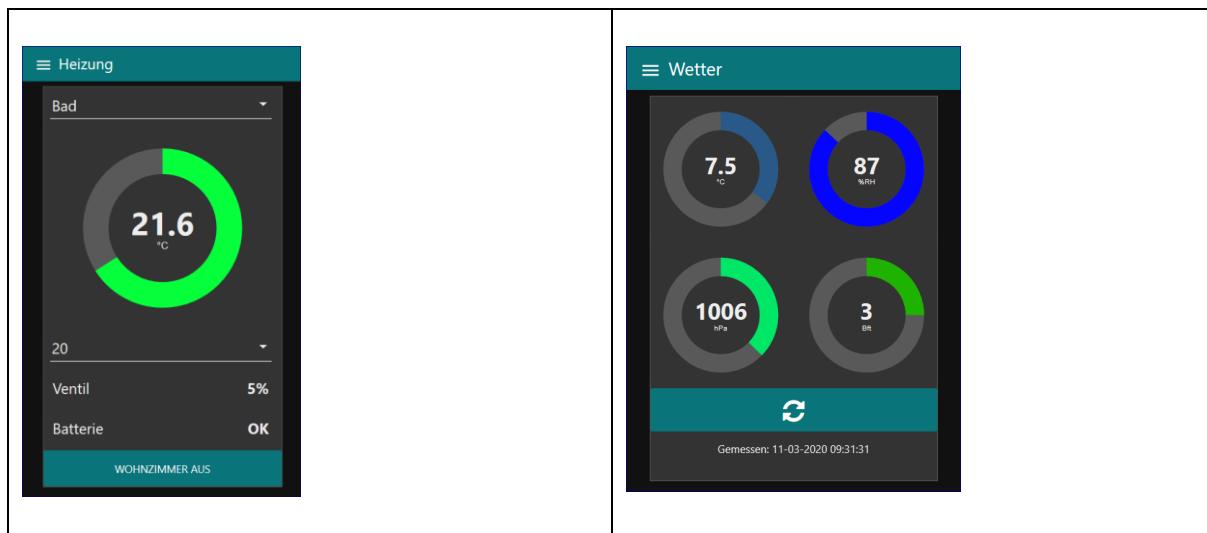
To control the Domoticz Home Automation System, running on a Raspberry Pi 3B+, via a browser-based application (WebUI, built with Node-RED).

- Easy-to-use User Interface accessible from any device capable of running a browser (smartphones, tablets, PC, SBC, TV)
- Control heating (HomematicIP) & lights (Philips Hue)
- Information only for weather data, key dates, and status postbox
- ... and more to follow ...
- ✓ The goal has been to develop a prototype first, i.e. workout the concept with initial functionality, to be able to enhance or modify the application further.
- ✓ This solution is also meant as a trigger for ideas or as a suggestion building an alternate User Interface to access this Home Automation & Information System.
- ✓ The User Interface is translated to the German language.
- ✓ In the meantime, this app has become main User Interface (using the dark theme).
- ✓ **Detailed information** about this app can be found on [GitHub](#).

**Screenshot** - example application accessed from a browser on an Android phone.  
(Dashboard theme: Light (default))

Select thermostat device. The actual temperature is shown as a donut. The setpoint is selected from a dropdown with fixed values. Info about the valve and the battery is also given. The data is refreshed every 30 seconds.	Select a light device from a dropdown. The intensity is set by a slider with indicator. The light can be switched on / off.	Weather information in donuts: Temperature, Humidity, Pressure, Wind. The data is refreshed every 5 minutes or via the refresh button.	Check the status of the postbox. Info is the flap has been opened. The state can be reset via button or refreshed. The battery voltage is monitored.	Various information starting with daylight, followed by important dates (from a roomplan) and lastly the version of this application.
---	---	--	--	---

**Screenshot** – example application accessed from browser running Windows 10 device.  
(Dashboard theme: Dark, two function selected)



# Web UI Quick Access Mobile

## Purpose

A Domoticz Customized Web User Interface, accessible from any browser, as a responsive, mobile-first front-end.

- Inform from or change selected devices which are frequently used.
- Libraries & styles stored locally (instead using Content Delivery Network):
  - [jQuery](#) v3.5.1 - Copyright JS Foundation and other contributors - [License](#)
  - [Bootstrap](#) v4.3.1 - Copyright 2011-2019 The Bootstrap Authors - [License](#)
  - [Font Awesome](#) Free 5.14.0 by @fontawesome - [License](#)  
(Icons: CC BY 4.0, Fonts: SIL OFL 1.1, Code: MIT License)
  - Versions are subject to change
- Use Domoticz roomplans & device properties to setup the HTML interface.
- Runs on the Domoticz Production system.
- Has been developed for personal use only.

The application is called **Quick Access Mobile** (QAM).

### Be Aware

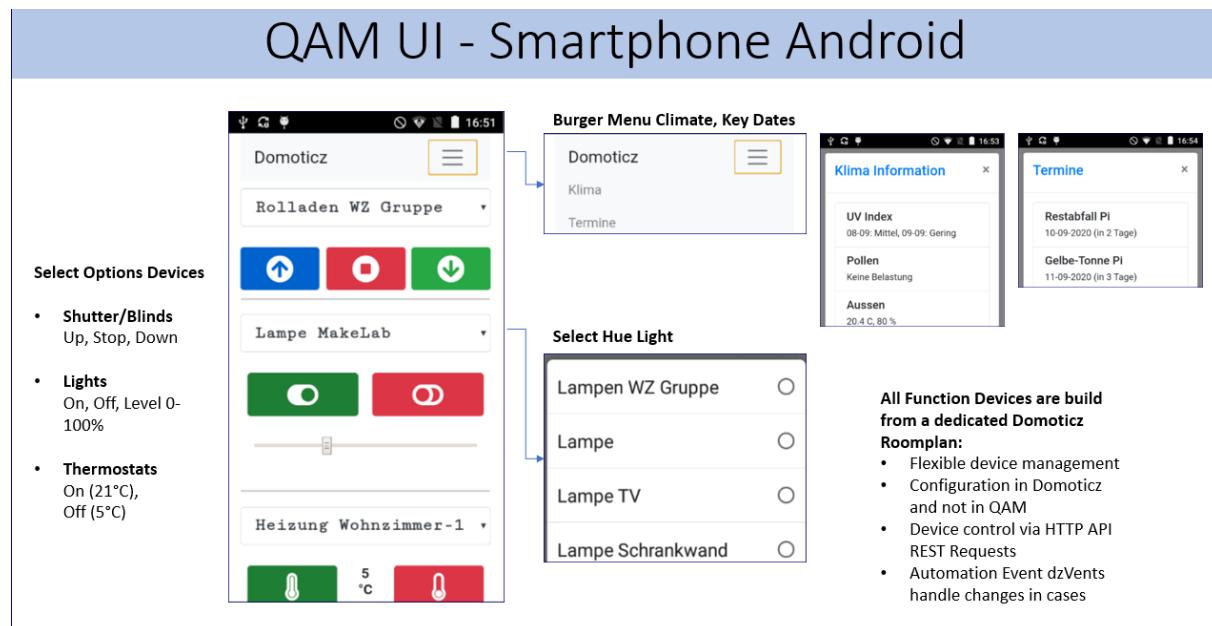
The next version of QAM has been developed and is published [here](#).

This chapter will not be updated, checkout the link for the latest version.

## Solution

A brief description on the solution.

Best to show a screenshot first of the application and outline the concept.



On the smartphone, the application (HTML page) is accessed from a webbrowser app with url:

```
http://domoticz-ip:port/7qam/index.html
```

## User Interface (UI)

The UI is using & built with:

- Bootstrap mobile-first flexbox grid layout with 12 columns and related classes (Note: also used by Domoticz, checkout the www folder).
- Font Awesome Icons for buttons
- Navigaion bar with Burger Menu with functions Climate (“Klima”) & Key Dates (“Termine”) which are shown in a modal dialog.
- Functions to control Shutter/Blinds, Lights and Thermostats using dropdowns to select a device and buttons or slider to change the device data in Domoticz.

### *Example HTML Code Navigation Bar*

```
<div class="container-fluid bg-light">
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Domoticz</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
    <div class="navbar-nav">
      <li class="nav-item"><a id="nav-item-climate" class="nav-link" onclick="showInformationDialog(IDX_ROOMPLAN_CLIMATE, 'nav-item-climate')"></a></li>
      <li class="nav-item"><a id="nav-item-keydates" class="nav-link" onclick="showInformationDialog(IDX_ROOMPLAN_KEYDATES, 'nav-item-keydates')"></a></li>
    </div>
  </div>
</nav>
</div>
```

### *Note*

The navigation bar has two entries defined as list items with anchor id's populated from room plan properties. When clicking on the item link, a modal dialog is shown which the device properties for the selected room plan.

## HTML Index Page

The HTML index page “index.html” and required libraries are stored on the Domoticz Production system (Raspberry Pi)

Folder	Files
~domoticz/www/qam	index.html basic.html (basic example page)
~domoticz/www/qam/css	Styles from the Bootstrap library
~domoticz/www/qam/js	JavaScript from the jQuery and Bootstrap libraries.
~domoticz/www/qam/webfonts	Font Awesome library

The libraries are download from the respective websites download section and included in the HTML HEAD section.

### *Example HTML Index Page HEAD Section*

```
<!-- INCLUDES = REMOTE but with LOCAL Bootstrap & jQuery -->
```

```
<link rel="stylesheet" href="css/bootstrap.css">
<link rel="stylesheet" href="css/all.min.css">
<script src="js/jquery.js"></script>
<script src="js/bootstrap.min.js"></script>
<script src="js/bootstrap.bundle.min.js"></script>
```

### Note

Example to include the libraries jQuery, Bootstrap, Font Awesome from CDN – there are many ways to do so:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@1.16.0/dist/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

## Room Plans

The layout has Burger Menu Dialogs and Functions, which use Domoticz Devices build from dedicated Domoticz Roomplans:

- Flexible plan & device management (add, change, delete, rename)
- Configuration in Domoticz and not in QAM
- Device control via HTTP API REST Requests
- Automation Event dzVents handle changes in cases

The entries are populated with room plan data at start/refresh of the HTML index page.

The screenshot shows the QAM UI - Concept Roomplans interface. In the center is a dialog box titled "TITLE" containing "QAM-Information" and "QAM-Switch-1". It features two large green and red buttons with circular icons. Below the buttons is the text "13:55:02, setSelectOptions, OK (Devices)" and the timestamp "v20200911". To the left is a sidebar titled "QAM-Information" with a list of entries: "QAM-Info-1 Hello World" and "QAM-Info-2 Hello World". At the bottom of this sidebar are buttons for "Schließen", "Show 25 entries", and "Idx" with values 34 and 35. To the right is another sidebar titled "Domoticz Roomplans" with a list of entries: "4 QAM-Information" and "5 QAM-Switches". At the top right of the main area is a yellow speech bubble containing the text "Basic Example". Arrows point from the sidebars to the central dialog, indicating they are connected components.

### Note

The layout of the basic example.

In the middle, the UI accessed from web browser via

```
http://domoticz-ip:port/qam/basic.html
```

In Domoticz 4 devices are used and assigned to 2 room plans, one for the Burger Menu (Information only modal dialogs) and the other for switches.

The room plan property “Name” is used for the Burger Menu item entry and for the modal dialog title.

The room plan devices are used for the content of the model information dialog header (device property “Name”) and content (device property “Data”).

Changing the room plan properties, i.e. the name or list of devices, will be shown after refreshing the HTML page.

#### *Example JavaScript Room Plan Definition*

```
// Define the idx of the room plans
const IDX_ROOMPLAN_CLIMATE = 23;
const IDX_ROOMPLAN_HUE = 20;
const IDX_ROOMPLAN_KEYDATES = 4;
const IDX_ROOMPLAN_SOMFY = 24;
const IDX_ROOMPLAN_THERMOSTATS = 21;
```

## Communication

The communication from and to Domoticz is via [HTTP API requests](#) using JavaScript.

The URLs have placeholders defined (i.e. #IDX#, #SWITCHCMD#), which are replaced prior calling.

The HTML layout has selectors which content is replaced by the HTTP API response.

The jQuery library is used for the JSON-encoded data from/to the Domoticz server using a GET HTTP request - getJSON().

#### *Examples URL JavaScript*

```
const URL_DOMOTICZ = "http://domoticz-ip:port/json.htm?";
// Get all devices property
var URL_GET_ALL_DEVICES = URL_DOMOTICZ + "type=devices&used=true&displayhidden=0";
// Get single device properties
var URL_GET_DEVICE = URL_DOMOTICZ + "type=devices&rid=#IDX#";
// Get all plans
var URL_GET_PLANS = URL_DOMOTICZ + "type=plans&order=name&used=true";
// Get all devices for a plan, i.e. roomplan 23 for the room Climate
var URL_GET_PLAN_DEVICES = URL_DOMOTICZ + "type=devices&plan=#IDX#";
// CMD = Somfy switch command Off, Stop, On
var URL_SWITCH_SOMFY = URL_DOMOTICZ +
"type=command&param=switchlight&idx=#IDX##switchcmd=#SWITCHCMD#";
// CMD = Somfy Group switch command level 10 (Open), 20 (Stop), 30 (Close)
var URL_SWITCH_SOMFY_GROUP = URL_DOMOTICZ +
"type=command&param=switchlight&idx=#IDX##switchcmd=Set%20Level&level=#LEVEL#";
// CMD = Hue switch command Off, Set%20Level=NN, On
var URL_SWITCH_HUE = URL_DOMOTICZ + "type=command&param=switchlight&idx=#IDX##switchcmd=#SWITCHCMD#";
// CMD = Thermostat switch state ON or OFF handled by customevent raspmatic_thermostat_onoff. JSON data
key : {"idx":123, "state":0|1}
var URL_THERMOSTAT_ONOFF = URL_DOMOTICZ +
"type=command&param=customevent&event=raspmatic_thermostat_onoff&data=#DATA#";
```

#### *Example Set Burger Menu Item from Room Plan*

Define the HTML code with id=”nav-item-climate”

```
<li class="nav-item">
<a id="nav-item-climate"
class="nav-link"
onclick="showInformationDialog(IDX_ROOMPLAN_CLIMATE, 'nav-item-climate')">
```

```
</a>
</li>
```

### In JavaScript jQuery document ready function set the plan name(s)

```
$(document).ready(function() {
    ...
    // Set the name of the various burger menu items
    setPlanName(URL_GET_PLANS, "nav-item-climate", IDX_ROOMPLAN_CLIMATE);
    ...
});
```

The JavaScript function to get the plan data from Domoticz via HTTP API request to set the room plan property Name as Burger Menu Item

```
/*
 * Get & set the name of a plan using the plan idx.
 * Example: setPlanName(URL_GET_PLANS, "nav-item-information", 4)
 */
function setPlanName(requestUrl, selectorPlan, planIdx)
{
    console.log("setPlanName: " + requestUrl);
    $.getJSON(
        requestUrl,
        {
            format: "json"
        },
        function(data)
        {
            name = "UNKNOWN";
            if (typeof data != 'undefined')
            {
                // Get the plan array from JSON result
                plans = data.result;
                // Search for the plan to get the name from the key "Name"
                plans.forEach (function (plan, index) {
                    if (plan.idx == planIdx) {
                        name = plan.Name;
                        document.getElementById(selectorPlan).innerHTML = name;
                    }
                });
                setStatus("setPlanName, " + planIdx + ", " + name + ", " + data.status);
            }
            else
            {
                document.getElementById(selectorPlan).innerHTML = name;
            }
        }
    );
}
```

#### *Note*

It is possible to run several versions of QAM accessing different Domoticz systems. Just define several HTML indices files with different Domoticz IP addresses as defined in the JavaScript Constant URL\_DOMOTICZ.

#### **Example**

The various QAM HTML index files are stored on the Domoticz production system to access the Domoticz production and development systems.

#### Location

```
/home/pi/domoticz/www/qam/index-prod.html
```

#### URL

```
const URL_DOMOTICZ = "http://domoticz-ip:port/json.htm?";
```

#### Access

```
http://domoticz-ip:port/qam/index-prod.html
```

**Location**

```
/home/pi/domoticz/www/qam/index-dev.html
```

**URL**

```
const URL_DOMOTICZ = "http://domoticz-ip:port/json.htm?";
```

**Access**

```
http://domoticz-ip:port/qam/index-dev.html
```

## Get Started

To get started, recommend using the basic example.

### Step 1

Create a folder on the Domoticz system:

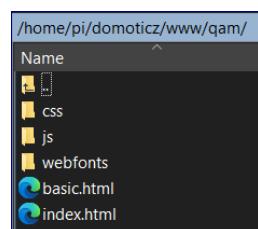
```
<domoticz-path>/www/qam
```

Example Raspberry Pi:

```
/home/pi/domoticz/www/qam
```

### Step 2

Extract the archive “qam.zip” to the new created folder, resulting in folder structure (example Raspberry Pi):



### Step 3

Create or Select Domoticz Room Plan(s) and note the Idx of the room plans.

Recommend starting with 2 room plans.

Get the Domoticz IP address, i.e. 192.168.1.179 and port, i.e. 8080.

### Step 4

Define the Domoticz IP address & Port and the room plans in the HTML page “basic.html”.

Change the JavaScript constants:

```
const URL_DOMOTICZ = "http://domoticz-ip:port/json.htm?";
const IDX_ROOMPLAN_QAM_INFORMATION = 4;
const IDX_ROOMPLAN_QAM_SWITCHES = 5;
```

#### Note

If also changing the constant names, ensure to set those on the document.ready function as well

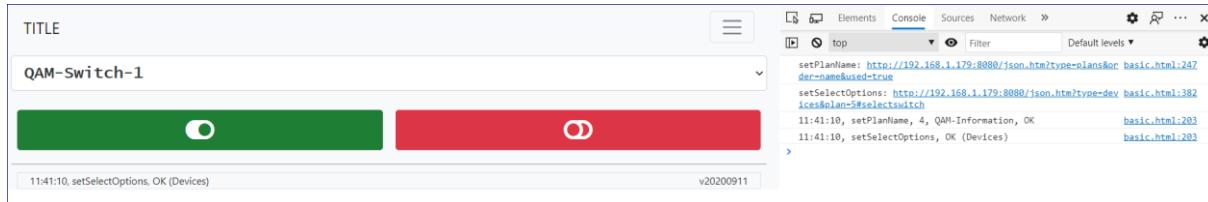
```
// Set the name of the various burger menu items
setPlanName(URL_GET_PLANS, "nav-item-information", IDX_ROOMPLAN_QAM_INFORMATION);
// Define the select options
setSelectOptions(IDX_ROOMPLAN_QAM_SWITCHES, "selectswitch");
```

### Step 5

Open the URL in a webbrowser, start from the development device.

```
http://domoticz-ip:port/qam/basic.html
```

Press F12 and checkout the console for information/errors.



#### Note

The debug information is set by the flag DEBUG in the JavaScript section:

```
const DEBUG = true; // true|false;
```

# Web Radio (Volumio)

## Purpose

To listen to & control, via Domoticz Dashboard, Web Radio provided by [Volumio](#) - the Open Audiophile Music Player.

### Features to control Volumio Web Radio

- Set Volumio status Play or Stop.
- Show the title of the current song, updated real time or refresh via switch.
- Set Volume 0 – 100 or switch On | Off.
- Bookmark the current song to the favorites list.
- Indicate status OFF, PLAY, STOP.
- Control via Domoticz Dashboard with dedicated Room Plan.
- Handle Volumio offline state without spoiling logs with error messages.
- Solution must be open for enhancements.

Of course, many options to enhance, but the intention is not to replace the Volumio Web UI.

### Screenshot Domoticz Dashboard with Room Volumio

The screenshot shows the Domoticz Dashboard with the following sections:

- Light/Switch Devices:**
  - Volumio Play: On
  - Volumio Refresh: On
  - Volumio Volume: A slider set to approximately 80, with On/Off buttons available.
  - Volumio Bookmark: On
- Utility Sensors:**
  - Volumio Current Song: This is the song title
  - Volumio Favorites: This is the last bookmarked song title
  - Volumio Status: PLAY (green triangle icon)

Volumio Current Song List sample	Volumio Favorites List sample
Show 25 entries Date 2019-02-10 12:18:25 song title 2019-02-10 12:14:04 song title 2019-02-10 12:08:38 song title 2019-02-10 12:04:18 song title	Show 25 entries Date 2019-02-10 11:45:05 song title 2019-02-10 11:33:13 song title 2019-02-10 11:18:31 song title 2019-02-10 11:15:55 song title
<b>Volumio Status PLAY = switch Volumio Play set to On</b>	
Volumio Status	PLAY ▲
<b>Volumio Status STOP = if the server is started or if switch Volumio Play is set to Off</b>	
Volumio Status	STOP ▲
<b>Volumio Status OFF = server not reachable (network) or server turned off (shutdown)</b>	
Volumio Status	OFF ▲

## Volumio Setup

A Volumio server has been setup on a [Pine A64](#) (one of the first devices shipped back 2016).

Download the Volumio 2 Pine A64 SD Image from [here](#) (see additional [info](#)).

Image version: Volumio Digital Audio Player [2.383-2018-03-17].

The Pine A64 is connected to an external amplifier and not using a DAC (<TODO> planned for future).

### Get Started

To get started, read the [Volumio manual](#) first.

- Connect to Volumio from browser with URL **volumio.local**.  
If not working, determine the server IP address via f.e. the connected router interface.
- Configuration done via the Volumio Web UI Settings.
- Set fixed IP addresses for WLAN and Ethernet.
- Enable [SSH](#) (required to access the server via WinSCP or Putty).
- Created an USB with loads of MP3 and plugin to the USB port of the PineA64.
- Do shutdown the server via the Volumio menu Settings > Shutdown.

## Solution Options

Experimented with solutions Domoticz dzVents and Node-RED using the Volumio API or the MPD (Music Player Deamon) commands.

My preferred solution is to go with Domoticz **dzVents Lua scripting** ([reference](#)).

### Automation Scripts

Identified two challenges to resolve:

1. The dzVents script event timer updates every 1 minute which means that if a new song is playing, the displayed current song is still the previous and could also result in bookmarking the wrong song.  
*Resolved:* Adding a switch enabling to update current state.
2. If the Volumio is not reachable, the Domoticz Error Log is getting spoiled with messages.  
*Resolved:* If Volumio state is OFF, set play mode to OFF which does not trigger the timer to request Volumio status update.

## Node-RED

### *Option 1*

with the libraries [MPD](#) and [Advanced Ping](#) went well until handling volume changes – did not work via the *setvol* command. There might be more issues with MPD commands.

### *Option 2*

Decided to use Node-RED to control Volumio via REST API commands as documented [here](#), thus to stick to what's defined (supported) by Volumio.

The available subset of Volumio REST API commands are suitable for this solution.

The communication between Volumio and Domoticz is based on MQTT messaging.

### To know

This solution does not update the status of Domoticz Devices if Volumio is controlled via the Volumio Web Interface or any other (like an App).

<TODO>For future solutions, consider implementing in Volumio MQTT to be handled by Domoticz.

# Solution Automation Script

## Domoticz Configuration

### Devices

<b>Idx</b>	<b>Name</b>	<b>Type</b>	<b>SubType</b>	<b>Function</b>
145	Volumio Bookmark	Light/Switch	Switch <i>Switch Type:</i> Push On Button	Switch ON adds (bookmarks) the current song playing to Favorites device (List)
146	Volumio Favorites	General	Text	Favorite songs
148	Volumio Volume	Light/Switch	Switch <i>Switch Type:</i> Dimmer	Set the volume between 0-100
149	Volumio Play	Light/Switch	Switch <i>Switch Type:</i> On/Off	Set Play (switch On) or Stop (switch Off)
150	<i>Volumio Shutdown</i>	<i>Light/Switch</i>	<i>Switch</i>	<i>NOT USED</i> <i>&lt;TODO&gt;Find solution how to shut down.</i>
151	Volumio Status	General	Alert	Show the status of Volumio: OFF (Red) STOP (Yellow) PLAY (Green)
155	Volumio Refresh	Light/Switch	Switch <i>Switch Type:</i> Push On Button	Updates the current song and the status.

## Roomplan

Created a roomplan *Volumio* with the previous defined devices (Domoticz GUI Setup > More Options > Plans > Roomplan).  
The roomplan (IDX=5) is used as the Domoticz Dashboard to control Volumio.

Roomplan Volumio	Dashboard with Roomplan Volumio																		
<p>5 Volumio</p> <p>Showing 1 to 5 of 5 entries</p> <p>Edit    Delete</p> <p>Devices (Select Plan first to Edit...)</p> <p>Show <input type="button" value="▼"/> entries</p> <table border="1"> <thead> <tr> <th>Idx</th> <th>Name</th> </tr> </thead> <tbody> <tr><td>149</td><td>Volumio Play</td></tr> <tr><td>155</td><td>Volumio Refresh</td></tr> <tr><td>148</td><td>Volumio Volume</td></tr> <tr><td>144</td><td>Volumio Current Song</td></tr> <tr><td>145</td><td>Volumio Bookmark</td></tr> <tr><td>146</td><td>Volumio Favorites</td></tr> <tr><td>151</td><td>Volumio Status</td></tr> <tr><td>150</td><td>Volumio Shutdown</td></tr> </tbody> </table> <p>Showing 1 to 8 of 8 entries</p>	Idx	Name	149	Volumio Play	155	Volumio Refresh	148	Volumio Volume	144	Volumio Current Song	145	Volumio Bookmark	146	Volumio Favorites	151	Volumio Status	150	Volumio Shutdown	<p><b>Light/Switch Devices</b></p> <ul style="list-style-type: none"> <li>Volumio Play <input checked="" type="button" value="On"/> <input type="button" value="Off"/></li> <li>Volumio Refresh <input type="button" value="On"/></li> <li>Volumio Volume <input checked="" type="button" value="On"/> <input type="button" value="Off"/></li> <li>Volumio Bookmark <input type="button" value="On"/></li> </ul> <p><b>Utility Sensors</b></p> <ul style="list-style-type: none"> <li>Volumio Current Song This is the song title</li> <li>Volumio Favorites This is the last bookmarked song title</li> <li>Volumio Status <input type="button" value="PLAY"/> </li> </ul>
Idx	Name																		
149	Volumio Play																		
155	Volumio Refresh																		
148	Volumio Volume																		
144	Volumio Current Song																		
145	Volumio Bookmark																		
146	Volumio Favorites																		
151	Volumio Status																		
150	Volumio Shutdown																		

## Volumio API Commands

The commands are used in nodes as payload, either incoming or outgoing. To learn which commands are available and how to use, check out the Reference documentation.

### Volumio ([Reference](#))

- volumio.local/api/v1/getstate
- volumio.local/api/v1/commands/?cmd=play&N=0
- volumio.local/api/v1/commands/?cmd=stop
- volumio.local/api/v1/commands/?cmd=volume&volume=NN

## Automation Scripts

The dzVents Lua script events are created and maintained via Domoticz GUI > Setup > More Options > Events.

Script Name	Purpose
volumio_getstate	Obtain and update the devices Volumio Status and Volumio Current Song playing.
volumio_play	Switch the device Volumio Play to set the play mode to PLAY or STOP.
volumio_setvolume	Set the device Volumio Volume between 0-100 or switch ON   OFF.
volumio_bookmark	Switch device Volumio Bookmark to add (bookmark) the current song to the Volumio Favorites device.

The scripts are

- rather comprehensive and can be downloaded [here](#).
- well documented to understand the logic.

## Notes

### Track Played

In the dzVents Lua script *volumio\_play*, the Volumio play command uses the ordinal track number 0, i.e. the first track of the queue:

```
volumio.local/api/v1/commands/?cmd=play&N=0
```

If there are more tracks, i.e. web radios defined then change the track number or define a User Variable (DEF\_VOLUMIO\_TRACK) holding the track number to play.

Idx	Variable name	Variable type	Current value
11	DEF_VOLUMIO_TRACK	Integer	0

The User Variable value can be added to the play command ...play&N=

Event name: volumio\_play

(extract)

```
local IDX_VOLUMIOPLAYSWITCH = 149
local IDX_VOLUMIOSTATUS = 151
local IDX_DEF_VOLUMIO_TRACK = 11

local cmdplay = 'play&N=' -- track queue number is added from user variable
local cmdstop = 'stop'
local cmd

if (domoticz.devices(IDX_VOLUMIOPLAYSWITCH).state == 'On') then
    cmd = cmdplay .. tostring(domoticz.variables(IDX_DEF_VOLUMIO_TRACK).value)
    domoticz.log(cmd, domoticz.LOG_INFO)
    domoticz.devices(IDX_VOLUMIOSTATUS).updateAlertSensor(1, MSGVOLUMIOPLAY)
else
    cmd = cmdstop
    domoticz.devices(IDX_VOLUMIOSTATUS).updateAlertSensor(2, MSGVOLUMIOSTOP)
end
```

## Logging

Check from time-to-time the log of the devices and clean up.

Examples to clear log entries (recommended) because every change is logged:  
Volumio Status, Volumio Current Song.

## Refresh

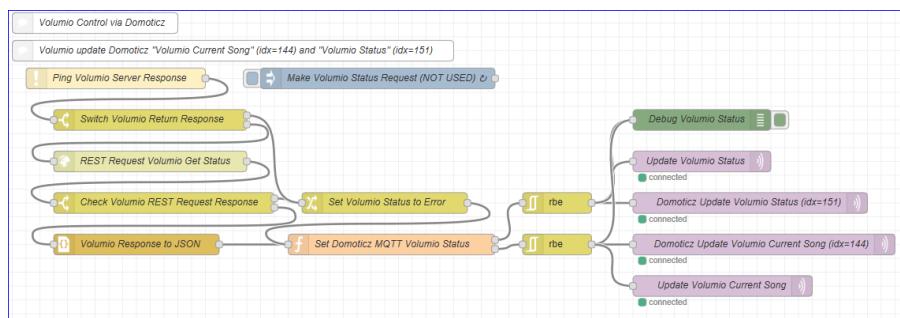
### Switch Volumio Refresh

Prior bookmarking the current song, press *Refresh* first to ensure the current song played is actual (because of the Domoticz one-minute refresh interval).

The same applies, if not checked the Dashboard for a while – press Refresh to update.

### Node-RED Option

Use a Node-RED flow to ping, every 10 seconds, the Volumio server and update the status on the Dashboard.



## Solution Node-RED

### Domoticz Configuration

#### Devices

Created following Virtual Sensors with function

Idx	Name	Type	SubType	Function
145	Volumio Bookmark	Light/Switch	Switch	Set (switch On) the current song playing to Favorites
146	Volumio Favorites	General	Text	Favorite songs
148	Volumio Volume	Light/Switch	Switch	Set the volume between 0- 100
149	Volumio Play	Light/Switch	Switch	Set Play (switch On) or Stop (switch Off)
150	Volumio Shutdown	Light/Switch	Switch	NOT USED
151	Volumio Status	General	Alert	Show the status: OFF (Red), STOP (Yellow), PLAY (Green)

#### Note

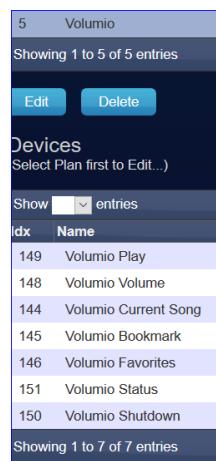
<TODO>The device *Volumio Shutdown* is not used – seeking for a proper solution to shut down the Volumio server.

#### Roomplan

Created a roomplan *Volumio* with the previous defined devices (Domoticz GUI Setup > More Options > Plans > Roomplan).

The roomplan is used as the Domoticz Dashboard (see earlier screenshot) to control Volumio.

#### Screenshot Roomplan number 5 named Volumio with 7 devices



#### Volumio API Commands

The commands are used in nodes as payload, either incoming or outgoing. To learn which commands are available and how to use, check out the Reference documentation.

## Volumio

([Reference](#))

- volumio.local/api/v1/getstate
- volumio.local/api/v1/commands/?cmd=play
- volumio.local/api/v1/commands/?cmd=stop
- volumio.local/api/v1/commands/?cmd=volume&volume=NN

The getstate command is triggered every 5 seconds by an *Inject Node*.

The response is either a string in case the Volumio server is not reachable (which is converted to a JSON string with status error) or a JSON string containing current song information.

## Domoticz

([Reference](#))

- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":4, "svalue": "OFF"};
- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":1, "svalue": "PLAY"};
- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":2, "svalue": "STOP"};
- {"command": "udevice", "idx": idxvolumiocurrentsong, "svalue": currentsong};
- {"command": "switchlight", "idx": idxvolumiobookmark, "switchcmd": "Off"};
- {"command": "udevice", "idx": idxvolumiofavorites, "nvalue":0, "svalue": currentsong};

The listed message payloads are created in *Function Nodes* with variables, like the idx names and current song (i.e. var idxvolumiostatus = 151;).

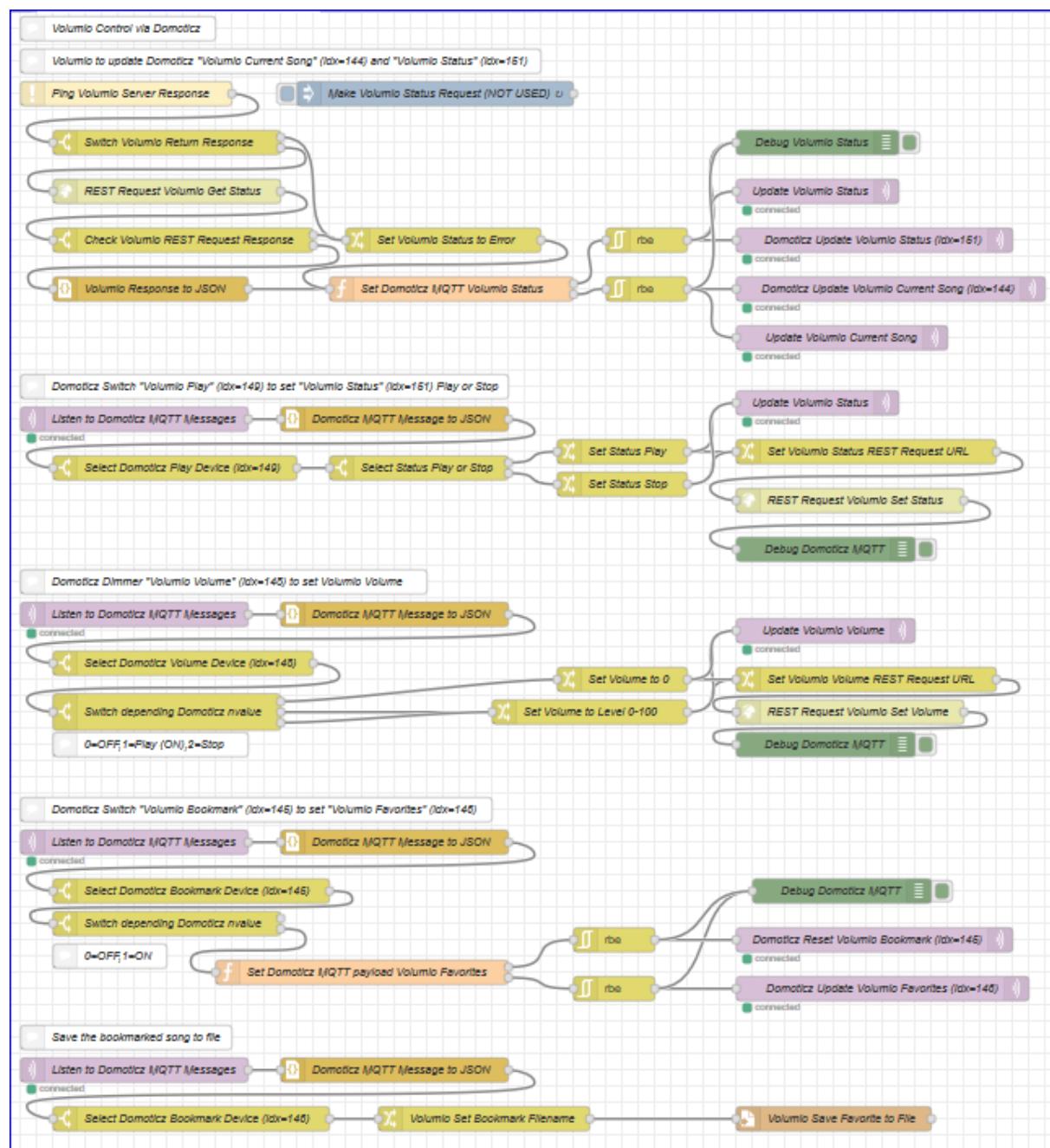
The output of the *Function Node* goes to a *MQTT Out Node*, which connects to the Domoticz MQTT broker (localhost:1883) and publishes the messages.

## Node-RED Flows

### Volumio Control

The comprehensive Volumio Control flow with subflows:

- Volumio to update Domoticz "Volumio Current Song" (IDX=144)
- Domoticz Switch "Volumio Play" (IDX=149) to set "Volumio Status" (IDX=151) PLAY or STOP
- Domoticz Dimmer "Volumio Volume" (IDX=148) to set Volumio Volume
- Domoticz Switch "Volumio Bookmark" (IDX=145) to set "Volumio Favorites" (IDX=146)
- If Volumio shutdown or not reachable set "Volumio Status" (IDX=151) OFF

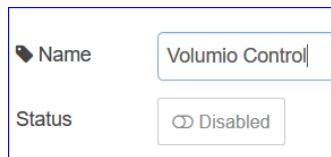


## Notes

- Tried to set meaningful nodes names for easier understanding of the flows.
- The comments nodes and the function nodes contain additional information.
- For testing commands used *Inject nodes*, which are removed from the final flow.
- Flow development:  
Opened in a browser following windows (tabs):
  - Domoticz Dashboard
  - Domoticz Log  
(Domoticz Setup > Log)
  - Node-RED Development
  - Optional PuTTY session showin the Node-RED log  
(console command node-red-log)
- Flow development:  
Used Debug nodes to check results or learn message content (esp. MQTT).  
Give the Debug nodes meaning full names to easier read the log (Node-RED Debug tab).
- To check if the Volumio server is reachable, the Ping Node is used. Depending response, the Volumio REST request to get the status is “making”.

## Disable Flow

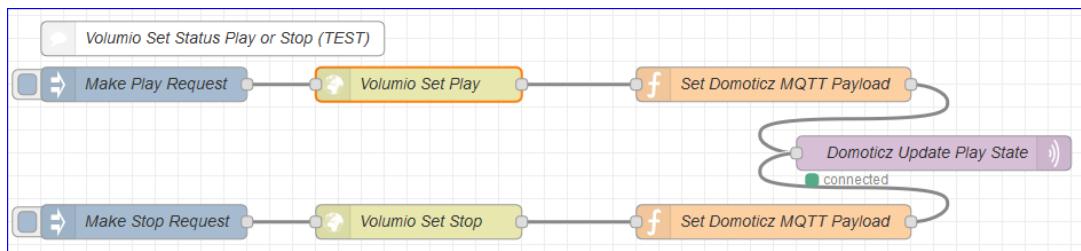
The Node-RED flow *Volumio Control* can be enabled and disabled by clicking on the Tab *Volumio Control*:



If the flow is enabled again, then the Volumio Status is updated depending reachability of the Volumio server (OFF, PLAY or STOP).

## Test Example

Use the Inject Node to set the Volumio status to Play or Stop and update the Domoticz device *Volumio Status*.



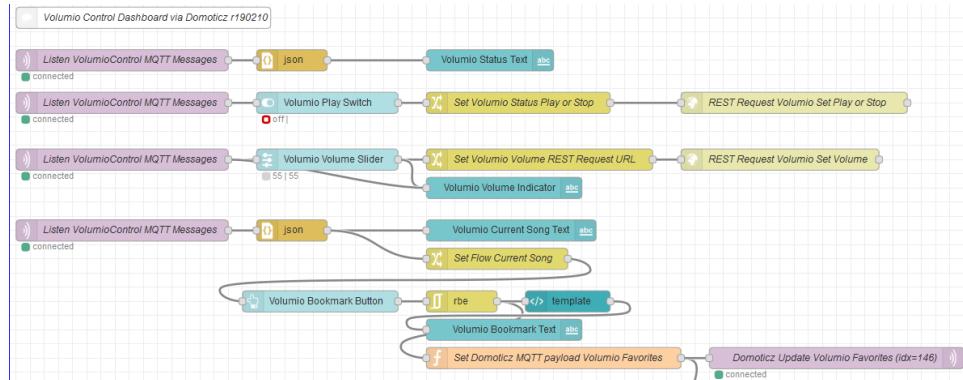
## Note

This flow can be simplified using Change Node to set the status to then trigger Volumio and update the Domoticz Device.

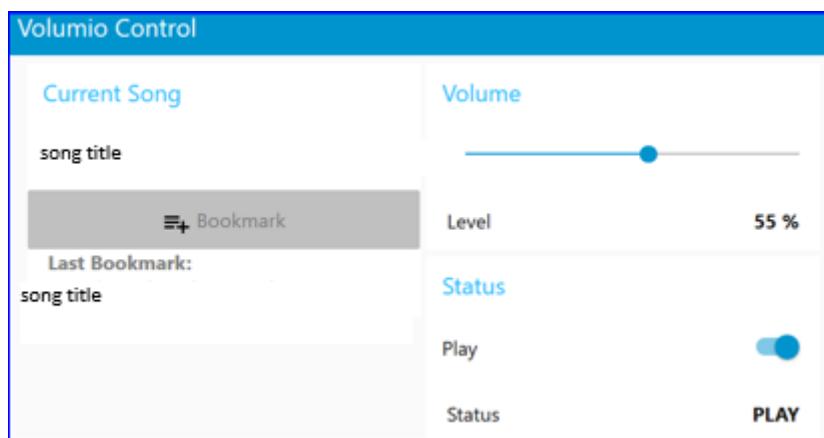
## Volumio Control Dashboard

For Node-RED, [Dashboard UI](#) nodes are available, which could be used to create a Domoticz Alternate Dashboard (accesses via browser).

Created a flow *Volumio Control Dashboard*, to explore the use of Dasboard UI nodes.



Displayed in browser with <http://domoticz-ip:1880/ui>



### Description (Dasboard UI Node Types)

- Display the current song playing (ui\_text)
- Button to bookmark the current song (ui\_button)
- Show the last bookmarked song (ui\_text)
- Set the volume 0-100 (ui\_slider)
- Show volume level (ui\_text)
- Set status play or stop (ui\_switch)
- Show the status OFF,PLAY,STOP (ui\_text)

## Dashboard Properties

Tab: Volumio Control, Groups: Current Song & Bookmark, Volume, Status

Used the flow *Volumio Control* to create dedicated MQTT messages (prefix volumiocontrol), which are used by the flow *Volumio Control Dashboard* to set properties.

MQTT Message	Properties
<b>Topic</b>	<b>volumiocontrol/status</b>
Payload Sample	{"command":"udevice","idx":151,"nvalue":1,"svalue":"PLAY"}
Usage	To set the status of Volumio in a Dashboard ui_text node:
<b>Topic</b>	<b>volumiocontrol/currentsong</b>
Payload Sample	{"command":"udevice","idx":144,"svalue":"Current Song Title"}
Usage	To set the text of the current song playing in a Dashboard ui_text node: Value format: {{msg.payload.svalue}}
<b>Topic</b>	<b>volumiocontrol/volume</b>
Payload Sample	NN containing the level
Usage	To set the volume level of the slider and text of the indicator. Value format: {{msg.payload}} &percnt;

## Favorites List

The bookmarked songs are added to the favorites device *Volumio Favorites* (IDX=146) and are listed (log).

The list is used to follow up on favorite song via other media.

Show 25 entries	
Date	
2019-02-10 11:45:05	song title
2019-02-10 11:33:13	song title
2019-02-10 11:18:31	song title
2019-02-10 11:15:55	song title

The favorite songs are stored in the Domoticz database domoticz.db, located in folder /home/pi/domoticz/ on the Raspberry Pi.

The table LightingLog contains the records.

Access via SQLite Tool:

To select the favorite songs, use SQL Query command like

```
SELECT * FROM LightingLog WHERE DeviceRowID=146;
```

This enables to create applications using these records from the database.

<TODO>Brainstorm a Favorite Songs application. Thoughts: 1) Node-RED with the SQLite3 Library to select and list in a customised widget. 2) Lazarus or B4J Desktop application.

# Wind (TFA 30.3168)

## Purpose

To display the Wind Direction and Wind Speed obtained from the Anemometer.

## Solution

The **Wind Sensor Device (Wind Meter TFA Dostmann 30.3168)** “Windmesser” measures wind direction, wind speed and temperature.

The device is connected to the hardware RFXTrx433E (see [RFXCOM RFXtrx433E](#)):  
28, RFXtrx433e, Windmesser, Wind, TFA, 236.00;SW;9;12;12.4;12.4  
(Idx, Hardware, name, Type, SubType, Data).

The measured temperature is not used as provided by a **Temperature & Humidity Device (TFA Dostmann / Wertheim TS34C)**, which means that (only) the wind direction and wind speed are used and displayed on the dashboard.

The solution is to create a Virtual Sensor Type Wind named “Wind” and assign the wind direction and wind speed from the Wind Sensor.

### Wind Virtual Sensor Device List Entry (IDX=117)

117	VirtualSensors	140C5	1	Wind	Wind	WTGR800	0;N;0;0;0;0
-----	----------------	-------	---	------	------	---------	-------------

### Weather Widget Wind Virtual Sensor



#### Note

The data displayed is provided by the Event Lua Script `script_device_wind`.

### Wind Sensor Device with actual data (IDX=28)

28	RFXtrx433e	9D16	0	Windmesser	Wind	TFA	304.00;WNW;12;12;21.2;21.2
----	------------	------	---	------------	------	-----	----------------------------

The data contains: WB,WD,WS,WG,T,TW

- WB = Wind bearing (0-359)
- WD = Wind direction (S, SW, NNW, etc.)
- WS = 10 \* Wind speed [m/s]
- WG = 10 \* Gust [m/s]
- T = 21.2 = Temperature
- TW = 21.2 = Temperature Windchill

## Automation Script

To update the wind direction and wind speed from the Wind Sensor Device data and assign to the Wind Virtual Sensor.

This is done via a dzVents Lua script event using the Domoticz Event Editor.

Event name: wind\_update

```
--[[[wind_update.lua
If a value of the device Windmesser (idx=28) changes,update the value (Wind direction & speed) of
the Virtual Sensor Wind named Wind (ix=117). Use print(device.dump()) ONLY ONCE to get the properties.
]]]

-- Idx of the devices
local IDX_WINDMESSER = 28;
local IDX_WIND = 117;

-- Event handling changes of the Windmesser device
return {
  on = {
    devices = {
      IDX_WINDMESSER
    }
  },
  execute = function(domoticz, device)
    -- print(device.dump())
    -- domoticz.log('Device ' .. device.name .. ' changed ', domoticz.LOG_INFO)

    bearing = device.direction
    direction = device.directionString
    speed = device.speed
    gust = device.gust
    temperature = device.temperature
    chill = device.chill
    -- Update the virtual sensor Luftdruck
    domoticz.devices(IDX_WIND).updateWind(bearing,direction,speed,gust, temperature, chill)
  end
}
```

### Domoticz Log

```
2018-09-10 16:09:12.797 (RFXtrx433e) Wind (Windmesser)
2018-09-10 16:09:12.020 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2018-09-10 16:09:12.943 Status: dzVents: Info: Handling events for: "Windmesser", value:
"281.00;W;7;13;21.2;21.2"
2018-09-10 16:09:12.944 Status: dzVents: Info: ----- Start internal script: wind_update: Device:
"Windmesser (RFXtrx433e)", Index: 28
2018-09-10 16:09:12.944 Status: dzVents: Info: Device Windmesser changed
2018-09-10 16:09:12.945 Status: dzVents: Info: ----- Finished wind_update
```

# User Variables

The User Variables defined are mainly used for Automation Events with dzVents.

In addition, used by Node-RED flows to f.e. set a new value.

Node-RED is an option, to configure User Variables other than using the Domoticz GUI or scripts.

## Syntax

Defined following naming convention syntax for User Variables:

- Prefix DEF or IDX or TH,
- Suffix = the device name, without special characters and blanks (replace by underscore),
- Prefix & suffix in UPPERCASE connected with an underscore.

PREFIX\_DEVICENAME in UPPERCASE

Prefix	Meaning	Example
DEF	Default	DEF_ALERTMSG
IDX	Device Index	IDX_ALERTMSG
TH	Threshold	TH_RPI_HDDUSAGE

### Notes

The max length of a string variable is 200 bytes.

## List

### SQL

List of User Variables selected, via the sqlite3 CLI - see [SQLite Shell](#), from the Domoticz database domoticz.db, table UserVariables:

```
$ cd domoticz
$ sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
sqlite> .open domoticz.db
sqlite> .header on
sqlite> .mode column
sqlite> SELECT * FROM UserVariables;
ID      Name          ValueType  Value   LastUpdate
-----  -----        -----    -----  -----
3       TH_RPI_TEMPERATURE 0        50      2018-09-11 19:36:34
4       TH_RPI_HDDUSAGE    0        70      2018-09-05 11:15:20
5       TH_RPI_MEMORYUSAGE 0        80      2018-09-05 11:15:47
6       IDX_ALERTMSG     0        55      2018-09-05 18:58:40
7       DEF_ALERTMSG     2        No alert 2019-07-08 14:17:57
8       TH_STOCK_RDS     1        27      2019-03-03 09:53:40
9       TH_STOCK_RDS_NOTIF 0        0       2019-02-16 10:51:07
10      TH_AMBIENT_LIGHT 0        111     2019-02-27 15:15:43
11      DEF_VOLUMIO_TRACK 0        0       2019-03-01 10:27:47
12      DEF_STOCK_RDS_COUN 0        1500    2019-05-17 11:17:39
13      DEF_SECURITY_ALARM 0        0       2020-03-16 08:55:00
14      TH_ALERTTOEMAIL    0        4       2019-11-11 07:44:44
15      TH_COFFEEAMACHINEMO 0        1100    2019-07-29 08:59:32
```

```
16      TH_POSTBOXNOTIFIER 1      1.0      2019-12-11 17:56:31
17      DEF_NOTES          2      [1]:Hello 2020-05-29 15:56:35
sqlite>
```

The Value Types are:

```
0 = Integer, e.g. -1, 1, 0, 2, 10
1 = Float, e.g. -1.1, 1.2, 3.1
2 = String e.g. On, Off, Hello
3 = Date in format DD/MM/YYYY
4 = Time in 24 hr format HH:MM
```

## Examples SQL Statements

Select	SELECT * FROM UserVariables.
Update	UPDATE UserVariables SET Value = 1 WHERE Name = "DEF_SECURITY_ALARM"

## Python

Simple Python3 script to list the User Variables.

Source: user\_variables\_list.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import urllib.request
from urllib.error import HTTPError,URLError
import sys
import json
"""

HTTP API Request (running locally on the Domoticz development device)
URL = 'http://127.0.0.1:8080/json.htm?type=command&param=getuservariablesx'
HTTP Response (JSON decodes)
{'result': [
    {'LastUpdate': '2020-02-06 11:47:03', 'Name': 'TH_AMBIENTLIGHT', 'Type': '0', 'Value': '100', 'idx': '1'},
    {'LastUpdate': '2020-02-20 10:32:18', 'Name': 'TH_TIMECONTROL', 'Type': '1', 'Value': '2.5', 'idx': '2'}],
 'status': 'OK', 'title': ' GetUserVariables'}
"""

URL = 'http://127.0.0.1:8080/json.htm?type=command&param=getuservariables'
if __name__ == "__main__":
    sys.tracebacklimit = 0
    try:
        req = urllib.request.Request(URL)
    except urllib.request.RequestException as e:
        print("Request not valid:{}".format(e.code))
        raise
    try:
        # parsing response
        response = urllib.request.urlopen(req).read()
        content = json.loads(response.decode('utf-8'))
        # print(content)
        # get the status
        if content['status'] == "OK":
            # get the result array
            result_array = content['result']
            # loop over the array and fetch the device with idx=4
            print('Name', 'Idx', 'Value', sep=';')
            for item in result_array:
                print(item['Name'], item['idx'], item['Value'], sep=';')
        else:
            print("Response Error:Check the HTTP API request URL:{}".format(URL))
    except urllib.error.HTTPError as e:
        print("HTTP Error code:{}".format(e.code))
    except urllib.error.URLError as e:
        print("URL Error Reason:{}".format(e.reason))
```

### Run Script

```
pi@DoPro:~/domoticz/scripts/python $ python3 user_variables_list.py
```

```
Name;Idx;Value
TH_AMBIENTLIGHT;1;100
TH_TIMECONTROL;2;2.5
```

# Usage Scripts

Examples on how to use the User Variables in scripts.

## **dzVents, Lua**

Read more [here](#) how to use with dzVents.

### *Read Value*

```
-- Update the alert message with level.  
function msgbox.alertmsg(domoticz, level, msg)  
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)  
end
```

```
local message = domoticz.variables('DEF_ALERTMSG').value
```

### *Set Value*

```
domoticz.variables('DEF_ALERTMSG').set('Hello World')
```

## Browser Interactive

Examples on how to use the User Variables in scripts.

Read more [here](#) how to use.

### *Read Value*

A value is obtained by using the idx of the user variable.

```
http://rpi-ip/json.htm?type=command&param=getuservariable&idx=8
```

### Response

```
{  
    "result" : [  
        {  
            "LastUpdate" : "2019-02-15 14:29:34",  
            "Name" : "TH_STOCK_STOCKA",  
            "Type" : "1",  
            "Value" : "29",  
            "idx" : "8"  
        }  
    ],  
    "status" : "OK",  
    "title" : " GetUserVariable"  
}
```

### *Set Value*

Set a new value for the threshold of a monitored stock. The value is type 1 (float).

```
http://rpi-ip/json.htm?type=command&param=updateuservariable&vname=TH_STOCK_STOCKA&vtype=1&vvalue=29
```

### Response

```
{  
    "status" : "OK",  
    "title" : " UpdateUserVariable"  
}
```

## Node-RED

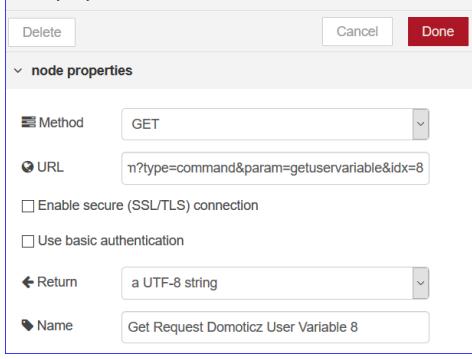
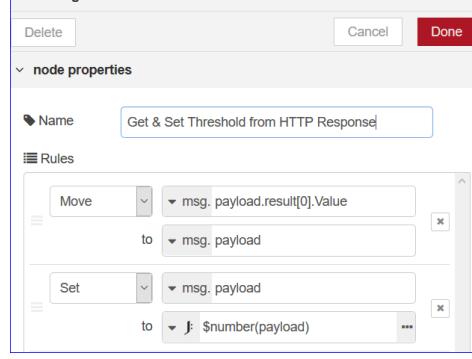
Examples on how to use the User Variables in Node-RED.

For Node-RED, there are several possibilities.

This example uses HTP Response Nodes.

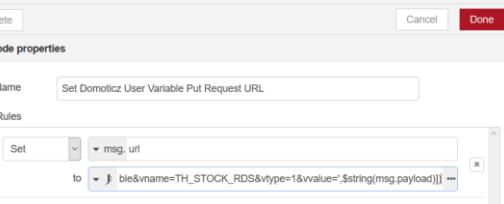
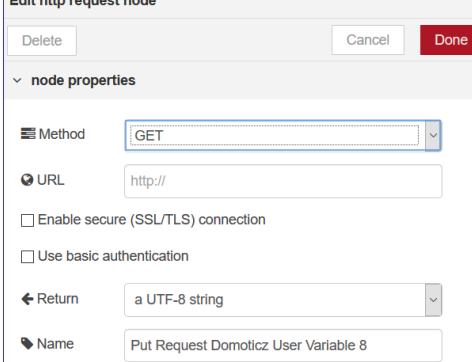
### Read Value

Same request as previous used in the browser request.

 <p>Use a JSON node to convert the output to JSON Object used by the Change Node.</p>	 <p>Two rules first move the property Value from the JSON response to the msg.payload, then use that string to get converted to a number using JSONata function.</p>
--	--

### Set Value

Set a new value for the threshold of a monitored stock. The value is type 1 (float).

 <p>Set the property msg.url using JSONata function join (JSON/API + Value)</p>	 <p>The request uses the previous defined msg.url to send to Domoticz</p>
--	---

## Node-RED Dashboard

The Node-RED Dashboard has input widgets.

This makes it possible to build a web page (with several tabs and groups) to configure User Variables.

Next an example setting the value of a User Variable.

### Example Ambient Light Threshold

Set the threshold of the Ambient Light Sensor. If below threshold, then a light is switched on. The threshold is used by a dzVents script to control the light (see [Ambient Light](#)).

#### Enhancement Idea

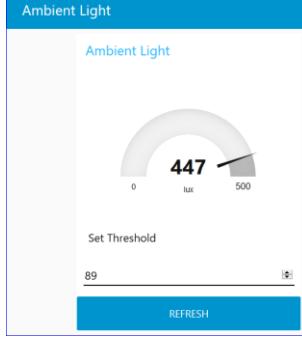
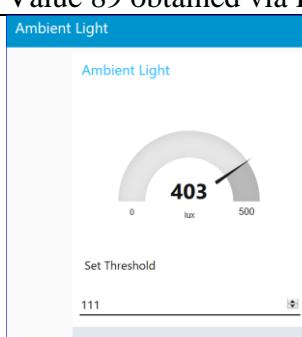
Define a User Variable (like TH\_AMBIENT\_LIGHT\_SWITCH) holding the idx of the light (better switch) to be turned ON when the value is below threshold. A ui\_dropdown list could be defined to select the light (switch) and update the User Variable.

### Node-RED Dashboard

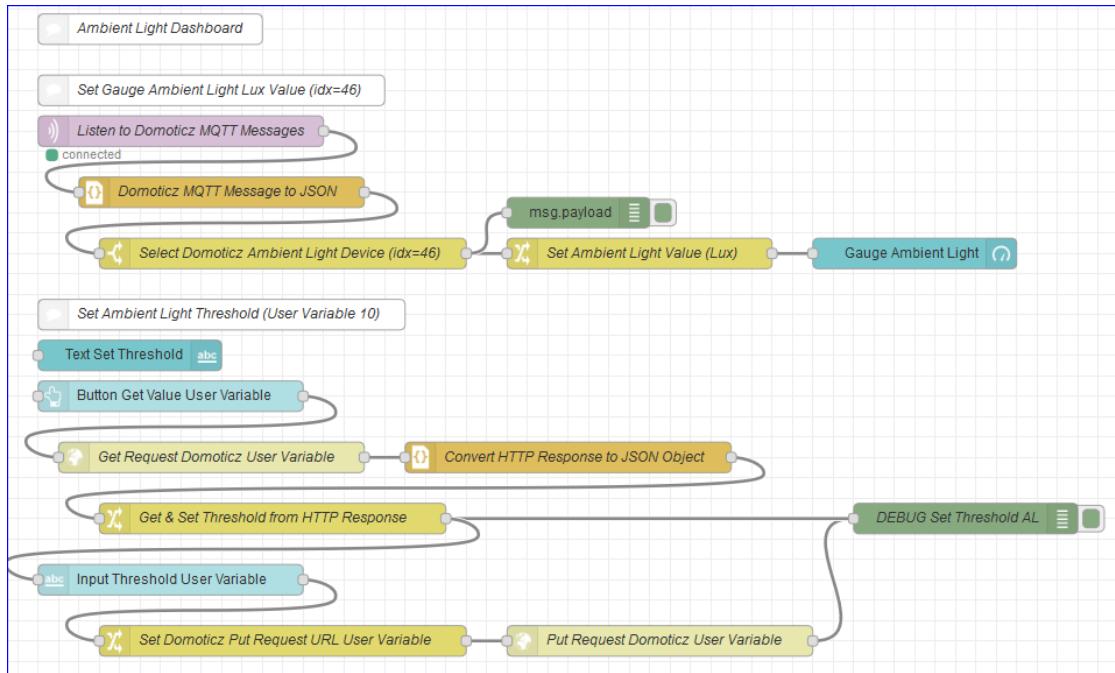
A ui\_gauge node displays the actual Ambient Light Lux value, which is obtained from MQTT payload of the Ambient Light device. Example from which property "svalue1" is used by the gauge:

```
{"Battery":255, "RSSI":5, "description":"The ambient light intensity in Lux using a BH1750 sensor.", "dtype":"Lux", "id":82045, "idx":46, "name":"ESPEasy Ambient Light", "nvalue":0, "stype":"Lux", "svalue1":3050.00, "unit":1}
```

The threshold is set through the ui\_input node, which is configured for number input. Entering the new number and pressing Enter, the value is updated in Domoticz.

 <b>Value 89 obtained via Refresh</b>	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Idx</th> <th>Variable name</th> <th>Variable type</th> <th>Current value</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>TH_AMBIENT_LIGHT</td> <td>Integer</td> <td>89</td> </tr> </tbody> </table> <b>Domoticz User Variable List (extract for IDX=10)</b>	Idx	Variable name	Variable type	Current value	10	TH_AMBIENT_LIGHT	Integer	89
Idx	Variable name	Variable type	Current value						
10	TH_AMBIENT_LIGHT	Integer	89						
 <b>Changed to 111 and pressed enter</b>	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Idx</th> <th>Variable name</th> <th>Variable type</th> <th>Current value</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>TH_AMBIENT_LIGHT</td> <td>Integer</td> <td>111</td> </tr> </tbody> </table> <b>Refresh the page in the browser, shows the updated value for the User Variable</b>	Idx	Variable name	Variable type	Current value	10	TH_AMBIENT_LIGHT	Integer	111
Idx	Variable name	Variable type	Current value						
10	TH_AMBIENT_LIGHT	Integer	111						

### Node-RED Flow



## Overview API Requests

### Defined User Variable

Variable Name: TH\_AMBIENT\_LIGHT, Type: Integer, Idx: 10

### HTTP Request Get User Variable

<http://localhost:8080/json.htm?type=command&param=getuservariable&idx=10>

### HTTP Response (JSON Format)

```
{
  "result": [
    {
      "LastUpdate": "2019-02-27 14:37:28",
      "Name": "TH_AMBIENT_LIGHT",
      "Type": "0",
      "Value": "89",
      "idx": "10"
    }
  ],
  "status": "OK",
  "title": " GetUserVariable"
}
```

### HTTP Request Update User Variable

[http://localhost:8080/json.htm?type=command&param=updateuservariable&vname=TH\\_AMBIENT\\_LIGHT&vtype=0&value=NNNN](http://localhost:8080/json.htm?type=command&param=updateuservariable&vname=TH_AMBIENT_LIGHT&vtype=0&value=NNNN)

Note

vtype is 0 which is an Integer.

### HTTP Response (JSON Format)

```
{
  "status": "OK",
  "title": "UpdateUserVariable"
}
```

# Custom Page

## Purpose

To (quickly) change selective user variables via a dedicated custom web page instead using the Domoticz default GUI > Setup > More Options > User variables.

### *Note*

Below solution has been used to test.

The function [Custom Settings](#) is based on this solution and being enhanced further.

## Solution

Custom webpage "User\_Variables.html" with edit fields & set button for selective variables.  
The HTML file must be copied to the folder:

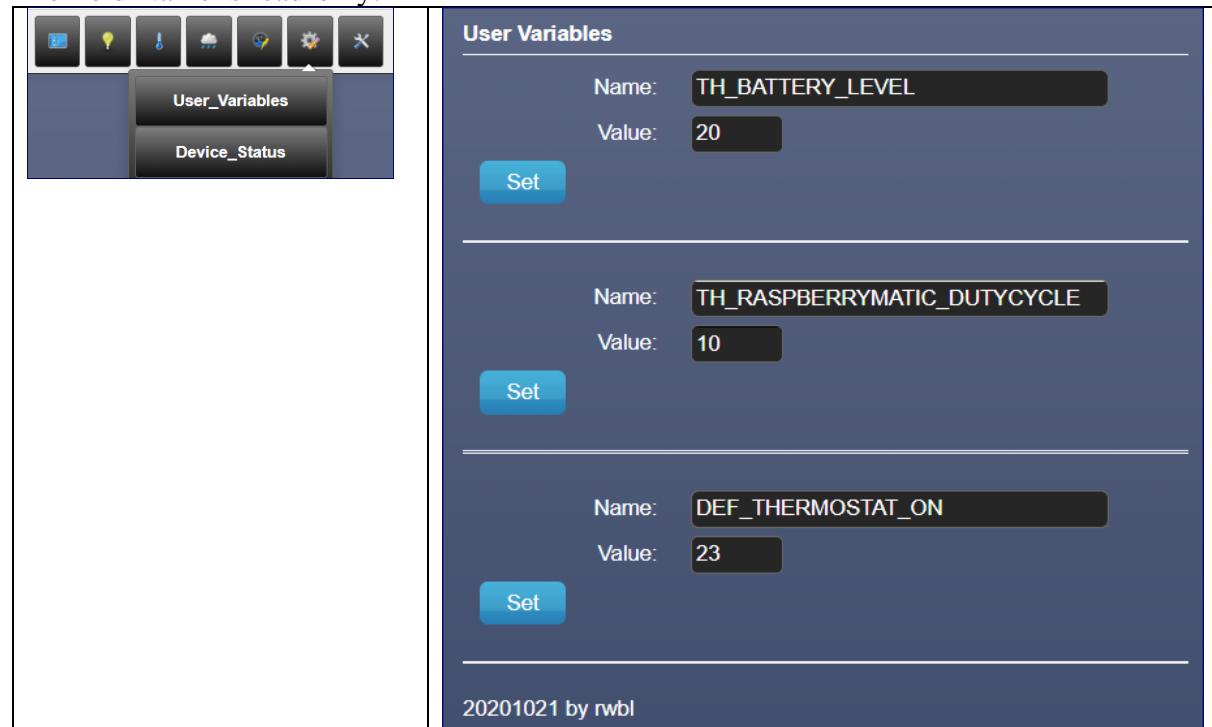
```
~domoticz/www/templates
```

Direct page access from a browser:

```
http://domoticz-ip:port/ #/Custom/User_Variables
```

### Screenshot

GUI Custom Dropdown Menu entries and the Custom Page User\_Variables with two entries.  
The field Name is read-only.



### *Note*

The page layout is inspired by the Domoticz page:

```
~domoticz/www/views/uservvariables.html
```



## HTML File User\_Variables

Source: User\_Variables.html

The source is well documented and outlines how to add a new user variable.

```
<!--
domoticz-homeautomation-workbook function User_Variables
Domoticz custom page:
User_Variables.html (/home/pi/domoticz/www/templates)
Purpose:
To change the value of selected user variables.
Any other settings, to be set via Domoticz GUI Setup > Settings (the standard way).
Notes
20201022 rwbl
-->
<style>
</style>

<!--
HTML
-->
<div class="container">
<div class="page-header-small">
<h4 data-i18n="User Variables"></h4>
</div>
<!--
Each user variable consists out of a table with 4 fields variablename (readonly), variablevalue
(mandatory), variableidx (readonly), variablename (readonly).
The id of the table must set and unique, i.e. table id="raspmaticdutycycle".
The field data is initially set when loading/refeshing the page.
Any new user variables copy & paste the first entry <table id=...></table> and change the id.
Changes required:
* Set the table id, i.e. batterylevelcheck
* Set the selector in function updateUserVariable(), i.e. updateUserVariable('#batterylevelcheck');
* Set the input type, i.e. type="number"
* Set the min/max values, i.e. min="0", max="100"
* Amend the document load function getUserVariable('#uservariable', uservariableidx), i.e.
getUserVariable('#batterylevelcheck', 24);
-->

<!--
Battery Level Check (idx=24)
-->
<table id="batterylevelcheck" class="display" border="0" cellpadding="0" cellspacing="20">
<tr>
<td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span></label></td>
<td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
</tr>
<tr>
<td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span></label></td>
<td><input id="variablevalue" type="number" min="0" max="100" style="width: 50px; padding: .2em;" class="text ui-widget-content ui-corner-all" required /></td>
</tr>
<tr>
<td colspan="2">
<a class="btnstyle3" onclick="updateUserVariable('#batterylevelcheck');" data-i18n="Set"></a>
</td>
</tr>
<!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden -->
<tr hidden>
<td align="right" style="width:110px"><label for="variableidx"><span data-
i18n="Idx"></span></label></td>
<td><input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;" class="text ui-widget-content ui-corner-all" readonly /></td>
</tr>
<tr hidden>
<td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Type"></span></label></td>
<td><input id="variablename" type="number" style="width: 50px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
```

```

        <!-- 0=Integer,1:Float,2=String3=Date4=Time5=DateTime -->
    </tr>
</table>
<hr>

<!--
 RaspberryMatic DutyCycle (idx=23)
-->
<table id="raspmaticdutycycle" class="display" border="0" cellpadding="0" cellspacing="20">
    <tr>
        <td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span></label></td>
        <td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
    </tr>
    <tr>
        <td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span></label></td>
        <td><input id="variablevalue" type="number" min="1" max="20" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" required /></td>
    </tr>
    <tr>
        <td colspan="2">
            <a class="btnstyle3" onclick="updateUserVariable('#raspmaticdutycycle');" data-i18n="Set"></a>
        </td>
    </tr>
    <!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden -->
    <tr hidden>
        <td align="right" style="width:110px"><label for="variableidx"><span data-
i18n="Idx"></span></label></td>
        <td><input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" readonly /></td>
    </tr>
    <tr hidden>
        <td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Type"></span></label></td>
        <td><input id="variablename" type="number" style="width: 50px; padding: .2em;" 
class="text ui-
widget-content ui-corner-all" readonly /></td>
        <!-- 0=Integer,1:Float,2=String3=Date4=Time5=DateTime -->
    </tr>
</table>
<hr>

<!--
 Thermostat Temperature On (idx=21)
-->
<table id="thermostaton" class="display" border="0" cellpadding="0" cellspacing="20">
    <tr>
        <td align="right" style="width:110px"><label for="variablename"><span data-
i18n="Name"></span></label></td>
        <td><input type="text" id="variablename" style="width: 250px; padding: .2em;" class="text ui-
widget-content ui-corner-all" readonly /></td>
    </tr>
    <tr>
        <td align="right" style="width:110px"><label for="variablevalue"><span data-
i18n="Value"></span></label></td>
        <td><input id="variablevalue" type="number" min="6" max="23" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" required /></td>
    </tr>
    <tr>
        <td colspan="2">
            <a class="btnstyle3" onclick="updateUserVariable('#thermostaton');" data-i18n="Set"></a>
        </td>
    </tr>
    <!-- The next fields are hidden, but the values are used. if want to show, then remove parameter
hidden -->
    <tr hidden>
        <td align="right" style="width:110px"><label for="variableidx"><span data-
i18n="Idx"></span></label></td>
        <td><input id="variableidx" type="number" value="0" style="width: 50px; padding: .2em;" 
class="text ui-widget-content ui-corner-all" readonly /></td>
    </tr>
    <tr hidden>

```

```

        <td align="right" style="width:110px"><label for="variabletype"><span data-
i18n="Type"></span></label></td>
        <td><input id="variabletype" type="number" style="width: 50px; padding: .2em; " class="text ui-
widget-content ui-corner-all" readonly /></td>
        <!-- 0=Integer,1:Float,2=String3=DateTime 4=DateTime -->
    </tr>
</table>
<hr>

<!--
    Add More
-->

<!-- Version info -->
<div id="settingsversion" class="bannercontent"></div>
</div>

<!--
    JavaScript
-->
<script type="text/javascript" charset="utf-8">

    // Update the value of a user variable
    // Get UV Data: json.htm?type=command&param=getuservariable&idx=IDX
    // Update UV Value:
    json.htm?type=command&param=updateuservariable&vname=USERVARIABLENAME&vtype=USERVARIABLETYPE&vvalue=USE
    RVARIABLEVALUE
    function updateUserVariable(variableSelector) {
        var uvIdx = $(variableSelector + ' ' + '#variableidx').val();
        var uvName = $(variableSelector + ' ' + '#variablename').val();
        var uvType = $(variableSelector + ' ' + '#variabletype').val();
        var uvValue = $(variableSelector + ' ' + '#variablevalue').val();
        var requestUrl = "json.htm?type=command&param=updateuservariable&idx=" + uvIdx + "&vname=" + uvName
        + "&vtype=" + uvType + "&vvalue=" + uvValue;
        console.log("updateUserVariable - URL: " + requestUrl);
        $.ajax({
            url: requestUrl,
            async: false,
            dataType: 'json',
            success: function (data) {
                if (typeof data != 'undefined') {
                    if (data.status == "OK") {
                        bootbox.alert($.t('User variable saved'));
                    }
                    else {
                        ShowNotify($.t(data.message), 2500, true);
                    }
                }
            },
            error: function () {
                ShowNotify($.t('Problem saving user variable!'), 2500, true);
            }
        });
    };

    // Get the user variable properties
    // Get UV Data: json.htm?type=command&param=getuservariable&idx=IDX
    // Parameter:
    // variableSelector (String): #raspmaticdutycycle
    // uvIdx (Integer): 23
    function getUserVariable(variableSelector, uvIdx) {
        // var uvIdx = $(variableSelector + ' ' + '#variableidx').val();
        // Define the http api url to get uservariable properties name and type used to set the new value
        var requestUrl = "json.htm?type=command&param=getuservariable&idx=" + uvIdx;
        console.log("setUserVariable - Get data: " + requestUrl);
        $.ajax({
            url: requestUrl,
            async: false,
            dataType: 'json',
            success: function (data) {
                if(typeof data.result == "undefined"){
                    console.log("setUserVariable - no data found (status: " + data.status + ", title: " +
data.title + ")");
                    return false;
                }
            }
        });
    };

```

```
// console.log("setUserVariable - Succesfully retrieved (status: " + data.status + ", title: " +  
data.title + ")");  
    // Get the uservariable name and type  
    var uvName = data.result[0].Name;  
    var uvType = data.result[0].Type;  
    var uvValue = data.result[0].Value;  
    console.log("getUserVariable Idx,Name,Type,Value: " + uvIdx + "," + uvName + "," + uvType + "," +  
+ uvValue);  
    // Update the user value properties  
    $(variableSelector + ' ' + '#variablename').val(uvName);  
    $(variableSelector + ' ' + '#variablevalue').val(uvValue);  
    $(variableSelector + ' ' + '#variableidx').val(uvIdx);  
    $(variableSelector + ' ' + '#variabletype').val(uvType);  
},  
error: function () {  
    console.log("[ERROR] Status all devices. Cannot communicate to the Domoticz server!");  
    return false;  
}  
});  
};  
  
// Sleep for ms  
function sleep(milliseconds) {  
    const date = Date.now();  
    let currentDate = null;  
    do {  
        currentDate = Date.now();  
    } while (currentDate - date < milliseconds);  
}  
  
function setVersion(pageVersion) {  
    $('#settingsversion').html(pageVersion);  
};  
  
// Load the document  
$(document).ready(function(){  
    setVersion("20201021 by rwbl");  
    // Get the user variable data with selector, idx  
    getUserVariable('#raspmaticdutycycle', 23);  
    sleep(50);  
    getUserVariable('#thermostaton', 21);  
    sleep(50);  
    getUserVariable('#batterylevelcheck', 24);  
});  
  
</script>
```

# Custom Icons

## Purpose

Custom icons are used for several devices.

*Note*

Not all devices support icon customizing, i.e. General, Text (status Domoticz 2020.2). Check Domoticz future versions if possible to assign a custom icon to for example a Text device.

## Create Icons

Each Custom Icon requires three icons, defined as PNG files:  
(size defined as width pixel x height pixel):

- Icon.png (size 16x16 pixel, icon small)  
Used for example by the Dashboard
- Icon\_On.png (size 48x48 pixel, icon On state)  
Used for example by the device widget
- Icon\_Off.png (size 48x48 pixel, icon Off state)  
Used for example by the device widget

*Note*

The icons Icon\_On.png & Icon\_Off.png are mandatory (even if the device is not a switch type)!

## Icon List File

The icons are defined in a CSV file *icons.txt* which is used to import the icons.

Each line describes an icon with 3 comma separated entries:

**Icon file name; Icon interface name; Icon description**

Icon;Icon;Custom Icon
-----------------------

*Note*

The first two entries (icon file name and icon interface name) MUST match the icon file name!

## Icon Archive

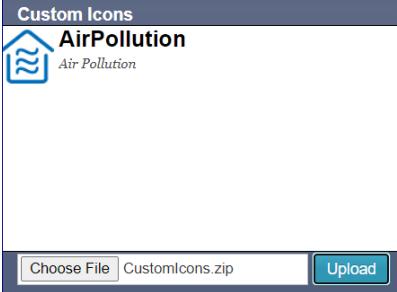
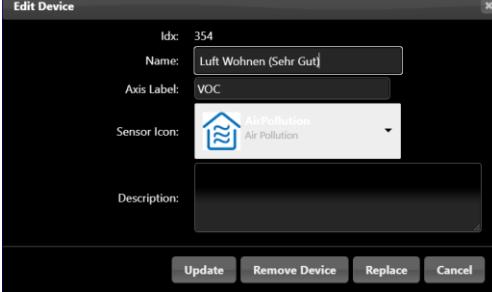
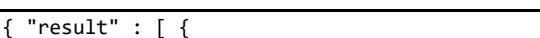
The icons are imported using an archive.

The archive *CustomIcons.zip* contains following mandatory files:

- Icons.txt
- Icon.png
- Icon\_On.png
- Icon\_Off.png

# Icon Import

Example adding a single icon AirPollution.

Step	Content
Create folder to hold the icons, icons description and archive.	<code>~AirPollution</code>
Create 3 icons.	<code>AirPollution.png (16x16 pixels)</code> <code>AirPollution48_On.png (48x48 pixels)</code> <code>AirPollution48_Off.png (48x48 pixels)</code>
Create file <code>icons.txt</code> with single line content: <i>icon file name; icon interface name; icon description.</i>	<code>AirPollution;AirPollution;Air Pollution</code>
Create archive <code>CustomIcons.zip</code> with the files.	<code>icons.txt</code> <code>AirPollution.png</code> <code>AirPollution48_On.png</code> <code>AirPollution48_Off.png</code>
Upload the icons from the Domoticz GUI: GUI > Setup > More Options > Custom Icons.	Choose file <code>CustomIcons.zip</code> from folder <code>~AirPollution</code> and click Upload 
Note After uploading the icons, the folder <code>~domoticz/www/images</code> contain the imported icon files.	
Select a device widget from the Domoticz GUI, Edit, Select the new Sensor Icon <i>Air Pollution</i> and Update.	
Check if the device widget shows the new icon (48x48 pixel).	
Check if the Dashboard (example Mobile) shows the new icon (16x16 pixel).	
As an FYI: The icon information as stored in the Domoticz database, retrieved via HTTP API request: <code>http://domoticz-ip:port/json.htm?type=custom_light_icons</code>	<pre>{ "result" : [ { "description" : "Air Pollution", "idx" : 101, "imageSrc" : "AirPollution", "text" : "AirPollution" }, ... ] }</pre>
Note If the new icon is not listed, restart Domoticz ( <code>sudo service domoticz restart</code> ) and try again.	

# Icon Import (2<sup>nd</sup> example)

Import two Custom Icons *Refresh* and *Bookmark* used by the Function Volumio.

**Create the six icon files**  
(16x16, 48x48, 48x48 pixel)

```
Refresh.png, Refresh48_On.png, Refresh48_Off,  
Bookmark.png, Bookmark48_On.png, Bookmark48_Off.png
```

**Create file icons.txt**

```
Refresh;Refresh;Refresh Status  
Bookmark;Bookmark;Bookmark Song
```

**Create the archive *CustomIcons.zip***  
with following files

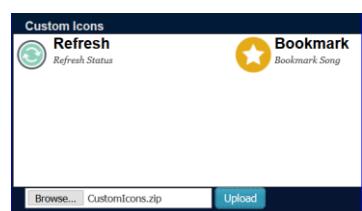
```
icons.txt  
Refresh.png  
Refresh48_On.png  
Refresh48_Off  
Bookmark.png  
Bookmark48_On.png  
Bookmark48_Off.png
```

**Import the icons in Domoticz**

Open Domoticz GUI > Setup > More Options > Custom Icons.

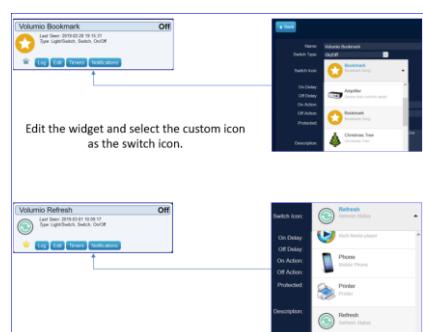
Browse for the file *CustomIcons.zip* and upload.

After upload, the custom icons are shown and listed when editing the icon on a device (via Widget > Edit).



**Assign an Icon to a Device Widget**

The icons are added to the device widget by selecting from the icon list (edit):



# Additional Information

## Get List of Icons

### HTTP API Request

```
http://domoticz-ip:port/json.htm?type=custom_light_icons
```

#### HTTP API Response

The HTTP API request returns a JSON string.

Below is an extract of the JSON result, showing the two new icons added with their IDX. The icons are selected from the file `/home/pi/domoticz/www/switch_icons.txt` and Domoticz database query on table `CustomImages`.

```
{
  "result" : [
    ...
    {
      "description" : "Set a Bookmark",
      "idx" : 102,
      "imageSrc" : "Bookmark",
      "text" : "Bookmark"
    },
    {
      "description" : "Refresh Status",
      "idx" : 101,
      "imageSrc" : "Refresh",
      "text" : "Refresh"
    },
    ...
  ],
  "status" : "OK"
}
```

#### Notes

- The custom image IDX is 100 + ID (from the field ID of database table CustomImages). The *Refresh icon* is the first entry in the file icons.txt, has field ID=1 and IDX=101 (100+1).
- The IDX of the custom icon can be used for example when creating Plugin devices (Parameter Image=idx, i.e. Image=101). Ensure to add the custom image prior creating new devices when using the optional parameter Image=idx.

## Location Custom Images

### Domoticz Database

This is an example querying, the Domoticz database on the development server, for custom images.

Only one custom image *Airquality* has been added:

```
Using username "pi".
Linux dodev 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l

cd domoticz
sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
```

```
sqlite> SELECT ID,Base,Name,Description FROM CustomImages;
1|Airquality|Airquality|Air Quality
sqlite> .exit
```

*Extract of the Query result for the icon list via HTTP API*

```
http://rpi-ip:8080/json.htm?type=custom_light_icons

{
    "result" : [
        ...
        {
            "description" : "Air Quality",
            "idx" : 101,
            "imageSrc" : "Airquality",
            "text" : "Airquality"
        },
        ...
    ],
    "status" : "OK"
}
```

## Folder

The images are copied from the *CustomIcons.zip* to the folder

```
/home/pi/domoticz/www/images/
```

```
ls /home/pi/domoticz/www/images/Airquality*.*
/home/pi/domoticz/www/images/Airquality48_Off.png
/home/pi/domoticz/www/images/Airquality48_On.png
/home/pi/domoticz/www/images/Airquality.png
```

## SQL statement to set a Custom Icon

Setting a custom icon via SQL example.

Example:

Assign the custom icon “Air Quality” which has IDX=101, to the device with IDX=41.

```
UPDATE DeviceStatus SET CustomImage = 101 WHERE ID=41
```

# Explore

This chapter is used to explore areas, like Domoticz functions, MQTT, SQL, Node-RED and more.

## API Interaction

### Purpose

To explore how to interact using Domoticz API commands and handle response accordingly. Ensure also to lookup the reference [Domoticz API/JSON Url's](#).

This chapter does not go into the depth – just a small example on setting the brightness of a Hue Light via a remote-control device, which could be a micro-controller like an ESP8266 or ESP32 or a Raspberry Pi Zero W.

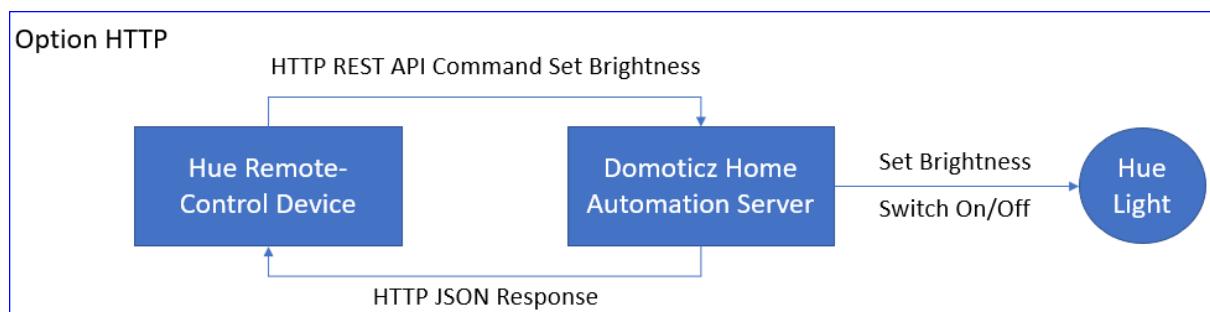
The remote-control device requires communication with the Domoticz system which is integrated in the SMART Home network.

### Option HTTP

Domoticz allows you to interact with all devices using HTTP API Commands generating JSON returns.

This means, it is possible to send commands to the Domoticz system to act upon and receive a response back.

Sending a command and view the server response can be easily done via a browser.



#### *Note*

Under the hood, Domoticz uses HTTP API Commands, which is the preferred way to communicate with the devices.

### Example HTTP API Request via Browser

#### *Goal*

Set the brightness to 50% for Hue Device with Domoticz IDX=118.

#### *HTTP API Request URL*

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=50
```

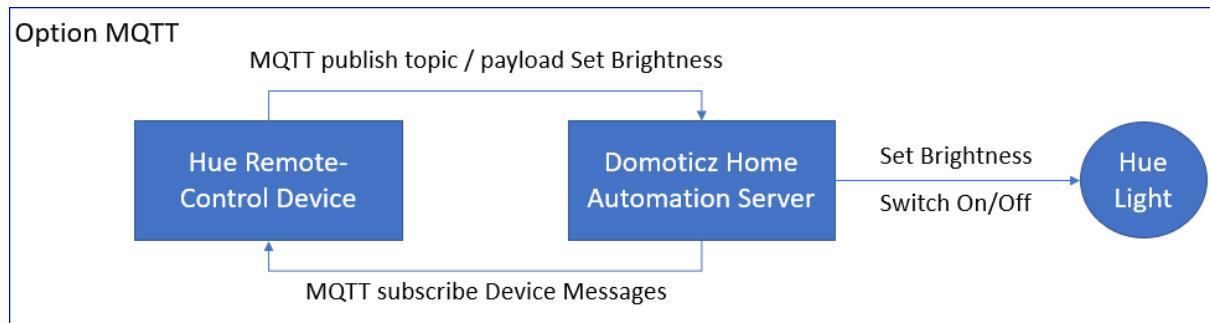
#### *HTTP JSON Response*

```
{  
    "status": "OK", "title": "SwitchLight"  
}
```

## Option MQTT

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. MQTT provides a publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications.

Read more in chapter [MQTT](#).



To be able to use MQTT in Domoticz, a MQTT Broker (like mosquito) and a MQTT Client (Hardware MQTT Client Gateway with LAN interface) must be installed ([read](#)).

Communication from and to Domoticz works via JSON.

Default MQTT topics of the Domoticz for incoming (domoticz/in) and outgoing (domoticz/out) messages.

To set the brightness of the Hue Light, a message must be published from the Hue Remote-Control to Domoticz.

### Example MQTT Messages publish & subscribe

#### *Goal*

Set the brightness to 50% for the Hue Device with Domoticz IDX=118.

#### *Publish*

The MQTT message to publish requires a topic and payload.

Topic:

domoticz/in

Payload:

```
{"command": "switchlight", "idx": 118, "switchcmd": "Set Level", "level": 100}
```

#### *Note*

Lookup for the payload syntax in the API reference.

The API reference uses HTTP syntax which is used for MQTT as well.

Again, the example:

### HTTP

```
type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=100
```

### MQTT

```
{"command": "switchlight", "idx": 118, "switchcmd": "Set Level", "level": 100}
```

### *Subscribe*

Topic:

domoticz/out

After publishing the message, the light brightness gets updated and Domoticz responses.

When subscribing to topic domoticz/in, the message payload received for the device with IDX=118:

```
{"Battery" : 255,"Level" : 100,"RSSI" : 12,"description" : "", "dtype" : "Light/Switch","id" : "00000005","idx" : 118,"name" : "Hue MakeLab","nvalue" : 2, "stype" : "Switch", "svalue1" : "100", "switchType" : "Dimmer","unit" : 1}
```

### Python Script Example

```
#!/usr/bin/env python

import time
import paho.mqtt.client as mqtt

# Define globals
version="MQTT Test v20190408"
# MQTT broker running on the Raspberry Pi
broker= "localhost"
# MQTT Domoticz topic for sending messages to Domoticz - Ensure the topic is in lowercase
topic = "domoticz/in"
# Device Idx to get its value = Hue Light named MakeLab
idx = 118
payload='{"command": "switchlight", "idx": %s, "switchcmd":"Set Level", "level": 0}' % (idx)

# MQTT Callback Functions
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

def on_log(client, userdata, level, buf):
    print("log: ",buf)

print(version)
print("Creating new instance")
client = mqtt.Client("P1")
# Attach callback functions
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log
print("Connecting to the broker", broker)
client.connect(broker, 1883, 60)
# Wait
time.sleep(4)
# Publish
print("Publishing message to topic",topic, "with payload",payload)
# Publish.
# Example Domoticz Log:
# 2019-04-09 09:25:19.481 MQTT: Topic: domoticz/in, Message: {"command": "switchlight", "idx": 118,
# "switchcmd":"Set Level", "level": 50}
# 2019-04-09 09:25:19.511 (Philips Hue Bridge 1) Light/Switch (Hue MakeLab)
client.publish(topic,payload)
# Wait
time.sleep(4)
# Stop the loop
client.loop_stop()
```

## Recommendation

Use **HTTP API commands** to stick to the Domoticz preferred way of communication and parse the JSON response accordingly.

MQTT is more complex to use in terms of publishing & subscribing to messages.

## Example Syntax HTTP vs MQTT

The purpose is to show the syntax for setting the

- brightness of a Hue Light (IDX=118) to 50% and
- text of a Virtual Text Sensor

via HTTP or MQTT.

It helps understanding how to apply commands for HTTP vs. MQTT.

Lookup for the payload syntax in the [Domoticz JSON/API reference](#).

The API reference uses HTTP syntax which is also used for MQTT commands.

### Example Set Hue Light Brightness

#### HTTP API Request

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=50
```

#### HTTP JSON Response

```
{
  "status": "OK", "title": "SwitchLight"
}
```

#### MQTT Command

```
{
  "command": "switchlight", "idx": 118, "switchcmd": "Set Level", "level": 50
}
```

#### MQTT JSON Response

```
{
  "Battery" : 255, "Level" : 50, "RSSI" : 12, "description" : "", "dtype" : "Light/Switch", "id" : "00000005", "idx" : 118, "name" : "Hue MakeLab", "nvalue" : 2, "stype" : "Switch", "svalue1" : "50", "switchType" : "Dimmer", "unit" : 1
}
```

## Summary

Comparison HTTP parameters vs MQTT Key: Value pairs

HTTP	MQTT Key: Value
param=switchlight	"command": "switchlight"
idx=118	"idx": 118
switchcmd=Set%20Level	"switchcmd": "Set Level"
level=50	"level": 50

## Example Set Virtual Text Sensor

### HTTP API Request

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=52&nvalue=0&svalue>Hello%20World
```

### HTTP JSON Response

```
{
  "status": "OK", "title": "Update Device"
}
```

### MQTT

```
payload='{"command": "udevice", "idx": 52, "nvalue": 0, "svalue": "Hello World"}
```

### MQTT JSON Response

```
{
  "Battery" : 255, "RSSI" : 12, "description" : "Display control message from several events.", "dtype" : "General", "id" : "00082051", "idx" : 52, "name" : "Control Message", "nvalue" : 0, "stype" : "Text", "svalue1" : "Hello World", "unit" : 1
}
```

### Domoticz Log

```
2019-04-09 09:52:27.424 MQTT: Topic: domoticz/in, Message: {"command": "udevice", "idx": 52, "nvalue": 0, "svalue": "Hello World"}
```

### Summary

Comparison HTTP parameters vs MQTT Key: Value pairs

HTTP	MQTT Key: Value
param=udevice	"command": "udevice"
idx=52	"idx": 52
nvalue=0	"nvalue": 0
svalue=Hello%20World	"svalue": "Hello World"

# Custom Pages

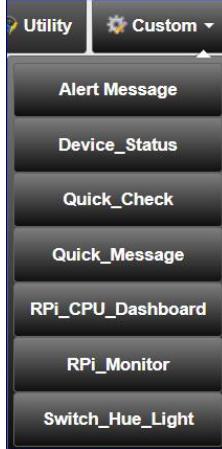
## Purpose

To explore how to create simple Domoticz Custom Pages.

### Notes

- A Custom Page is integrated in the Domoticz User Interface, which enables to use Domoticz modules & styling.
- The examples show how to get or set Domoticz device data.
- The communication between a custom page and Domoticz via HTTP API requests.
- The latest source can be found in the [GitHub](#) repository folder src.

Beside the examples, several functions make also use of Custom Pages, i.e. [Device Status](#), [Custom Settings](#) or [Notes](#).

Domoticz Development System	Domoticz Production System
	

# Prepare

## Setup

A custom page is accessible from the Domoticz GUI > Tab Custom.



This requires activation of Domoticz GUI > Setup > Settings > Active Menus > Custom.

The custom pages, files with the extension .html, are stored in folder

```
~domoticz/domoticz/www/templates
```

on the Domoticz System running on the Raspberry Pi (or other hardware).

The path to the Domoticz folder is on the Raspberry Pi:

```
/home/pi/domoticz/...
```

*Note*

The custom page menu is dynamically created by the JavaScript module

```
~domoticz/domoticz/www/app/app.js
```

and integrated in the Domoticz HTML structure.

## Development Hints

### Constants & Variables

- Ensure to use unique script wide constants & variables for a page, to avoid the redeclare error message.
- It is better to use local function variables instead of script wide vars.
- Use functions with parameters if using script wide constants.
- Some constants in the examples which strives to define only the Domoticz system URL and the idx of the devices used.

### Example Constants

```
// Alert Message
const AM_URL_DOMOTICZ = "http://domoticz-ip:8080";
const AM_IDX_ALERTMESSAGE = 110;
```

```
// Notes
const NOTES_ABOUT = "Domoticz Explore Custom Pages.";
const NOTES_URL_DOMOTICZ = "http:// domoticz-ip:8080";
const NOTES_IDX_NOTES = 111;
```

## CSS Font Awesome Icons

These are not included in the default Domoticz setup.

*Note*

This is status Domoticz v2020.2. It might change, so checkout newer Domoticz versions.

### Options to include Font Awesome Icons

There are two options to access the Font Awesome icons, either remote or local.

#### Remote Access

The Font Awesome icons are accessed remote from CDN by including the link:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

#### Local Access

The Font Awesome icons are locally accessed on the Domoticz system.

#### Installation

Download the archive from [here](#).

Create folder:

```
~domoticz/www/css/fontawesome
```

Extract the archive folder structure to the newly created folder.

In the HTML file, include the link referencing the css file “all.min.css”:

```
<link rel="stylesheet" href="css/fontawesome/css/all.min.css">
```

#### Usage

##### Icon arrow-circle-up



```
<button id="btnup" class="btn btn-block btn-success fa fa-arrow-circle-up fa-3x" title="UP">
</button>
```

##### Icon info-circle



```
<label>
  <i class="fa fa-info-circle" style="font-size:60px;color:blue;"></i>
</label>
```

##### Two icons stacked



```
<span class="fa-stack fa-lg">
  <i class="fa fa-car fa-stack-1x"></i>
  <i class="fa fa-ban fa-stack-2x text-danger" style="color:red;"></i>
</span>
```

Another example is *CSS Styling Button Block with Font Awesome Icons*.

Useful references: [Cheatsheet](#), Icon [Sizing](#) (fa-lg (+33%), fa-2x, fa-3x, fa-4x, fa-5x).

## CSS Styling

The Domoticz CSS Style Classes can be used in custom pages to keep the same User Interface style as defined in the Domoticz Settings (GUI > Setup > System > User Interface).

Recommend to lookup folder “~domoticz/domoticz/www/css” to check out what is defined, esp. the file bootstrap.css.

Find below some examples using the Domoticz CSS Style Classes.

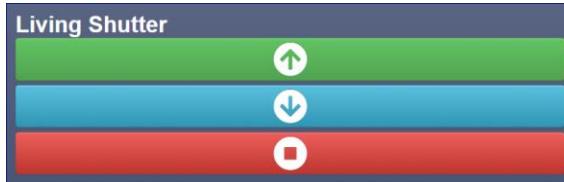
### Buttons

```
<button id="buttonsave" class="btn btn-success">Save</button>
<button id="buttoncopy" class="btn btn-warning">Copy</button>
<button id="buttoninfo" class="btn btn-info">Info</button>
```

### Button Small

```
<button id="buttonclear" class="btn btn-small btn-danger">Clear</button>
```

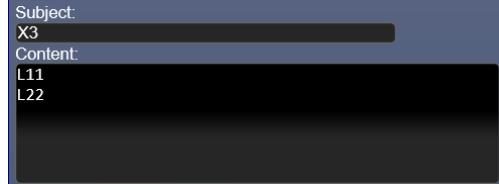
### Button Block with Font Awesome Icons



```
<!-- fontawesome accessed locally on the Domoticz server. Note: use all.min.css -->
<link rel="stylesheet" href="css/fontawesome/css/all.min.css">
<!--NOT USED fontawesome accessed remote from cdn -->
<!-- <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"> -->

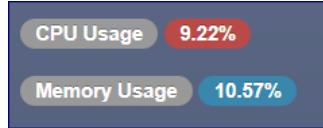
<div id="livingshutter" class="container-fluid">
  <div class="row-fluid span12">
    <div class="span12">
      <h1>Living Shutter</h1>
      <button id="btnclick" class="btn btn-block btn-success fa fa-arrow-circle-up fa-3x" title="UP"></button>
      <button id="btnclick" class="btn btn-block btn-info fa fa-arrow-circle-down fa-3x" title="DOWN"></button>
      <button id="btnclick" class="btn btn-block btn-danger fa fa-stop-circle fa-3x" title="STOP"></button>
    </div>
  </div>
</div>
```

## Form with an Input and Textarea Field



```
<form action="">
  <label for="notesubject" id="notesubjectlabel">Subject:</label><br>
  <input type="text" id="notesubject" placeholder="Subject" title="Set the note subject" style="width: 300px !important; min-width: 100px; max-width: 500px;" class="text ui-widget-content ui-corner-all"><br>
  <label for="notecontent" id="notecontentlabel">Content:</label><br>
  <textarea id="notecontent" rows="6" cols="50" placeholder="Add content ..." class="text ui-widget-content ui-corner-all"></textarea>
</form>
```

## Labels as Badges



```
<form action="">
  <label for="cpuusage" id="cpuusagelabel" class="badge">CPU Usage</label>
  <label id="cpuusage" class="badge badge-important"></label><br>
  <br>
  <label for="memoryusage" id="memoryusagelabel" class="badge">Memory Usage</label>
  <label id="memoryusage" class="badge badge-info"></label><br>
</form>
```

## Standalone Page

Develop the page as a standalone page first using the tags

```
<html><body> ... </body></html>
```

and run direct from browser, i.e.

```
file:///<path>/Switch_Hue_Light_Test.html.
```

## HTML Tags

### Important

For the final solution, do not use the tags html, head, title, body in the HTML file - except for the standalone tests.

## HTML API Requests

Test the HTML API requests first via browser url, i.e.

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=On
```

## Debug

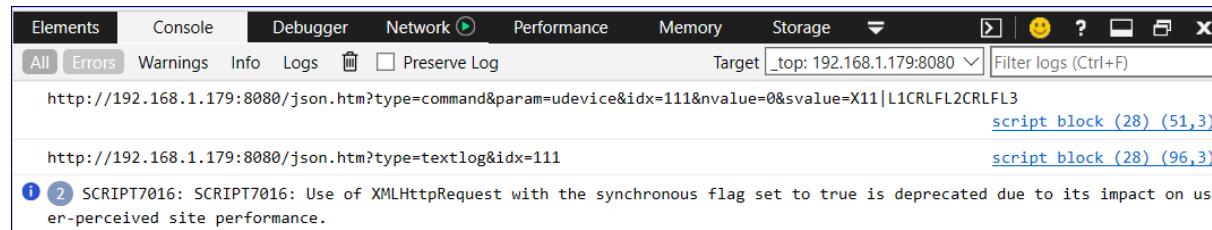
Use the browser debug (F12) to check the console output in tab Console.

This is important to ensure proper behaviour of the page.

In JavaScript use console.log() to write info.

```
console.log("Console Message");
```

Example:



## Explore Domoticz Source

Explore the Domoticz source code folder

```
~domoticz/www
```

(with subfolders) to learn how Domoticz handles web pages, esp. bootstrap.css.

As the custom page is embedded in the Domoticz GUI, it is possible to use defined functions also used by Domoticz, i.e.

- Bootstrap framework – Bootbox dialogs, i.e. alert, confirm, custom dialogs
- Bootstrap Grid
- ... explore more on what's possible ...

## Example Bootbox Alert in a JavaScript Function

## HTML

```
<p id="status">Status set by function setStatus</p>
```

## JavaScript

```
// update the selector with id status
function setStatus(msg,showdialog) {
    // Get the current time
    var dateToday = new Date();
    var strTime = dateToday.format("HH:MM:ss");
    msg = strTime + " : " + msg;
    $("#status").html(msg);
    if (showdialog){
        bootbox.alert(msg);
    }
    // console.log(msg);
}
```

## Custom Page Updates

After making page updates, refresh the GUI (F5) or clear the browser cache or restart Domoticz in case non expected behavior.

## Custom Page Filename

### Custom Menu Entry First Character

Domoticz sets the first character of the filename to uppercase, therefor used friendly name for the HTML file (i.e. Switch\_Hue\_Light).

### Custom Menu Entry Spaces

It is also possible to define file names with spaces instead of underscores used previous, i.e. Switch Hue Light which are then showed as the filename defined.

Do not use escaped characters, like %20 for a space as these are not converted when the dropdown menu is built.

## Custom Page Access

Access a custom page via Domoticz GUI > Custom > Filename

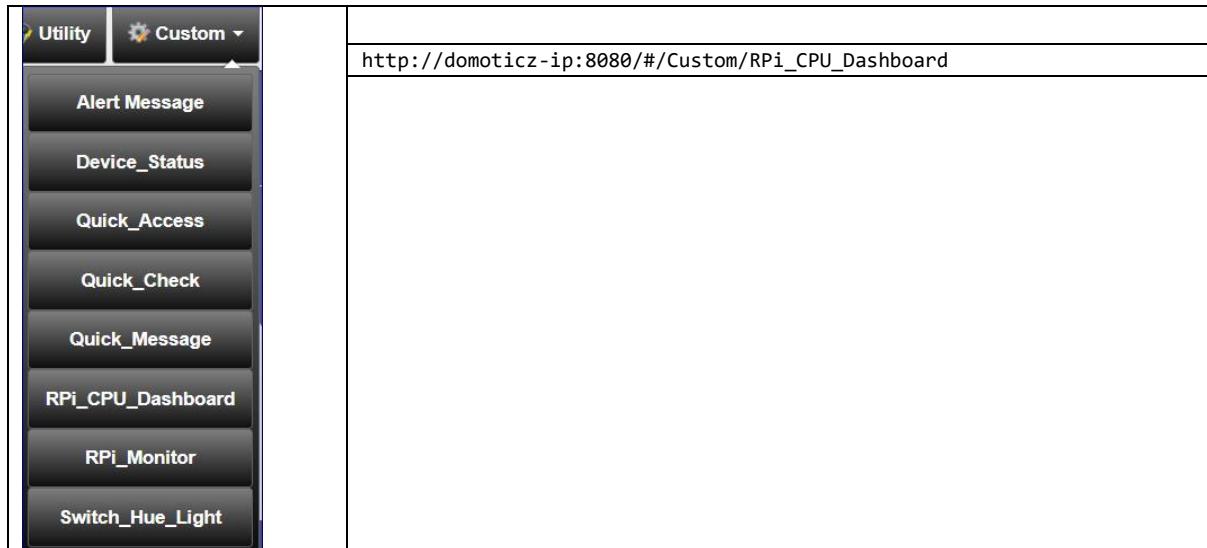
OR

Access a custom page via Browser URL: <http://domoticz-ip:8080/#/Custom/Filename>

Use the filename without .html extension.

### Example RPi\_CPU\_Dashboard

GUI	Browser URL
-----	-------------



## Custom Menu Sort

The custom menu is not sorted.

Status 20200824 with Domoticz 2020.2 (Build 12269) BETA channel.

**Ensure** to check future Domoticz versions for an option to sort the custom menu.

### Workaround

In the JavaScript file

```
~domoticz/www/app/app.js
```

add the **sort()** function to the **data.result.templates** array.

**Change** (at line 441 - Status 2020.2 build 12277)

```
...
if (typeof data.result.templates != 'undefined') {
    var customHTML = "";
    $.each(data.result.templates, function (i, item) {
        var cFile = item;
        ...
    })
    ...
}
```

**To**

```
...
if (typeof data.result.templates != 'undefined') {
    var customHTML = "";
    $.each(data.result.templates.sort(), function (i, item) {
        var cFile = item;
        ...
    })
    ...
}
```

### Notes

Every time Domoticz is updated and the Custom Menu is not sorted, set the workaround.

If the custom menu has not changed, restart Domoticz, i.e. on the RPi run from the CLI *sudo service domoticz.sh restart*) and clear the browser cache.

## How is the array `data.result.templates` filled?

This is done by running the HTML API function “getconfig”:

```
http://domoticz-ip:8080/json.htm?type=command&param=getconfig
```

### Result

```
{
  "AllowWidgetOrdering" : true,
  "DashboardType" : 0,
  "DegreeDaysBaseTemperature" : 18.0,
  "FiveMinuteHistoryDays" : 1,
  "MobileType" : 0,
  "TempScale" : 1.0,
  "TempSign" : "C",
  "WindScale" : 1.0,
  "WindSign" : "m/s",
  "language" : "en",
  "result" :
  {
    "EnableTabCustom" : true,
    "EnableTabDashboard" : true,
    "EnableTabFloorplans" : false,
    "EnableTabLights" : true,
    "EnableTabProxy" : false,
    "EnableTabScenes" : true,
    "EnableTabTemp" : true,
    "EnableTabUtility" : true,
    "EnableTabWeather" : true,
    "ShowUpdatedEffect" : false,
    "templates" :
    [
      "Switch_Hue_Light",
      "Quick_Check",
      "Alert_Message",
      "Device_Status",
      "Quick_Message",
      "RPi_Monitor"
    ]
  },
  "status" : "OK",
  "title" : "GetConfig"
}
```

The key `result.templates` is not sorted.

## External JavaScript Plugins

There are loads of external JavaScript plugins available. Started exploring how to use external plugins.

A first start has been made with the example [Raspberry Pi CPU Dashboard](#).

This solution uses the external JavaScript plugin [JustGage](#).

To use the plugin, modifications have been made to the Domoticz source file

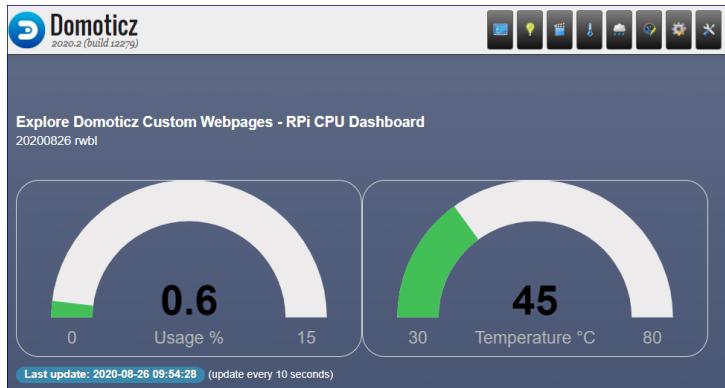
```
~domoticz/www/index.html
```

by adding references to the required JavaScript files located in the folder

```
~domoticz/www/app
```

Documented are the Domoticz source changes in the file SOURCECHANGES.md.

## Screenshot of the gauges embedded in the bootstrap grid of a custom web page



## HTTP API Requests

Domoticz HTTP API requests are used to retrieve and update data from the Domoticz database.

Tested two options to handle HTTP API requests

- `jQuery.getJSON()`
- **AJAX (Asynchronous JavaScript And XML)**

In exploring the Domoticz source code, AJAX requests seem to be used most.

AJAX uses a XMLHttpRequest object (to request data from the web server) and JavaScript & HTML DOM (to display and/or use the data).

The data returned (HTTP response) from the HTTP API request is in JSON format. Test the response using browser what key:value pairs are returned, to use those in JavaScript.

The response depends on the HTTP API command – lookup the Domoticz HTTP API documentation.

### Examples

<b>Device Update (setter)</b>	
HTTP API Command & Param	<code>type=command&amp;param=udevice&amp;idx=110</code>
HTTP Response	<code>{"status": "OK", "title": "Update Device"}</code>
JavaScript function (data)	<code>data.status</code> <code>data.title</code>
<b>Device Properties (getter)</b>	
HTTP API Command & Param	<code>type=devices&amp;rid=39</code>
HTTP Response (extract)	<pre>{     "ActTime": 1583397289,     "ServerTime": "2020-03-05 09:34:49",     "app_version": "4.11775",     "result": [         {             "Data": "0.7%",             "Name": "CPU Usage",             "idx": "39"         }     ],     "status": "OK",     "title": "Devices" }</pre>
JavaScript function (data)	<code>data.ServerTime</code> <code>data.result[0].Data</code> <code>data.status</code>

## jQuery.getJSON()

This example function uses `jQuery.getJSON()` to load JSON-encoded data from the server using a GET HTTP request.

```
var urlDomoticz = "http://domoticz-ip:8080";
var requestUrl = urlDomoticz;
var idxAlertMessage = 110;
var AlertLevel = 2;
var AlertText = "";

// get the alert message text from the full device properties
// update the text of the selector alertmessage
function init_alert_message(){
    requestUrl = urlDomoticz + "/json.htm?type=devices&rid=" + idxAlertMessage;
    $.getJSON(requestUrl,{format: "json"}, function(data){
        // Get the alert level & set the selector alertlevels
        AlertLevel = data.result[0].Level;
        $("#alertlevel").val(AlertLevel).trigger("chosen:updated");
        // Get the alert message & set selector alertmessage
        AlertText = data.result[0].Data;
        console.log(AlertText);
        $("#alertmessage").val(AlertText);
    });
}
```

## AJAX

### Option without Domoticz URL

```
getData("Domoticz Device Status", "json.htm?type=devices&used=true&displayhidden=0");
// Get all devices data from the domoticz database and create the select dropdown entries
// Hint requestUrl parameterSet:
// If this custom page is accessing on the same system, then the requesturl starts with json.htm...
// "json.htm?type=devices&used=true&displayhidden=0";
// No need add domoticzip:"http://domoticz-ip:port?json.htm?type=devices&used=true&displayhidden=0";
function getData(title, requestUrl) {
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            if(typeof data.result == "undefined"){
                console.log("No devices found (status: " + data.status + ", title: " + data.title + ")");
                return false;
            }
            // Array all device objects
            statusDevices = [];
            // String holding all devices option tag
            var optionList = "";
            optionValue = '<option value="0">Select...</option>';
            optionList += optionValue;
            // Sort the device object array by key Name
            data.result.sortBy(el => el.Name);
            // Loop over the device entries to build the option and push to the device objects array
            data.result.forEach(function(entry) {
                entryType = entry.Type;
                if (entry.Type === "General") { entryType = entry.SubType; }
                optionValue = '<option value="'+entry.idx+'">' + entry.Name+ ' - ' + entryType +'';
                optionList += optionValue;
                statusDevices.push(entry);
            });
            document.getElementById("deviceslist").innerHTML = optionList;
        },
        error: function () {
            console.log("[ERROR] Status all devices. Cannot communicate to the Domoticz server!");
            return false;
        }
    });
}
```

### Option with Domoticz URL

```
var urlDomoticz = "http://domoticz-ip:8080";
var serverTime = "";
var idxRAMUsage = 3;
var idxCPUUsage = 39;

// get a device property from the array result with 1 entry (index 0)
// example: get_device_property(idxCPUUsage, "Data","#cpuusage");
function get_device_property(idx,property,selector) {
    $.ajax({
        url: urlDomoticz + "/json.htm?type=devices&rid=" + idx,
        async: false,
        dataType: 'json',
        success: function (data) {
            propertyValue = data.result[0][property];
            console.log(propertyValue);
            $(selector).html(propertyValue);
            serverTime = data.ServerTime
            // bootbox.alert(propertyValue + "<br>" + serverTime);
        },
        error: function () {
            bootbox.alert("Error communicating to server!");
            return false;
        }
    });
    return false;
}
```

## Standalone Test Template

### jQuery.getJSON()

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="http://code.jquery.com/jquery.js"></script>
</head>
<body>
<!!-- REMOVE PREVIOUS FOR FINAL SOLUTION -->

<!-- Define the html code -->

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">
    var urlDomoticz = "http://domoticz-ip:8080";
    $(document).ready(function(){
        var requestUrl = urlDomoticz + "/json.htm?type=...";
        $.getJSON(requestUrl,{format: "json"}, function(data){
            console.log(data);
        });
    });
</script>
<!!-- REMOVE TAGS BELOW FOR FINAL SOLUTION -->
</body>
</html>
```

### AJAX

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="http://code.jquery.com/jquery.js"></script>
</head>
<body>
<!!-- REMOVE PREVIOUS FOR FINAL SOLUTION -->
<!-- Define the html code -->

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">
    $(document).ready(function(){
        var requestUrl = "json.htm?type=...";
        $.ajax({
            url: requestUrl,
            async: false,
            dataType: 'json',
            success: function (data) {
                console.log(data);
            },
            error: function () {
                // define action
                return false;
            }
        });
    });
</script>
<!!-- REMOVE TAGS BELOW FOR FINAL SOLUTION -->
</body>
</html>
```

## Alert Message (Form)

To set the text and level of an Alert device using HTML form with two fields Input & Selection.

<b>Explore Domoticz Custom Pages - Alert Message</b> 20200824 rwbl  Alert Message: <input type="text" value="Be aware: severe storm expected at 12:00."/> Select Alert Level: <input type="select" value="Red"/>  <input type="button" value="Submit"/> Press submit to set the alert message.	<b>Alert Message</b> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <b>Be aware: severe storm expected at 12:00.</b>          Last Seen: 2020-08-24 09:26:57          Type: General, Alert       </div> <div style="display: flex; justify-content: space-around; align-items: center;"> <span style="color: red;">!</span> <span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 5px; margin-right: 5px;">Log</span> <span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 5px; margin-right: 5px;">Edit</span> <span style="border: 1px solid #ccc; padding: 2px 5px; border-radius: 5px;">Notifications</span> </div>
09:08:57: Alert Message (idx=26) updated. Text: Be aware: severe storm expected at 12:00., Level: 4, Server response status: OK, title: Update Device	
<a href="http://domoticz-ip:8080/#/Custom/Alert_Message">http://domoticz-ip:8080/#/Custom/Alert_Message</a>	

Source: Alert\_Message.html

```
<!--
Domoticz custom page: Alert_Message.html (/home/pi/domoticz/www/templates)
Purpose: To set the level & text of an alert device.
Device(s) used:
Idx,Hardware,Name,Type,SubType,SwitchType
26,VirtualDevices,Alert Message,General,Alert,,

The HTML form contains:
* Input field to set the alert message
* Select element as drop-down list for the alert levels (0=gray, 1=green, 2=yellow, 3=orange, 4=red)
* Button to update the Domoticz Alert Device

When the page is loaded, the actual alert message and alert level are set.
Clicking on the button, after setting the alert message an alert level update the Domoticz device.
The Domoticz HTTP API request is used to retrieve and update data.
This function uses jQuery.getJSON() to load JSON-encoded data from the server using a GET HTTP request.

HTTP API Request Test to set the alert message & level to 3 (yellow):
http://domoticz-ip-
address:8080/json.htm?type=command&param=udevice&idx=110&nvalue=3&svalue>Hello%20World!
-->

<h3>Explore Domoticz Custom Pages - Alert Message</h3>
<p>20200824 rwbl</p>
<hr>

<form action="">
<p>
  <label for="alertmessage" id="alertmessagelabel">Alert Message:</label><br>
  <input type="text" id="alertmessage" title="Set the alert message" maxlength="200" size="50"><br>
</p>
<p>
  <label for="alertlevel">Select Alert Level:</label><br>
  <select id="alertlevel">
    <option value="0">Gray</option>
    <option value="1">Green</option>
    <option value="2">Yellow</option>
    <option value="3">Orange</option>
    <option value="4">Red</option>
  </select><br>
</p>
</form>
<hr>
<button id="setalertmessage" class="btn btn-danger">Submit</button>
<!-- Test Domoticz Result -->
```

```
<p id="status">Press submit to set the alert message.</p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
// set the url of the domoticz server
const AM_URL_DOMOTICZ = "http://domoticz-ip:8080";
// set the idx of the device
const AM_IDX_ALERTMESSAGE = 26;

// get the alert message text from the full device properties
// update the text of the selector alertmessage
function init_alert_message(url, idx){
    var requestUrl = url + "/json.htm?type=devices&rid=" + idx;
    console.log(requestUrl);
    $.getJSON(requestUrl,{format: "json"}, function(data){
        console.log(data);
        // Get the alert level & set the selector alertlevels
        var alertLevel = data.result[0].Level;
        console.log(alertLevel);
        $("#alertlevel").val(alertLevel).trigger("chosen:updated");
        // $("#alertlevel").text(data.result[0].Data);
        // Get the alert message & set selector alertmessage
        var alertText = data.result[0].Data;
        console.log(alertText);
        $("#alertmessage").val(alertText);
    });
}

function set_alert_message(url, idx){
    var dateToday = new Date();
    var strTime = dateToday.format("HH:mm:ss");
    // Define the request url to submit to the domoticz server
    var alertLevel = $("#alertlevel").children("option:selected").val();
    var alertText = $("#alertmessage").val();
    var requestUrl = url + "/json.htm?type=command&param=udevice&idx=" + idx + "&nvalue=" + alertLevel
+ "&svalue=" + alertText;
    console.log(requestUrl);
    // Submit the request and check the json result. example:
    // {"status" : "OK","title" : "Update Device"}{
    // Load JSON-encoded data from the server using a GET HTTP request.
    // The response JSON keys are accessed using parameter data.status, data.title
    $.getJSON(requestUrl,{format: "json"}, function(data){
        // update the selector with id huestatus
        $("#status").text(strTime + ": Alert Message (idx=" + idx + ") updated. Text: " + alertText + ", Level: " + alertLevel + ", Server response status: " + data.status + ", title: " + data.title);
    });
}

$(document).ready(function(){
    init_alert_message(AM_URL_DOMOTICZ, AM_IDX_ALERTMESSAGE);
    // assign the click function to the hue input checkbox with id hueswitch
    $("#setalertmessage").click(function(){
        set_alert_message(AM_URL_DOMOTICZ, AM_IDX_ALERTMESSAGE)
    });
});
</script>
```

## Quick Message (Bootbox)

To set the text and level of an Alert device via Bootstrap dialog with two fields Input & Selection.



Source: Quick\_Message.html

```
<!--
Domoticz custom page: Quick_Message.html (/home/pi/domoticz/www/templates)
Purpose: To set the text & level of an alert device by using bootbox dialog.
Idx,Hardware,Name,Type,SubType,SwitchType
26,VirtualDevices,Alert Message,General,Alert,,
20200824 by rwbl
-->

<!--
JavaScript
-->
<script type="text/javascript" charset="utf-8">
// set the url of the domoticz server
const QM_URL_DOMOTICZ = "http://domoticz-ip:8080";           // domoticz-ip
const QM_IDX_ALERT_MESSAGE = 26;
const QM_ALERT_TITLE = "Set Alert Message";

const formDefinition = '<form>'+
  '<p><input type="text" id="message" placeholder="Message" title="Set the message" style="width:100%;max-width:100%;" maxlength="200"></p>'+
  '<p><label for="level" id="levellabel" style="font-weight:bold; color:red;">Level: </label>'+
  '<input type="number" id="level" min="0" max="4" placeholder="1" value="1" title="Set the message level"></p>'+
  '<span style="color:gray">0=Gray, 1=Green, 2=Yellow, 3=Orange, 4=Red</span>'+
  '</form>';

function setMessage(title,url,idx) {
  var form = $(formDefinition);
  bootbox.confirm({
    title: title,
    message: form,
    callback: function (result) {
      console.log('This was logged in the callback: ' + result);
      if (result === true) {
        var alertMessage = form.find('input[id=message]').val();
        var alertLevel = form.find('input[id=level]').val();
        var requestUrl = url + "/json.htm?type=command&param=udevice&idx=" + idx + "&nvalue=" +
        alertLevel + "&svalue=" + alertMessage;
        console.log(requestUrl);
        $.ajax({
          url: requestUrl,
          async: false,
          dataType: 'json',
          success: function (data) {
            console.log("Set alert message successfully (status: " + data.status + ", title: " +
            data.title + ")");
            /* Console log example:
            This was logged in the callback: true
            http://dom-ip:8080/json.htm?type=command&param=udevice&idx=110&nvalue=1&svalue=This is a new message
            Set alert message successfully (status: OK, title: Update Device).
            */
          },
          error: function () {
        }
      }
    }
  })
}

setMessage(QM_ALERT_TITLE,QM_URL_DOMOTICZ,QM_IDX_ALERT_MESSAGE);
```

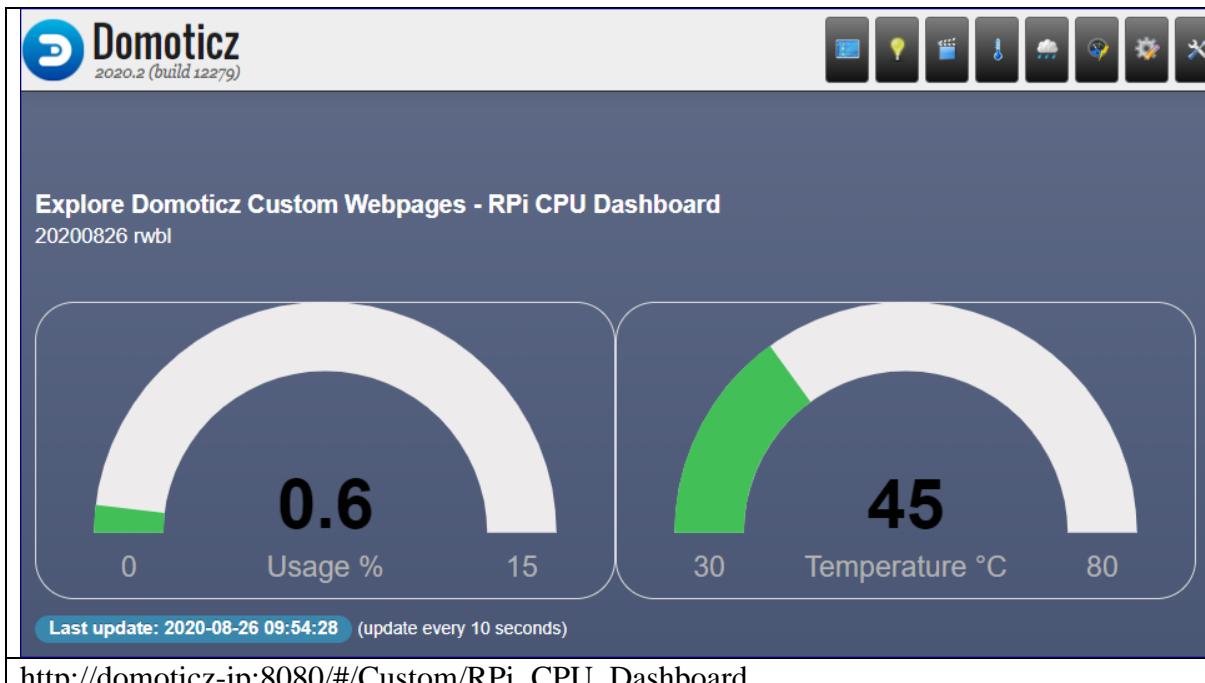
```
        console.log("[ERROR] Set alert message. Cannot communicate to the Domoticz server!");
        return false;
    }
});
}
else {
    console.log("Set alert message cancelled.");
}
});
}

$(document).ready(function(){
    setMessage(QM_ALERT_TITLE, QM_URL_DOMOTICZ, QM_IDX_ALERT_MESSAGE)
});

</script>
```

## Raspberry Pi CPU Dashboard (JustGage & Bootstrap Grid)

To monitor the RPi CPU Usage & Temperature in gauges, using the JavaScript plugin JustGage, embedded in Bootstrap grid with responsive behaviour.



Source: [RPi\\_CPU\\_Dashboard.html](#)

```
<!--
Domoticz custom page: RPi_CPU_Dashboard.html (/home/pi/domoticz/www/templates)
Purpose: To display the Raspberry Pi CPU Usage & CPU Temperature each in a gauge.
Device(s)
(Idx,Hardware,Name,Type,SubType,SwitchType):
13,RPi Hardware,RPi CPU Usage,General,Percentage,,,
14,RPi Hardware,RPi CPU Temperature,Temp,LaCrosse TX3,,

The HTML contains two gauges (JustGage) for each of the CPU data

The Domoticz HTTP API request is used to retrieve and update data.
This function uses ajax request to load JSON-encoded data from the server using a GET HTTP request.

HTTP API Request Test for single device to obtain properties in json format:
http://domoticz-ip-address:8080/json.htm?type=devices&rid=13
For the CPU Usage the data key is: "Data" : "0.52%" used as the propertyvalue.
For the CPU Temperature the data key is: "Data" : "47.0 C%" used as the propertyvalue.
The percentage & egrave sign are stripped from the data to display the value in the gauge.

To use the JustGage, the files raphael.min.js and justgauge.js are copied to the www/js folder.
The index ~domoticz/www/index.html is amended with:
<script charset="utf-8" src="js/raphael.min.js"></script>
<script charset="utf-8" src="js/justgauge.js"></script>
After adding, ensure to restart Domoticz: sudo service domoticz.sh restart

The gauges are display in a bootstrap grid (read here: https://getbootstrap.com/docs/3.4/css/)

TODO
* Instead amending www/index.html with the javascripts files, try to use the javascript files from the templates folder using script src.

20200826 rwbl
-->
<!-- Set the style of a gauge 50% boxed -->
```

```

<style>
.gauge {
    width: 50%;
    height: 50%;
    float: left;
    border: 1px solid #ddd;
    border-radius: 25px;
    box-sizing: border-box;
    margin: 20px 0px 10px 0px;
}
</style>

<!-- Create the bootstrap grid with 3 row and cols 1x12,2x6,1x12 --&gt;
&lt;div id="gauges" class="container"&gt;
&lt;div class="row"&gt;
&lt;div class="col-xs-6 col-md-12"&gt;
    &lt;h3&gt;Explore Domoticz Custom Webpages - RPi CPU Dashboard&lt;/h3&gt;
    &lt;p&gt;20200826 rwbl&lt;/p&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;div class="row"&gt;
&lt;div class="col-xs-6 col-md-6"&gt;
    &lt;!-- &lt;div class="badge badge-warning"&gt;CPU Usage %&lt;/div&gt; --&gt;
    &lt;div id="cpuusagegauge" class="gauge"&gt;&lt;/div&gt;
&lt;/div&gt;
&lt;div class="col-xs-6 col-md-6"&gt;
    &lt;!-- &lt;div class="badge badge-important"&gt;CPU Temperature °C&lt;/div&gt; --&gt;
    &lt;div id="cputemperaturegauge" class="gauge"&gt;&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;div class="row"&gt;
&lt;div class="col-xs-6 col-md-12"&gt;
    &lt;span id="status" class="badge badge-info"&gt;&lt;/span&gt;
    &lt;span&gt;&lt;small&gt;(update every 10 seconds)&lt;/small&gt;&lt;/span&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;

&lt;!--
JavaScript
Testing standalone this path is working:
Note: Raphael must be included before justgage
&lt;script src="http://domoticz-ip:8080/templates/raphael.min.js"&gt;&lt;/script&gt;
&lt;script src="http://domoticz-ip:8080/templates/justgage.js"&gt;&lt;/script&gt;
--&gt;

&lt;!-- JustGage JavaScripts files are added to ~domoticz/www/index.html
&lt;script charset="utf-8" type="module" src="templates/raphael.min.js"&gt;&lt;/script&gt;
&lt;script charset="utf-8" type="module" src="templates/justgage.js"&gt;&lt;/script&gt;
--&gt;
&lt;script type="text/javascript" charset="utf-8"&gt;
// set the url of the domoticz server
var RM_URL_DOMOTICZ = "http://domoticz-ip:8080";
// set the idx of the devices
var RM_IDX_CPU_USAGE = 13;
var RM_IDX_CPU_TEMPERATURE = 14;
// CPU Usage: Create a gaugeobject
var cpuusagegauge = new JustGage({
    id: "cpuusagegauge", // the id of the html element
    value: 0,
    min: 0,
    max: 15,
    decimals: 1,
    gaugeWidthScale: 0.8,
    valueMinFontSize: "24",
    relativeGaugeSize: true,
    customSectors: {percents: false,ranges: [{color : "#43bf58",lo : 0,hi : 10},{color : "#ff3b30",lo : 11,hi : 15}]},
    label: "Usage %"
});
// CPU Temperature: Create a gaugeobject
var cputemperaturegauge = new JustGage({
    id: "cputemperaturegauge", // the id of the html element
    value: 0,
    min: 30,
    max: 50,
    decimals: 0,
    gaugeWidthScale: 0.8,
    valueMinFontSize: "24",
    relativeGaugeSize: true,
    customSectors: {percents: false,ranges: [{color : "#43bf58",lo : 30,hi : 40},{color : "#ff3b30",lo : 41,hi : 50}]},
    label: "Temperature °C"
});
</pre>

```

```
max: 80,
decimals: 0,
gaugeWidthScale: 0.8,
valueMinFontSize: "24",
relativeGaugeSize: true,
customSectors: {percents: false,ranges: [{color : "#43bf58",lo : 0,hi : 70},{color : "#ff3b30",lo : 71,hi : 100}]},
label: "Temperature °C"
});

// update the selector with id status
function setStatus(msg) {
  $("#status").html(msg);
  // console.log(msg);
}

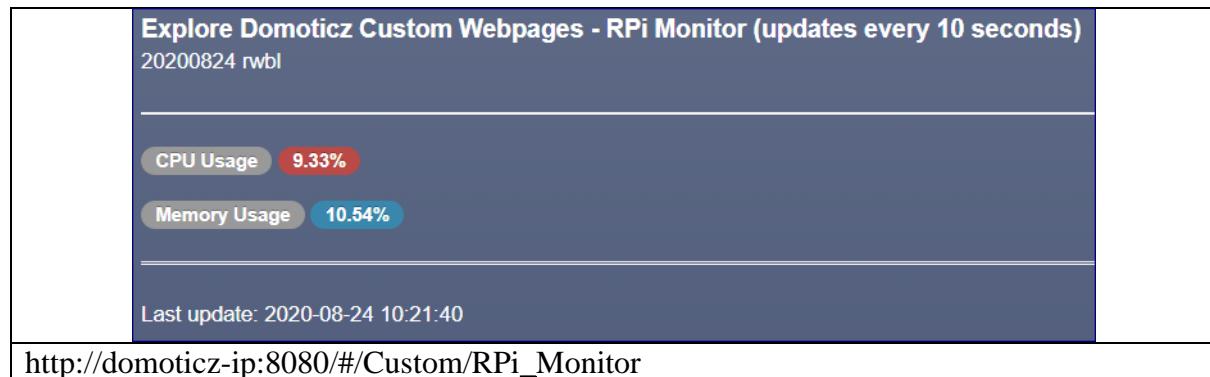
// get a device property from the array result with 1 entry (index 0)
// example:
// getDeviceProperty(RM_URL_DOMOTICZ, RM_IDX_CPU_TEMPERATURE, "Data","#cputemperature","
C",cputemperaturegauge);
function getDeviceProperty(url,idx,property,selector,replaceunit,gauge) {
$.ajax({
  url: url + "/json.htm?type=devices&rid=" + idx,
  async: false,
  dataType: 'json',
  success: function (data) {
    // Get the property value
    propertyValue = data.result[0][property].replace(replaceunit,"");
    console.log(propertyValue);
    $(selector).html(propertyValue);
    serverTime = data.ServerTime
    gauge.refresh(propertyValue);
  },
  error: function () {
    bootbox.alert("Error communicating to server!");
    return false;
  }
});
return true;
}

// get the device properties and update the html selector
function refresh_data(){
  clearInterval($.refreshTimer);
  getDeviceProperty(RM_URL_DOMOTICZ, RM_IDX_CPU_USAGE, "Data","#cpuusage","%",cpuusagegauge);
  getDeviceProperty(RM_URL_DOMOTICZ, RM_IDX_CPU_TEMPERATURE, "Data","#cputemperature","
C",cputemperaturegauge);
  setStatus("Last update: " + serverTime);
  $.refreshTimer = setInterval(refresh_data, 10000);
}

$(document).ready(function(){
  // Regular data updates
  refresh_data();
});
</script>
```

## Raspberry Pi Monitor (Badges)

To monitor the RPi CPU & Memory Usage using badges updated every 10 seconds.



Source: [RPi\\_Monitor.html](#)

```
<!--
Domoticz custom page: RPi_Monitor.html (/home/pi/domoticz/www/templates)
Purpose: To obtain Raspberry Pi CPU & RAM Usage
Device(s)
(Idx,Hardware,Name,Type,SubType,SwitchType):
13,RPi Hardware,RPi CPU Usage,General,Percentage,
15,RPi Hardware,RPi Memory Usage,General,Percentage,

The HTML form contains:
* Labels for the RPi CPU & Memory Usage

The Domoticz HTTP API request is used to retrieve and update data.
This function uses ajax request to load JSON-encoded data from the server using a GET HTTP request.

HTTP API Request Test for single device to obtain properties in json format:
http://domoticz-ip-address:8080/json.htm?type=devices&rid=9

TODO:
* Instead setting hardcoded label with text use the device property Name (data.result[0].name).
* Check message (F12): Use of XMLHttpRequest with the synchronous flag set to true is deprecated due to
its impact on user-perceived site performance.
-->

<h3>Explore Domoticz Custom Webpages - RPi Monitor (updates every 10 seconds)</h3>
<p>20200824 rwbl</p>
<hr>

<form action="">
  <label for="cpuusage" id="cpuusagelabel" class="badge">CPU Usage</label>
  <label id="cpuusage" class="badge badge-important"></label><br>
  <br>
  <label for="memoryusage" id="memoryusagelabel" class="badge">Memory Usage</label>
  <label id="memoryusage" class="badge badge-info"></label><br>
</form>
<hr>
<p id="status"></p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  var RM_URL_DOMOTICZ = "http://domoticz-ip:8080";
  // set the idx of the devices
  var RM_IDX_CPU_USAGE = 13;
  var RM_IDX_MEMORY_USAGE = 15;

  // update the selector with id status
  function setStatus(msg) {
    $("#status").html(msg);
    // console.log(msg);
  }
</script>
```

```
}

// get a device property from the array result with 1 entry (index 0)
// example: get_device_property(idxCPUUsage, "Data","#cpuusage");
function get_device_property(url, idx, property, selector) {
    $.ajax({
        url: url + "/json.htm?type=devices&rid=" + idx,
        async: false,
        dataType: 'json',
        success: function (data) {
            propertyValue = data.result[0][property];
            console.log(propertyValue);
            $(selector).html(propertyValue);
            serverTime = data.ServerTime
            // bootbox.alert(propertyValue + "<br>" + serverTime);
        },
        error: function () {
            bootbox.alert("Error communicating to server!");
            return false;
        }
    });
    return true;
}

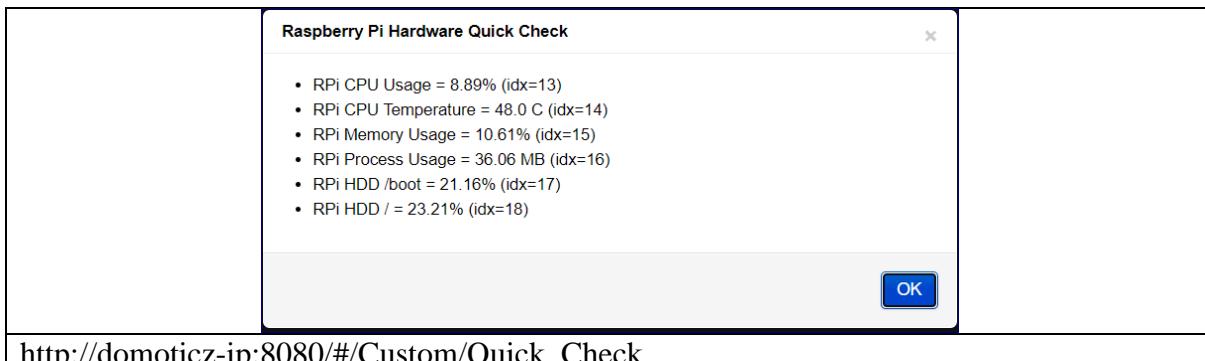
// get the device properties and update the html selector
function refresh_data(){
    clearInterval($.refreshTimer);
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_CPU_USAGE, "Data","#cpuusage");
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_MEMORY_USAGE, "Data","#memoryusage");
    setStatus("Last update: " + serverTime);
    $.refreshTimer = setInterval(refresh_data, 10000);
}

$(document).ready(function(){
    // Testing without refresh_data
    /*
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_CPU_USAGE, "Data","#cpuusage");
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_MEMORY_USAGE, "Data","#memoryusage");
    setStatus("Last update: " + serverTime);
    */

    // Regular data updates
    refresh_data();
});
</script>
```

## Raspberry Pi Quick Check (Bootbox)

To check the state of the RPi hardware in a Bootbox dialog.



[http://domoticz-ip:8080/#/Custom/Quick\\_Check](http://domoticz-ip:8080/#/Custom/Quick_Check)

Source: Quick\_Check.html

```
<!--
Domoticz custom page: Quick_Check.html (/home/pi/domoticz/www/templates)
Purpose: To quickly check some of the raspberry pi device data by using bootbox dialog.
Device(s) used: Idx and data properties
20200824 rwbl
-->

<!--
JavaScript
-->
<script type="text/javascript" charset="utf-8">
// set the url of the domoticz server
const QC_URL_DOMOTICZ = "http://domoticz-ip:8080";
const QC_DIALOG_TITLE = "Raspberry Pi Hardware Quick Check";

function getData(title, url) {
    var searchDevices = [13,14,15,16,17,18];
    console.log(searchDevices);
    var requestUrl = url + "/json.htm?type=devices&used=true&displayhidden=0";
    console.log(requestUrl);
    $.ajax({
        url: requestUrl,
        async: false,
        dataType: 'json',
        success: function (data) {
            // check there are entries in the history log
            if(typeof data.result == "undefined"){
                // console.log(data);
                console.log("Status all devices - no devices found (status: " + data.status + ", title: " + data.title + ")");
                return false;
            }
            console.log("Status all devices - Succesfully retreived (status: " + data.status + ", title: " + data.title + ")");
            statusDevices = [];
            var listDevices = "<ul>";
            data.result.forEach(function(entry) {
                // console.log(entry.idx + ", " + entry.Name + ", " + entry.Data);
                if (searchDevices.includes(Number(entry.idx))) {
                    listDevices += "<li>" + entry.Name + " = " + entry.Data + " (idx=" + entry.idx + ")</li>";
                    console.log("Pushing " + entry.idx + ", " + entry.Name + ", " + entry.Data);
                    statusDevices.push(entry);
                }
            });
            listDevices += "</ul>";
            console.log("Status all devices - Done");
            setMessage(title, listDevices);
        },
        error: function () {
            console.log("[ERROR] Status all devices. Cannot communicate to the Domoticz server!");
            return false;
        }
    });
}
```

```
        }
    });

function setMessage(title, msg) {
    bootbox.alert({
        title: title,
        message: msg,
        callback: function (result) {
            console.log('This was logged in the callback: ' + result);
        }
    });
}

$(document).ready(function(){
    getData(QC_DIALOG_TITLE, QC_URL_DOMOTICZ);
});

</script>
```

## Switch Hue Light (Form)

To switch an Hue Light On or Off using a checkbox.

Explore Domoticz Custom Pages - Switch Hue Light 20200315 rwbl <hr/> <input checked="" type="checkbox"/> Hue MakeLab (idx=118) <hr/> Last update: 2020-08-24 10:34:05	
<a href="http://domoticz-ip:8080/#/Custom/Switch_Hue_Light">http://domoticz-ip:8080/#/Custom/Switch_Hue_Light</a>	

Source: Switch\_Hue\_Light.html

```

<!--
Domoticz custom page: Switch_Hue_Light.html (/home/pi/domoticz/www/templates)
Purpose: To switch a Philips Hue light on / off.
Device used: Idx=118,Hardware=Philips Hue Bridge 1,Name=Hue
MakeLab,Type=Light/Switch,SubType=Switch,SwitchType=On/Off.
The switch (input type checkbox) is triggered by a Domoticz HTTP API request, which is defined &
submitted in the JavaScript function click.
This function uses jQuery.getJSON() to load JSON-encoded data from the server using a GET HTTP request.
Request URL to switch the light On/Off (case sensitive):
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=On | Off
-->

<h3>Explore Domoticz Custom Pages - Switch Hue Light</h3>
<p>20200315 rwbl</p>
<hr>

<form action="">
  <input type="checkbox" id="hueswitch" title="Select the checkbox to turn the light on">
  <label for="hueswitch" id="hueswitchlabel"></label><br>
</form>
<hr>
<p id="status">Select the checkbox to turn the light on.</p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  const SH_URL_DOMOTICZ = "http:// domoticz-ip:8080";
  // set the idx of the switch (=hue light)
  // var IDXSWITCH = 113;           // Hue WZ Rob
  const SH_IDX_SWITCH = 118;       // Hue MakeLab

  // get the status of the hue light on or off from the full device properties
  // update the selector huestatus
  function hue_status(url, idx){
    var requestUrl = url + "/json.htm?type=devices&rid=" + idx;
    // console.log(requestUrl);
    $.getJSON(requestUrl,{format: "json"}, function(data){
      // console.log(data);
      // set the checkbox depending light state = important to use off as data could contain "Off","Set
      Level...
      var ischecked = (data.result[0].Data == "Off") ? false : true;
      $("#hueswitch").prop('checked', ischecked);
      // set the switch name as label for the checkbox
      $("#hueswitchlabel").text(data.result[0].Name + " (idx=" + idx + ")");
      // document.getElementById("hueswitchlabel").innerHTML = data.result[0].Name + " (idx=" + idx +
      ")";
      // update the selector with id huestatus
      // console.log(data);
      $("#status").text("Hue Light " + idx + ", " + data.result[0].Name + ", Data: " +
      data.result[0].Data);
      //document.getElementById("huestatus").innerHTML = "Hue Light " + idx + ", " + data.result[0].Name
      + ", Data: " + data.result[0].Data;
    });
  }
</script>

```

```
}
```

```
function hue_switch(url, idx){
    // Get the current time
    var dateToday = new Date();
    var strTime = dateToday.format("hh:mm:ss");
    // get the togge button state
    var state = $("#hueswitch").is(":checked");
    // Define the switch command - ensure to use correct syntax for On and Off
    var switchCmd = (state == true) ? "On" : "Off";
    console.log(state);
    // Define the request url to submit to the domoticz server
    var requestUrl = url + "/json.htm?type=command&param=switchlight&idx=" + idx + "&switchcmd=" +
    switchCmd;
    console.log(requestUrl);
    // Submit the request and check the json result. example: {"status" : "OK","title" : "SwitchLight"}
    // Load JSON-encoded data from the server using a GET HTTP request.
    // The response JSON keys are accessed using parameter data.status, data.title
    $.getJSON(requestUrl,{format: "json"}, function(data){
        // update the selector with id huestatus
        $("#status").text(strTime + ": Hue Light (idx=" + idx + ") switched: " + switchCmd + ", Server
response status: " + data.status + ", title: " + data.title);
        // document.getElementById("huestatus").innerHTML = "Hue Light " + IDXSWITCH + " switched: " +
        switchCmd + ", Server Response title: " + data.title + ", status: " + data.status;
    });
}

$(document).ready(function(){
    hue_status(SH_URL_DOMOTICZ, SH_IDX_SWITCH);
    // assign the click function to the hue input checkbox with id hueswitch
    $("#hueswitch").click(function(){
        hue_switch(SH_URL_DOMOTICZ, SH_IDX_SWITCH)
    });
});
</script>
```

# Custom Pages Charts

## Purpose

To explore how to create responsive Custom Pages with embedded customized charts containing single or multiple device(s) data:

- Line charts
- Angular Gauges
- Column charts
- Combined charts

The **main purpose** is how to create customized charts with data from **multiple devices**.

To mention is that responsiveness is important as the charts are used on several devices, from PC to Mobile ... but depending the device (and its OS) used, the custom page layout differs. Experienced: Complex chart rendering on mobile devices could cause hard to read charts.

These JavaScript chart libraries are explored:

- [Highcharts](#) which is used by Domoticz as the charting engine.  
This enables to use the features as for the Domoticz Device Graphs/Logs.  
*The focus is this library because it is used by Domoticz as the default charting engine.*
- [Google Charts](#) as an alternative - Google Charts offer many ready-to-use chart types.  
Creating Google charts require external Internet access.

To explore how to use various [Data Sources](#) for the charts:

- Data in CSV format from Domoticz text device (DataTextCSV)
- Data as JSON object from Domoticz text device (DataTextJSON)
- Data from external file containing JSON object ((DataFileJSON))
- Data from the response of Domoticz HTTP API requests (DataHTTP)

The **example source files** are not included in this document but can be found in the [GitHub](#) repository folder [src](#) in the archive *explore\_custom\_pages\_charts.zip*.

There is a document *explore\_custom\_pages\_charts.pdf* outlining the examples.

### Notes

- The charts can use multiple datapoints from a single device or from multiple devices.
- The information in brackets is the type of data source used for the chart.
- The data is generated by Automation Event dzVents except for HTTP API requests.
- A test scenario, room Thermostat, is used to create the various charts.
- Other custom chart examples are created as well as using other devices.

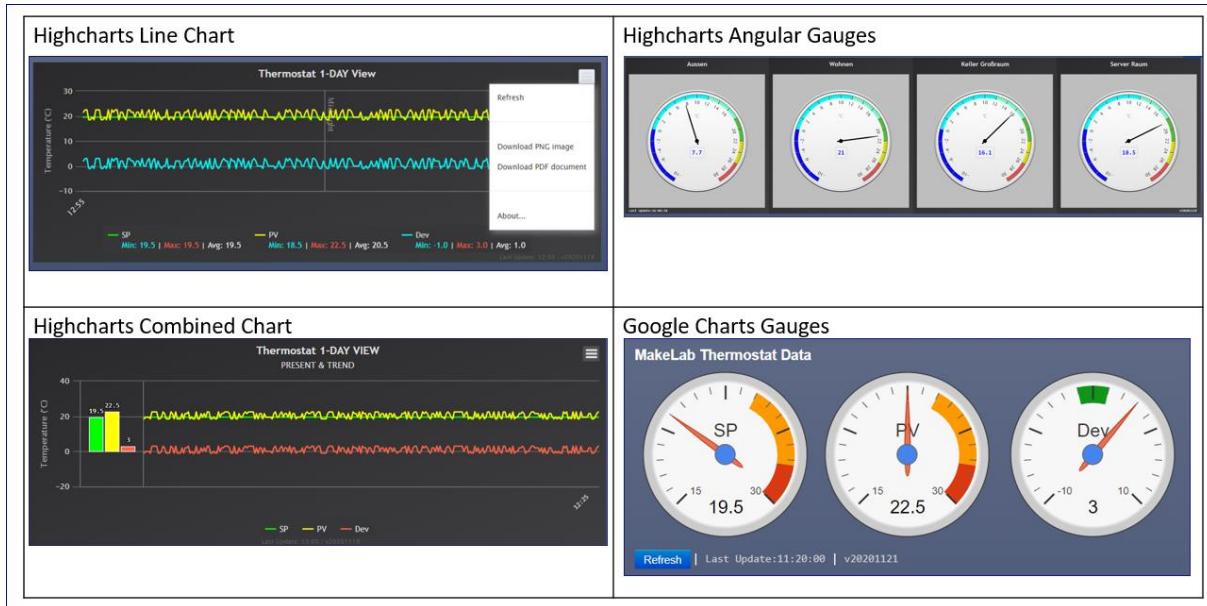
### Be Aware

This chapter is (just) an entry point on how to create charts.

Whilst exploring, found so much information on building all kinds of chart solutions.

Basically, had to stop creating more examples ... made so much fun.

## Screenshot Customized Highcharts & Google Charts

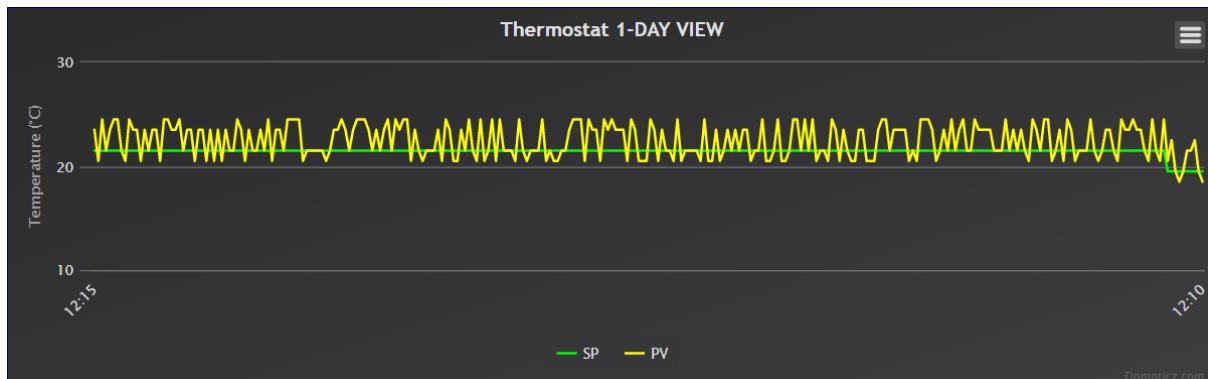


# Get Started

Prior outlining the concept with several solutions, let us start with a simple Highcharts Line Chart.

## Line Chart

The line chart displays the Thermostat Setpoint (SP) and Thermostat Temperature (PV) values in a 1-DAY VIEW.



Source: [highcharts\linecharts\datagetcsv\thermostat-1dayview.html](http://highcharts.com/linecharts/datagetcsv/thermostat-1dayview.html)

## Data Store

The line chart gets its data from a Domoticz text device acting as the data store. The text value is a CSV string in format (using semicolon as separator):

```
HHMM;SP;PV.
```

### Note

Check the Domoticz Log History settings

For this example, using 1 day (GUI > Settings > Log History > Light/Switches).

## Data Supplier

An Domoticz Automation Event dzVents runs every 5 minutes and stores the values in the text device.

```
-- thermostat_data_textcsv.dzvents
-- Idx of the devices
local IDX_SP = 30    -- Thermostat Temperature SetPoint (SP); Device: Thermostat, SetPoint
local IDX_PV = 31    -- Thermostat Temperature Process Value (PV); Device: Temp, LaCrosse TX3
local IDX_TS = 42    -- Text device holding the data as CSV string; Device: General, Text

return
{
  on = {
    timer = {
      'every 5 minutes at 00:00-23:59',
    },
  },
  logging = {
    level = domoticz.LOG_INFO, marker = 'THERMOSTATDATATEXTCSV',
  },
  execute = function(domoticz)
    -- Store new data in virtual text as CSV string (max length 200): hhmm,sp,pv
    local data = ("%s%%;%.1f;%.1f"):format(
      string.format("%02d", domoticz.time.hour),
      string.format("%02d", domoticz.time.minutes),
      domoticz.utils.round(domoticz.devices(IDX_SP).setPoint, 1),
      domoticz.utils.round(domoticz.devices(IDX_PV).temperature, 1)
    )
  end
}
```

```

        )
        -- Log the value & update text device
        domoticz.log(data)
        domoticz.devices(IDX_TS).updateText(data);
    end
}

```

## Custom Page

The Custom Page *thermostat\_1dayview.html*, requests, via a Domoticz HTPP API request the Text device history, from which the data for 24 hours is selected and used to draw the chart. The Custom Page has the two sections HTML (to display the chart) and JavaScript (to get the data and draw the chart).

In the page, there is no need to include any third-party libraries nor to use the HTML and BODY tags as the page is embedded in the Domoticz main page *index.html* located in folder *~domoticz/www*.

## HTML Section

The HTML section makes use of Domoticz styles (see source *~domoticz/www/css/styles.css*).

```

<div class="chart noselect">
    <div class="chartarea">
        <!-- Container to show the chart -->
        <div id="chartcontainer" class="device-log-chart"></div>
    </div>
</div>

```

### Note

The class *device-log-chart* is the default with size 300px.

There are other classes with different sizes, i.e. *device-log-chart\_md* = 200px and *device-log-chart\_sd* = 100px. Of course, could create an own custom chart container.

To cater best for responsiveness, experienced best results when sticking to the classes as defined by Domoticz for the HTML div tags

## JavaScript Section

To get the data, an asynchronous HTTP (Ajax) request is placed to the Domoticz system to get the history data.

The HTTP JSON response is parsed to get the data from the result array key. The data is added to the chart data arrays.

Some chart options are set, and the chart is drawn first. Thereafter updated every 5 minutes.

```

<script type="text/javascript" charset="utf-8">
    // Draw the chart with data arrays
    // Parameter:
    // dataCategories: xAxis labels
    // dataSP: Series for the yAxis first line
    // dataPV: Series for the yAxis second line
    function drawChart(dataCategories, dataSP, dataPV) {
        var title = {
            text: 'Thermostat 1-DAY VIEW'
        };
        var xAxis = {
            categories: dataCategories,
            labels: {
                formatter: function() {
                    if (this.isFirst) { return this.value; }
                    if (this.isLast) { return this.value; }
                }
            }
        };
        var yAxis = {
            title: {

```

```

        text: 'Temperature (\xB0C)'
    },
    min: 10,
    max: 30
};

var plotOptions = {
    series: {
        marker: {
            enabled: false
        },
        animation: {
            duration: 0
        }
    }
};

var series = [
    { name: 'SP', data: dataSP, color: '#00FF00' },
    { name: 'PV', data: dataPV, color: '#FFFF00' }
];

var json = {};
json.title = title;
json.xAxis = xAxis;
json.yAxis = yAxis;
json.plotOptions = plotOptions;
json.series = series;
$('#chartcontainer').highcharts(json);
}

// Draw the linechart
function updateChart() {
    // Log max hours
    let MAXHOURS = 24;
    // Domoticz IDX of the text device holding the devices data
    let IDX = 42;

    // AJAX call with draws the chart if successfull return
    $.ajax({
        url: "json.htm?idx=" + IDX + "&type=textlog",
        async: false,
        dataType: 'json',
        success: function (deviceData) {
            // console.log(deviceData);
            // No data found then leave
            if(typeof deviceData.result == "undefined"){
                console.log("[ERROR updateChart] No device data found (status: " + deviceData.status + ", title: " + deviceData.title + ")");
                $('#chartinfo').html("[ERROR] No device data found.");
                return false;
            }
            console.log("[INFO updateChart] Data retrieved, #Entries: " + deviceData.result.length + "(status: " + deviceData.status + ", title: " + deviceData.title + ")");
            // Add the data rows, in reverse order
            // Item data
            var dataArray;
            // Chart data arrays xaxis (#1), yaxis (#2)
            var dataCategories = [];
            var dataSP = [];
            var dataPV = [];
            deviceData.result.reverse().forEach(function(entry) {
                // Split the data csv string into the HHMM,sp,pv items (array size 3)
                dataArray = entry.Data.split(";");
                // Get the hours difference between now and the date of the entry
                var dateNow = new Date();
                var dateEntry = new Date(entry.Date);
                var hoursDiff = Math.abs(dateNow.getTime() - dateEntry.getTime()) / 3600000;
                // Get the time HH:MM from the Date key
                var timeArray = dateEntry.toTimeString().split(":");
                // If the entry hour is within MAXHOURS then row add entry: HH:MM, sp, pv
                if (hoursDiff <= MAXHOURS) {
                    dataCategories.push(timeArray[0]+":"+timeArray[1]);
                    dataSP.push(Number(dataArray[1]));
                    dataPV.push(Number(dataArray[2]));
                }
            });
        }
    });
}

```

```
// Draw the chart
drawChart(dataCategories, dataSP, dataPV);
},
error: function () {
    console.log("[ERROR] Device Data. Cannot communicate to the Domoticz server!");
    return false;
}
});

// Load the document and refresh every 300 seconds
$(document).ready(function(){
    // Show the chart first
    updateChart();
    // Timer triggered chart update
    setInterval(updateChart, 300000);
});

</script>
```

## Sources

The sources of the Automation Events dzVents and the Custom Pages HTML files are not included in this document but can be found in the [GitHub](#) repository folder [src](#) in the archive *explore\_custom\_pages\_charts.zip*.

The sources are well documented and contain more in-depth information.  
There is a document *explore\_custom\_pages\_charts.pdf* outlining the examples.

The archive is structured by the chart libraries Highcharts & Google Charts.

The chart libraries are structured by chart type with subfolders for data source.

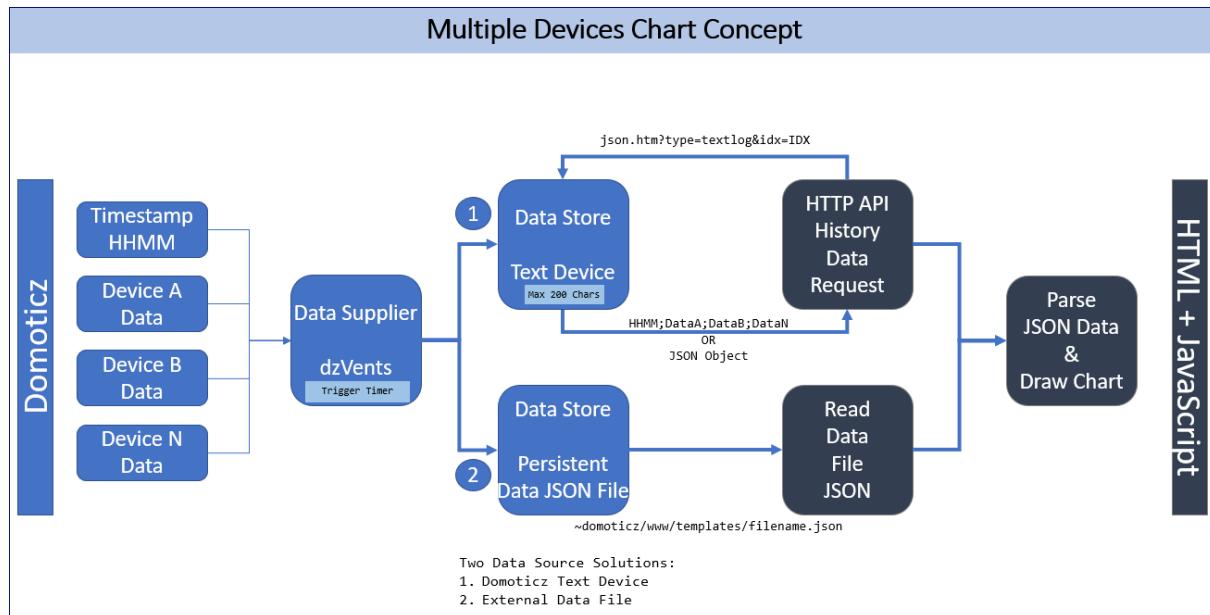
### Example File

Highcharts line chart with data source Domoticz Text Device with data in CVS format showing a 1-DAY VIEW of a thermostat.

```
highcharts\linechart\dataetxtcsv\thermostat-1dayview.html
```

# Concepts

Let start outlining the flow for **multi devices** data gathering to draw the chart.



**As JSON object** with timestamp with format

```
{"timestamp": "1235", "devices": [{"name": "MakeLab Thermostat Setpoint", "data": 17.5}, {"name": "MakeLab Temperature", "data": 19.5}]} 
```

The data is then requested by the Custom Page via [Domoticz HTTP API](#) request for [History](#) or [Status](#) data from the text device.

The HTTP response is parsed and used to draw the chart in the custom page.

*Note*

This solution is rather simple as it is fully based on Domoticz devices with HTTP API requests retrieving data from the Domoticz database.

No need to use persistent data or local file(s) to store the data.

**Limitation:**

The text device value has a maximum of 200 characters – in case more data required additional text devices are required.

## Solution 2 – External JSON File

The automation event uses persistent data, which is stored in an external file as JSON object. The Custom Page reads & parses the content of the file to draw the chart in the custom page.

Example of a file located:

```
~domoticz/www/templates/thermostat_1dayview_data.json
```

```
{
  "result" : [
    {"pv":16.5,"sp":17.5,"time":"12:38"},  
    ...  
  "status":"OK",  
  "title":"Thermostat Data"  
}
```

The Automation Event controls the number of datapoints:

```
local MAXDATAITEMS = 288 -- 288=24h * (60/5) for one day sampling every 5 minutes  
  
data = {  
  rows = {  
    history = true, maxItems = MAXDATAITEMS,  

```

## Custom Page

A Custom Page has two main sections:

- HTML for styling & page layout
- JavaScript for the logic, data supply and chart drawing

The goal is to use the Domoticz defined styles and classes as defined in the various `~domoticz/www` folders.

To mention are the folders – recommend to lookup the content:

- `css` - esp. `styles.css`, `bootstrap.css`
- `app/log` - containing the Domoticz Log layouts

### Note

Whilst exploring Custom Page ensure to regular clear the cache of the browser prior refreshing the page or custom menu.

If the browser response is getting slow, restart Domoticz.

### HTML Style Section

The definition of additional styles is kept to a minimum.

Some examples are:

- `infotext` - display info type of information mostly below the chart area
- `reload` - for a button to refresh the chart

```
<style>
.infotext {
    font-size: 11px;
    font-family: Consolas;
    color: lightgray;
    padding: 5px;
}
.reload {
    font-family: Lucida Sans Unicode;
    font-size: 12px;
    font-weight: bold;
}
</style>
```

## HTML Layout Section

The layout is within a div section with class *chart noselect*.

Within the section there is

- the chart area with (optional) embedded buttons and
- the (mandatory) chart container displaying the chart with options (burger menu)

Below the chart area, there is (again optional) a table with information.

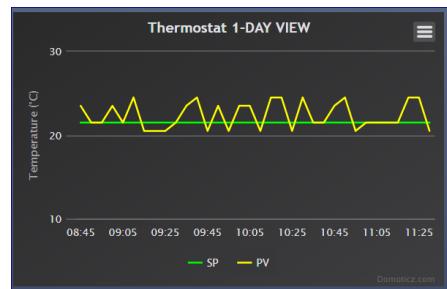
### Notes

- There are also examples where the functions of the embedded buttons are included in the Burger Menu as Custom menu Items.
- Some charts use Multiple *div* tags with style *inline-block* in the chart area to have multiple charts on a single row.
- The class *device-log-chart* sets the size of the chart to 300px (see *styles.css*). Other sizes are *device-log-chart\_md* (200px) and *device-log-chart\_sm* (100px). For other sizes, define own class(es) in the HTML style section.

## Some Line Chart Examples

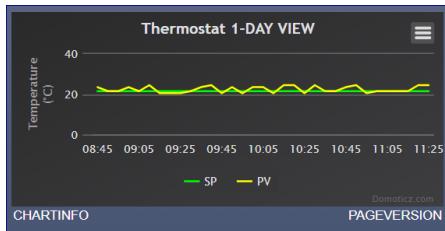
### Basic Chart (default size)

```
<div class="chart noselect">
  <div class="chartarea">
    <!-- Container to show the chart set by JavaScript-->
    <div id="chartcontainer" class="device-log-chart"></div>
  </div>
</div>
```



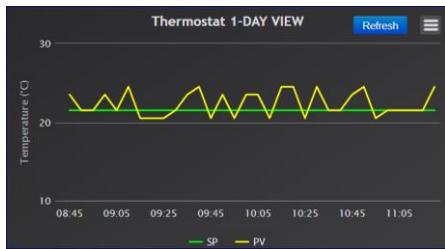
### Basic Chart (size md) with Chart Area Footer

```
<div class="chart noselect">
  <div class="chartarea">
    <!-- Container to show the chart -->
    <div id="chartcontainer" class="device-log-chart_md"></div>
    <table style="width:100%;">
      <tr>
        <td style="width:50%;">
          <div id="chartinfo" class="infotext">CHARTINFO</div>
        </td>
        <td style="width:50%;">
          <div id="pageversion" class="infotext" align="right">PAGEVERSION</div>
        </td>
      </tr>
    </table>
  </div>
</div>
```



### Basic Chart with Refresh Button in Zoom Area

```
<div class="chart noselect">
  <div class="chartarea">
    <div class="zoom-buttons-container">
      <button class="btn btn-small btn-primary" title="Refresh" onclick="refreshChart()">
        Refresh
      </button>
    </div>
    <!-- Container to show the chart -->
    <div id="chartcontainer" class="device-log-chart"></div>
  </div>
</div>
```



### Two Basic Charts on one Row small Size

```
<div class="chart noselect">
  <div class="chartarea">
    <!-- Container to show the chart -->
    <div id="chartcontainer1" style="height: 120px; display: inline-block;"></div>
    <div id="chartcontainer2" style="height: 120px; display: inline-block;"></div>
  </div>
</div>
```



#### Note

Preferrable want to have a table with 50% columns.

Tested but the charts lost its responsiveness, i.e. did not resize.

<TODO>: Explore how to create responsive tables containing charts.

## JavaScript Section

This is (bit) more complex.

It starts with a function to draw the chart with the required data for the axis as parameter.  
Any text is hardcoded in the function.

TODO: For next solutions, the text and any other parameter are considered to put in a JSON object.

Depending the chart type, several vars are defined as JSON objects holding the initial declarative chart setup.

The chart is drawn by using the function *Highcharts* with the JSON object and assign to the HTML selector *chartcontainer*.

### Example Function drawChart for Highcharts

```
function drawChart(dataCategories, dataSP, dataPV) {
    var title = { text: 'Chart Title' };
    var xAxis = { categories: dataCategories };
    var yAxis = { title: { text: 'DATA'}};
    var series = [
        { name: 'SP', data: dataSP, color: '#00FF00' },
        { name: 'PV', data: dataPV, color: '#FFFF00' }
    ];
    var json = {};
    json.title = title;
    json.xAxis = xAxis;
    json.yAxis = yAxis;
    json.series = series;
    $('#chartcontainer').highcharts(json);
}
```

*Note:* The example shows just a few JSON objects for the chart configuration.

The next function *updateChart* is used to get the data & draw the chart.

The data is retrieved from the Domoticz database via Ajax HTTP API request.

The HTTP API response holds the data as JSON object, which is parsed and assigned to JavaScript arrays for the chart data.

Parsing the JSON object depends on what type of chart, i.e. log chart with many datapoints or device status with only a single datapoint.

### Example Function updateChart reading data from Text Device with IDX=42

```
function updateChart() {
    let IDX = 42;
    $.ajax({
        url: "json.htm?idx=" + IDX + "&type=textlog",
        async: false,
        dataType: 'json',
        success: function (deviceData) { // No data found then leave
            if(typeof deviceData.result == "undefined"){
                console.log("[ERROR updateChart] No device data found (status: " + deviceData.status + ", title: " + deviceData.title + ")");
                return false;
            }
            var dataArray;
            var dataCategories = [];
            var dataSP = [];
            var dataPV = [];
            deviceData.result.reverse().forEach(function(entry) {
                dataArray = entry.Data.split(";");
                var dateNow = new Date();
                var dateEntry = new Date(entry.Date);
                var hoursDiff = Math.abs(dateNow.getTime() - dateEntry.getTime()) / 3600000;
```

```

        timeArray = dateEntry.toTimeString().split(":");
        dataCategories.push(timeArray[0]+":"+timeArray[1]);
        dataSP.push(Number(dataArray[1]));
        dataPV.push(Number(dataArray[2]));
    });
    var minPV = Math.min.apply(Math, dataPV);
    var maxPV = Math.max.apply(Math, dataPV);
    drawChart(dataCategories, dataSP, dataPV);
    var updated = new Date().toLocaleTimeString();
    $('#chartinfo').html('Updated:' + updated + ' | ' + 'PV Min:' + minPV + ',Max:' + maxPV);
},
error: function () {
    console.log("[ERROR] Device Data. Cannot communicate to the Domoticz server!");
    return false;
}
});
}
}

```

The JavaScript section then has some functions to

- Refresh the chart when pressing the Refresh Button defined on the chart (as zoom button)
- Set the page version

Finally, the jQuery document ready function is called to create and refresh the chart

```
// Load the document and refresh every 300 seconds
$(document).ready(function(){
    setVersion("v20201122");
    updateChart();
    setInterval(updateChart, 300000);
});
```

As mentioned, strongly recommend to read/study the sources codes of the custom pages.

## Automation Script

A dzVents snippet on how the CSV string for the text device holding multiple devices charts data:

```

-- Idx of the devices
local IDX_SP = 30 -- Thermostat Temperature SetPoint (SP); Device: Thermostat, SetPoint
local IDX_PV = 31 -- Thermostat Temperature Process Value (PV); Device: Temp, LaCrosse TX3
local IDX_TS = 42 -- Text device holding the data as CSV string; Device: General, Text

return
{
on = {
    timer = {'every 5 minutes at 00:00-23:59',},},
logging = {level = domoticz.LOG_INFO, marker = 'THERMOSTATDATA',},
},
execute = function(domoticz)
    -- Store the new data in the virtual text sensor as CSV string (max length 200): hhmm,sp,pv
    domoticz.devices(IDX_TS).updateText(("%"..s..";%.1f;%.1f"):format(
        string.format("%02d", domoticz.time.hour),
        string.format("%02d", domoticz.time.minutes),
        domoticz.utils.round(domoticz.devices(IDX_SP).setPoint, 1),
        domoticz.utils.round(domoticz.devices(IDX_PV).temperature, 1)
    )));
    -- Log the value
    domoticz.log(domoticz.devices(IDX_TS).text)
end
}
```

Next some chart examples ...

## Test Scenario

The scenario is a room "MakeLab" with a Domoticz Thermostat and a Temperature device. These two separate devices have their own data, defined as Thermostat Setpoint (SP) & Thermostat Temperature (PV, Process Value).

An Automation Event dzVents *thermostat\_simulator.dzvents* acts as a simulator to randomly set, every 5 minutes, the PV to +/- 3% of the SP.

The SP and PV values are used for the various chart solutions.

Another Automation event *thermostat\_data.dzvents* acts as the data supplier and generates every 5 minutes data. Depending the solution, the data is stored in a Text device as CSV string or JSON object or in an external file as JSON object (using persistent data).

## Domoticz Devices

The Domoticz devices for the tests are virtual sensors from type Hardware Dummy.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
30	VirtualSensors	001406E	1	MakeLab Thermostat Setpoint	Thermostat	SetPoint	21.5
31	VirtualSensors	1406F	1	MakeLab Temperature	Temp	LaCrosse TX3	24.5 C
32	VirtualSensors	00082032	1	MakeLab Thermostat Battery	General	Percentage	30%
42	VirtualSensors	00082042	1	MakeLab Thermostat Data	General	Text	1025,21.5,20.5
43	VirtualSensors	00082043	1	MakeLab Thermostat JSON	General	Text	{"timestamp": "1235", "devices": [{"name": "MakeLab Thermostat Setpoint", "data": 17.5}, {"name": "MakeLab Temperature", "data": 19.5}]}]

### Note

If the Text Device should not be made visible, add \$ as a name prefix, i.e. "\$MakeLab Thermostat Data".

## Domoticz Log History

The log history is set to 1 day in the GUI settings Log History (see GUI > Setup > Settings > Log History > Light/Switches > Days), but this can be higher because the Custom Page has a constant for MAXHOURS set to 24.



The Text Device data is stored in the Domoticz database table LightingLog.

There is no calendar table like for Temperatures or Meters.

The text device history or status data, JSON object, is used as data source for various graphs.

Here some HTTP Request Examples with response (JSON object snippets):

**Text Device History** – full data log (N datapoints):

```
http://domoticz-ip:port/json.htm?idx=IDX&type=textlog
```

```
{
  "result" :
  [
    {"Data" : "0913;18.5;21.5", "Date" : "2020-11-05 09:13:00", "User" : "", "idx" : "4"},
```

```
    ...  
],  
"status" : "OK", "title" : "TextLog"  
}
```

### Text Device Status – present state (1 datapoint):

```
http://domoticz-ip:port/json.htm?type=devices&rid=IDX
```

```
{  
  "app_version" : "2020.2 (build 12554)",  
  "result" :  
    [  
      { ... "Data" : "0913;21.0;24.0", ..., "idx" : "42" }  
    ],  
  "status" : "OK", "title" : "Devices"  
}
```

The key “result”, JSON array, contains the data for the charts.

With these 3 devices, several chart types are explored embedded in a Domoticz Custom Page:

- Line Chart – Thermostat 24 Hours with the two lines SP and PV
- Gauges – Three VU type gauges showing the present values (in Celsius) for the Thermostat Setpoint SP, Temperature PV and Control Deviation (calculated as Dev = PV – SP)
- Other charts in progress

# Highcharts

## Purpose

To explore how to create Custom Pages with embedded [Highcharts](#), with single or multiple device(s) data.

## Highcharts Library

The library is included in Domoticz and loaded by the file:

```
~domoticz/www/index.html
```

In a Custom Page, the library can be used without additional reference.

*Note*

The Highcharts library files are in the Domoticz folder

```
~domoticz/www/js/highcharts
```

Next a few examples – more can be found in the archive *explore\_custom\_pages\_charts.zip*.

## Thermostat Log

To display history log data in a **Line Chart** with 24-hour data (sampling every 5 minutes) for two devices Thermostat Temperature Setpoint (SP) and Room Temperature Process Value (PV). The deviation between PV and SP is calculated and displayed as a line.

Set Point (SP) & Temperature (PV) 24 hour data (sample rate 5 mins) with chart additions.

Thermostat 1-DAY View

Temperature (°C)

SP: Min: 19.5 | Max: 19.5 | Avg: 19.5    PV: Min: 18.5 | Max: 22.5 | Avg: 20.5    Dev: Min: -1.0 | Max: 3.0 | Avg: 1.0

Refresh   Download PNG image   Download PDF document   About...

Last update: 12:55 | 2020-11-05

```
Data Content HTTP API Response
{
  "result": [
    {
      "Data": "0913;18.5;21.5",
      "Date": "2020-11-05 09:13:00",
      "User": "",
      "idx": "4"
    },
    ...
  ],
  "status": "OK",
  "title": "TextLog"
}
```

Item	Description
File Source	highcharts\linechart\data:textcsv\thermostat-1dayview-advanced.html (archive: explore_custom_pages_charts.zip)
Data Source	Domoticz virtual sensor Text Device History Log with 1 day setting (data:textcsv)
Data Source Location	Domoticz HTTP API Request: <a href="http://domoticz-ip:port/json.htm?type=textlog&amp;idx=IDX Text Device">http://domoticz-ip:port/json.htm?type=textlog&amp;idx=IDX Text Device</a>
Data Content	CSV string: HHMM;SP;PV from the HTTP API Response key Data
Chart Additions	<ul style="list-style-type: none"> <li>• Data series as array with device name, datapoints, line color</li> <li>• Chart series legend with Min, Max, Avg</li> <li>• Plot line at midnight</li> <li>• Chart credits with last update and link to view the log of the text device with datapoints</li> <li>• Customized context menu with own function Refresh, two standard menu items (Download ...) and About Dialog (Bootbox)</li> </ul>

## Thermostat Log Present & Trend

To display the present values & trend in a **Combined Column & Line Chart**.

Set Point (SP), Temperature (PV) & Deviation (Dev) 24 hour data (sample rate 5 mins)

Thermostat 1-DAY VIEW  
PRESENT & TREND

Temperature (°C)

SP: 19.5   PV: 22.5   Dev: 3

12:55

SP   PV   Dev

Last update: 12:55 | 2020-11-05

```
Data Content HTTP API Response
{
  "result": [
    {
      "Data": "0913;19.5;21.5",
      "Date": "2020-11-05 09:13:00",
      "User": "",
      "idx": "4"
    },
    ...
  ],
  "status": "OK",
  "title": "TextLog"
}
```

Item	Description
File Source	highcharts\columnlinechart\data:textcsv\thermostat-1dayview.html (archive: explore_custom_pages_charts.zip)
Data Source	Domoticz virtual sensor Text Device History Log with 1 day setting (data:textcsv)
Data Source Location	Domoticz HTTP API Request: <a href="http://domoticz-ip:port/json.htm?type=textlog&amp;idx=IDX Text Device">http://domoticz-ip:port/json.htm?type=textlog&amp;idx=IDX Text Device</a>
Data Content	CSV string: HHMM;SP;PV from the HTTP API Response key result[NNN].Data
Chart Additions	Credits bottom centre Last Update + Version; xAixs plot lines to built box around column chart; xAixs label last value only

## Temperature Dashboard Angular Gauges

To display in **Angular Gauges** with **Speedometer** look, actual temperatures from selected room plan devices.

Four Temperature Devices selected from Domoticz Room Plan

```
Data Content HTTP API Response
{
  "ServerTime": "2020-11-09 19:28:40",
  "result": [
    {"Name": "Aussen", "Temp": 7.7, "idx": "344"}, {"Name": "Wohnen", "Temp": 21, "idx": "350"}, {"Name": "Keller Gro\u00dfer Raum", "Temp": 16.1, "idx": "351"}, {"Name": "Server Raum", "Temp": 18.5, "idx": "356"}],
  "status": "OK",
  "title": "Devices"
}
```

Item	Description
File Source	highcharts\angulargaugeschart\datahttpplan\temperature-dashboard.html (archive: explore_custom_pages_charts.zip)
Data Source	Temperature Devices data (datahttpplan)
Data Source Location	Domoticz HTTP API Request: <a href="http://domoticz-ip:port/json.htm?type=devices&amp;plan=IDX ROOMPLAN">http://domoticz-ip:port/json.htm?type=devices&amp;plan=IDX ROOMPLAN</a>
Data Content	JSON array “result” with key:value pairs
Chart Additions	4 Gauges as individual Highcharts Angular Gauges charts added to HTML DIV container as child

### Notes

The chart is inspired by [this](#) Highcharts example = Many thanks for providing.

# Google Charts

## Purpose

To explore how to create Custom Pages with embedded [Google Charts](#), with single or multiple device(s) data. Google Charts offer many ready-to-use chart types.

## Google Charts Library

Prior outlining the examples, a note about including the Google Charts Library, the loader, in JavaScript.

### Option HEAD Section

As taken from the Google Charts documentation, web pages with Google Charts should include the following lines in the <head> section

*Note*

**External Internet access is required.**

```
<script src="https://www.gstatic.com/charts/loader.js"></script>
<script>
  google.charts.load('current', {packages: ['corechart']});
  google.charts.setOnLoadCallback(drawChart);
  ...
</script>
```

This is possible by adding to the file ~domoticz/www/index.html.

Example to make a change at line 103 (with source 2020.2 build 122279):

*Change*

```
<script src="js/noty/noty.min.js"></script>
<script charset="utf-8" src="js/data-tables/jquery.dataTables.min.js"></script>
```

*To*

```
<script src="js/noty/noty.min.js"></script>


<script src="https://www.gstatic.com/charts/loader.js"></script>


<script charset="utf-8" src="js/data-tables/jquery.dataTables.min.js"></script>
```

The loader is loaded once.

### Option jQuery getScript()

Alternative is to load the library in the <SCRIPT> section, via

```
$.getScript('https://www.gstatic.com/charts/loader.js',function (){...
```

```
<!-- load the visualisation API -->
<script type="text/javascript">
  // use this to load the api if not added to the ~domoticz/www/index.html page head section:
  $.getScript('https://www.gstatic.com/charts/loader.js',function (){
    google.charts.load('current', { packages: ['corechart'] });
    google.charts.setOnLoadCallback(drawChart);
  });
}
```

The loader is loaded every time the script is called..

**This option is being used in the examples.**

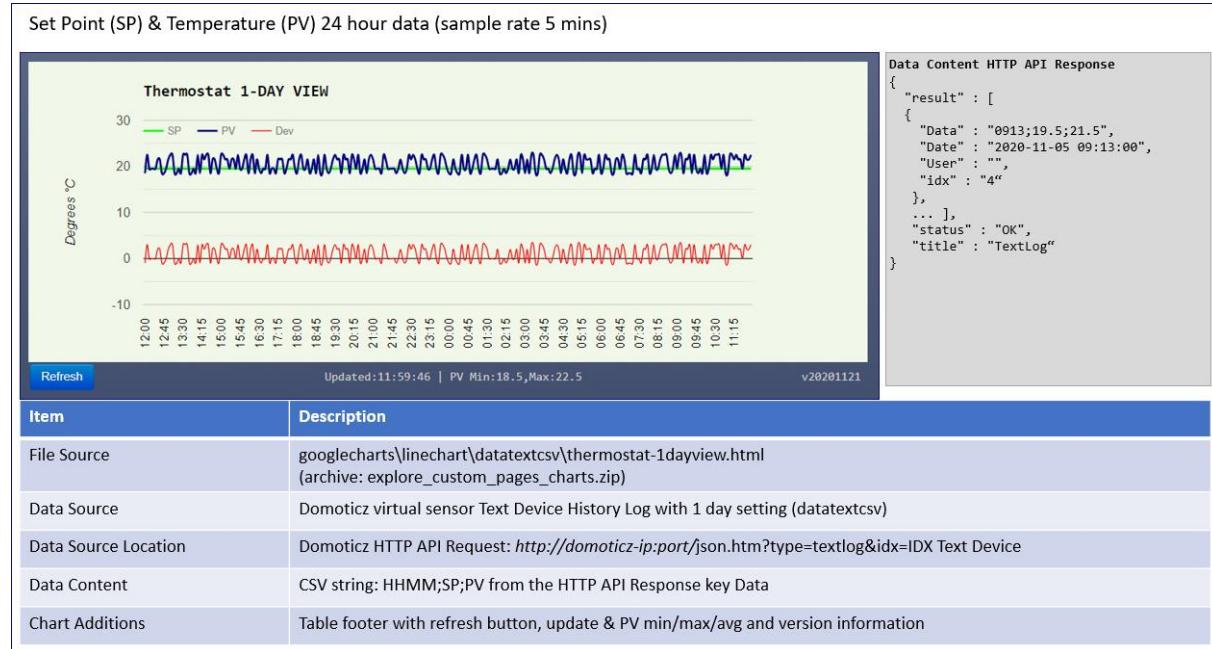
Recommend reading the Google Charts [FAQ](#) esp. the questions:

- Can I use charts offline?
- Can I download and host the chart code locally, or on an intranet?

Ensure to comply to the Google Chart Terms.

## Thermostat Log

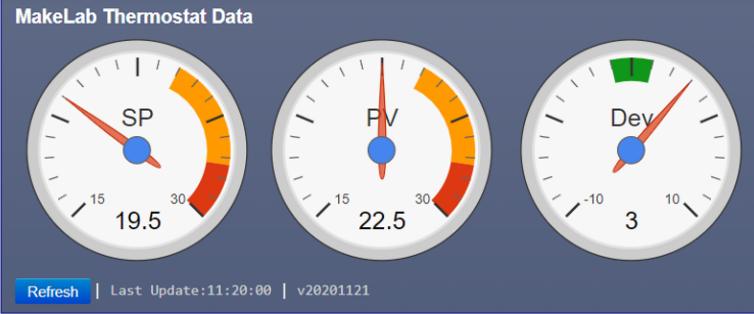
To display history log data in a **Line Chart** with 24-hour data (sampling every 5 minutes) for two devices Thermostat Temperature Setpoint (SP) and Room Temperature Process Value (PV). The deviation between PV and SP is calculated and displayed as a line.



## Thermostat Dashboard

To display several device data in a **Gauges Chart**.

Set Point (SP), Temperature (PV) & Deviation (Dev)



**MakeLab Thermostat Data**

SP: 19.5 | PV: 22.5 | Dev: 3

Refresh | Last Update: 11:20:00 | v20201121

```
Data Content HTTP API Response
{
  "app_version": "2020.2 (build 12554)",
  "result": [
    {
      ...
      "Data": "0913;21.0;24.0",
      "Name": "MakeLab Thermostat Data",
      ...
      "idx": "42"
    }
  ],
  "status": "OK",
  "title": "Devices"
}
```

Item	Description
File Source	googlecharts\gaugeschart\datahttpid\thermostat_dashboard.html (archive: explore_custom_pages_charts.zip)
Data Source	Domoticz virtual sensor Text Device Status (datahttpid)
Data Source Location	Domoticz HTTP API Request: <a href="http://domoticz-ip:port/json.htm?type=devices&amp;rid=IDX">http://domoticz-ip:port/json.htm?type=devices&amp;rid=IDX</a> Text Device
Data Content	CSV string: HHMM;SP;PV from the HTTP API Response key Data
Chart Additions	Chart device name; Table footer with refresh button, update & PV min/max/avg and version information

### Notes

The chart is refreshed every 5 minutes but if want to use for real time, then change value to 1 minute (the dzVents data generator timer must be set to 1 minute as well) or even higher if the data is updated more frequent than 1 minute.

The gauges are assigned to HTML selectors in div tag blocks. To ensure the gauges are on a single line and are also responsive, the div style is set to inline-block in the HTML section.

```
<div class="chart noselect">
  <div class="chartarea">
    <h3 id="devicename"></h3>
    <div id="gaugeschart" class="device-log-chart_md" style="display: inline-block;"></div>
    <div id="gaugesdevchart" class="device-log-chart_md" style="display: inline-block;"></div>
    <br/>
    <div style="display: inline-block; text-align: left;">
      <button class="btn btn-small btn-primary" style="float:none!important;display:inline;" onclick="refreshChart();">Refresh</button>
      | <span id="chartinfo" class="infotext"></span>
      | <span id="pageversion" class="infotext"></span>
    </div>
    <!-- Chart footer with device attribute lastupdate and the pageversion
    <table style="width:100%;">
      <tr>
        <td style="width:10%;" align="left"><button class="btn btn-small btn-primary" style="float:none!important;display:inline;" onclick="refreshChart();">Refresh</button></td>
        <td style="width:80%;"><div id="chartinfo" class="infotext" style="width:100%;text-align:center"></div></td>
        <td style="width:10%;"><div id="pageversion" class="infotext" style="text-align:right"></div></td>
      </tr>
    </table>
    -->
  </div>
</div>
```

The div with id “gaugeschart” shows the gauges for SP and PV with scale 15 to 30 Degrees C.

The div with id “gaugesdevchart” shows the gauge for Dev with scale -10 to 10 Degrees C. The difference in scale and the colouring is the reason why two different div’s are used.

```
var gaugeschart = new google.visualization.Gauge(document.getElementById('gaugeschart'));
gaugeschart.draw(gaugesdata, gaugesoptions);

var gaugesdevchart = new google.visualization.Gauge(document.getElementById('gaugesdevchart'));
gaugesdevchart.draw(gaugesdevdata, gaugesdevoptions);
```

# Installation

The custom pages are stored in folder

```
~domoticz/www/templates
```

## Custom Page

1. From the archive *explore\_custom\_pages\_charts.zip*, extract the selected file (i.e. *Highcharts/linechart/datagetcsv/thermostat-1dayview.html*) and copy the file to the Domoticz www/templates folder.
2. Modify the HTML file JavaScript section:  
Device IDX and, if required, the URL for the HTTP API request.
3. Ensure the Domoticz GUI Custom Menu is enabled in the settings:  
GUI > Setup > Settings > System > Active Menus > Tick Custom
4. Clear the browser cache.
5. Refresh Domoticz GUI.
6. Check the Domoticz GUI > Tab Custom if the file name appears in menu dropdown.
7. Select the menu item.
8. Press F12 to check the browser console log for errors (Debug window).

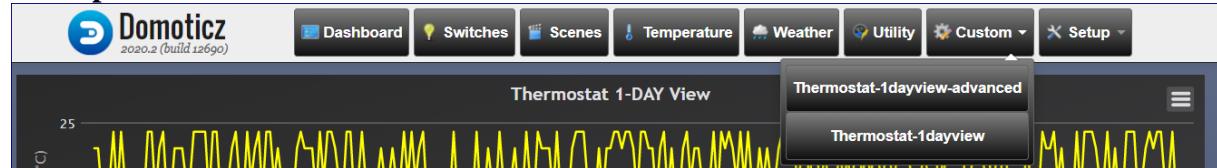
## Automation Event

If using the 1-DAY VIEW chart, the Domoticz Text Device & the Automation event dzVents *thermostat\_data\_textcsv.dzvents* must be created.

Create an empty Automation Event dzVents:

- GUI > Setup > More Options > Event > + > dzVents > Minimal.
- Clear the content.
- Copy & paste the content of the file *thermostat\_data\_textcsv.dzvents*.
- Amend the IDX of the devices used and change the call to updateText with the devices.
- Set the event to On.
- Save the event.
- Check the Domoticz Log (GUI > Setup > Log).

## Example Custom Menu



## Data Sources

As mentioned, the data is generated in regular intervals by Automation Events dzVents Lua except for the HTTP API requests.

### DataTextCSV

Chart data from a Domoticz text device holding text in CSV format (separated by semicolon). The text device value has a maximum of 200 characters – in case more data required additional text devices are required.

```
HHMM;Data1;DataN
```

#### HTTP Request

```
http://domoticz-ip:port/json.htm?type=textlog&idx=42
```

#### HTTP Response (Example Snippet)

```
{
  "result" :
  [
    {"Data" : "0913;18.5;21.5", "Date" : "2020-11-05 09:13:00", "User" : "", "idx" : "4"},
    ...
  ],
  "status" : "OK", "title" : "TextLog"
}
```

## DataTextJSON

Chart data from a Domoticz text device holding text in JSON format.

The text device value has a maximum of 200 characters – in case more data required additional text devices are required.

The history data is retrieved from a text device holding the devices data as JSON.

#### HTTP Request

```
http://domoticz-ip:port/json.htm?type=textlog&idx=43
```

#### HTTP Response (Example Snippet)

```
{
  "result" :
  [
    {"Data" : "SEE BELOW", "Date" : "2020-11-13 18:00:00", "User" : "", "idx" : "1999"},
    ...
  ],
  "status" : "OK", "title" : "TextLog"
}
```

The Data key is a JSON string:

```
"Data" : "
{
  "timestamp": "1800",
  "devices": [
    {"name": "MakeLab Thermostat Setpoint", "data": 17.5},
    {"name": "MakeLab Temperature", "data": 16.5}
  ]
}"
```

The timestamp is used for x-axis labels.

The devices array for the series (i.e. 2) with data.

#### *Note*

This sample has already 130 chars out of 200 max.

For this sample could limit to 103 charts by using shorter names:

```
{"t":"1612","d": [{"n":"MakeLab Thermostat Setpoint", "d":17.5}, {"n":"MakeLab Temperature", "d":16.5}]}]
```

## DataJSONFile

Chart data from a local file holding a JSON object.

The data is retrieved from a JSON file located in the www\templates folder.

```
~domoticz/www/templates/thermostat_1dayview_data.json
```

The Data file format:

JSON object with key result holding an array of sample data.

#### **Example**

Two devices Thermostat & Temperature.

Each sample has the keys:

time (HH:MM), sp (Thermostat Setpoint C), pv (Thermostat Process Value Temperature C)

```
{
  "result" :
  [
    {"pv":22.0,"sp":22.0,"time":"11:17"}, {"pv":25.0,"sp":22.0,"time":"11:16"}, ...
  ],
  "status":"OK","title":"Thermostat Data"
}
```

## DataHTTPPlan

Chart data from the response of a HTTP API request for devices data from a plan, i.e. room plan.

The data is retrieved from selective temperature devices assigned to a room plan.

#### **HTTP Request**

```
json.htm?type=devices&plan=23
```

## HTTP Response (Example Snippet)

```
{
  "ServerTime" : "2020-11-10 09:53:16",
  "result" :
  [
    {"Name" : "Aussen", "Temp" : 6.7, "idx" : "344", },
    {"Name" : "Wohnen", "Temp" : 19.4, "idx" : "350", },
    {"Name" : "Server Raum", "Temp" : 18.8, "idx" : "351", },
    {"Name" : "Keller Gro\u000fdraum", "Temp" : 16.2, "idx" : "356", },
  ],
  "status" : "OK","title" : "Devices"
}
```

*Note*

If the data to be retrieved is running on the same system as the custom page, the HTTP Request URL for the Ajax URL is

```
json.htm?type...
```

Else the full URL is required

```
http://domoticz-ip:port/json.htm?type...
```

# Custom UI

## Purpose

To explore how to build a customized web based user interface with the goal to develop the function [Web UI Quick Access Mobile](#) running on the Domoticz Prouction system.

The next example is a starter and more to follow.

## Approach

For exploring, a simulator (dzVents running on the Domoticz Development system) is developed to get or set Domoticz device data.

Simulated are

- Room Temperature with a Thermostat Setpoint & Temperature device
- Thermostat battery level with a percentage device

The devices are assigned to a room plan. The room is called MakeLab.

An Automation Event dzVents “custom\_ui\_simulator.dzvents” is running every minute to randomly set values for the temperature and battery level.

If the thermostat setpoint is changed, the random generated temperature is also changing around the new setpoint.

The custom UI is built step-by-step and the HTML filename is called **index.html**.

The target is to remote access the index.html running on the Domoticz system:

```
http://domoticz-ip:8080/cui/index.html
```

with external libraries/style dependencies stored locally on the Domoticz system in a dedicated folder **cui**.

This access option is described: [Remote Access with Local Libraries \(PREFERRED\)](#).

The soure of the simulator see: [Appendix: Automation Event Simulator](#).

# Domoticz Configuration

## Devices

The Domoticz devices are created by using the Virtual Sensors Hardware:

Idx	Hardware	ID	Unit	Name	Type	SubType	
32	VirtualSensors	00082032	1	MakeLab Thermostat Battery	General	Percentage	0%
31	VirtualSensors	1406F	1	MakeLab Temperature	Temp	LaCrosse TX3	0.0 C
30	VirtualSensors	001406E	1	MakeLab Thermostat Setpoint	Thermostat	SetPoint	20.5

## Roomplan

The roomplan, named MakeLab with IDX=3, lists the 3 devices with IDX=30,31,32.

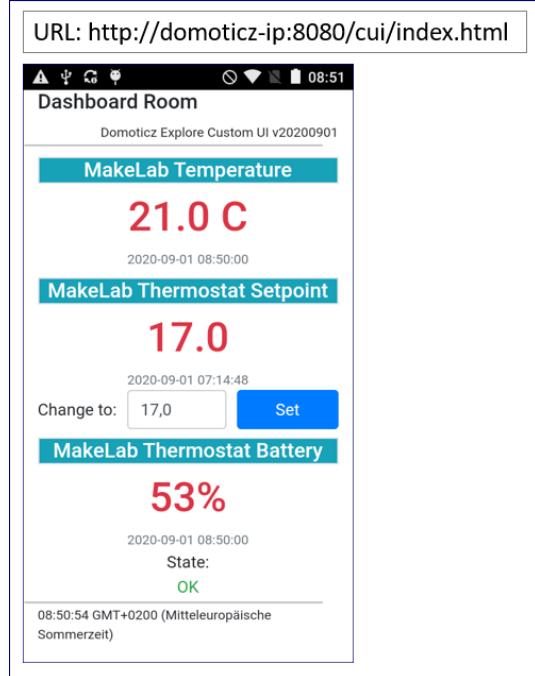
Idx	Hardware
3	MakeLab

Showing 1 to 1 of 1 entries (filtered from 3)

Idx	Hardware
31	MakeLab Temperature
30	MakeLab Thermostat Setpoint
32	MakeLab Thermostat Battery

## Custom UI HTML File

### Screenshot Mobile Device - Android Smartphone



The solution is based on [Remote Access with Local Libraries](#).

The custom UI index file is **index.html** with the libraries/styles:

- **jQuery** - Event handling (JavaScript library)
- **Bootstrap** - Responsive mobile-first UI (CSS, JavaScript library)

### Installation

Create on the Raspberry Pi a folder

```
/home/pi/domoticz/www/cui.
```

Extract the archive **cui.zip** to the created folder. The folder structure is:

```
~domoticz/www/cui/index.html
~domoticz/www/cui/css/bootstrap.css
~domoticz/www/cui/js/bootstrap.js
~domoticz/www/cui/js/jquery.js
```

In addition, there is the device simulator dzVents script **custom\_ui\_simulator.dzvents**.

Create a new event from type dzVents, copy & paste the content of the simulator file and save.



The custom\_ui\_simulator.dzvents file is just for saving and not needed as stored in the Domoticz database.

If not done already, create the 3 devices as mentioned earlier and set the idx of the devices accordingly in the simulator event.

The custom UI dashboard can be accessed from a Browser:

```
http://domoticz-ip:8080/cui/index.html
```

## HTML Index File

The index file has 3 sections:

### HEAD

To set the META tags viewport, the includes for Bootstrap and jQuery and additional CSS styles.

### HTML

To define the Bootstrap grid with

- selectors being set with data from Domoticz HTTP API requests (type=devices)
- input field and button to set data by Domoticz HTTP API requests (type=command)
- GET = Display the data from 3 devices: Temperature, Thermostat Setpoint, Thermostat Battery Level.
- SET = Set the Thermostat Setpoint

### JavaScript

To handle HTTP API requests to SET (Button click) or GET (refresh interval every 60 seconds) data

## Examples GET & SET

**HTTP API GET** request URL to get all devices data in JSON format with key/value pairs:

```
http://domoticz-ip:port/json.htm?type=devices&used=true&displayhidden=0
```

From the HTTP API response JSON array, data for 3 devices is selected using the device idx.

```
{
  ...
  "app_version" : "2020.2 (build 12283)",
  "result" :
  [
    {
      ...
      "Data" : "17.0",
      "LastUpdate" : "2020-09-01 07:14:48",
      "Name" : "MakeLab Thermostat Setpoint",
      "idx" : "30"
    },
    {
      ...
      "Data" : "22.0 C",
      "LastUpdate" : "2020-09-01 09:28:00",
      "Name" : "MakeLab Temperature",
      "trend" : 3
    },
    {
      ...
    }
  ]
}
```

```

        "Data" : "34%",
        "LastUpdate" : "2020-09-01 09:28:00",
        "idx" : "32"
    }
],
"status" : "OK",
"title" : "Devices"
}

```

**HTTP API SET** request URL to set the setpoint, in Celsius, of the thermostat with IDX=30

```
http://domoticz-ip:port/json.htm?type=command&param=setsetpoint&idx=30&setpoint=20.5
```

Prior testing the webpage, submit the URL in a browser and check response:

```
{"status" : "OK","title" : "SetSetpoint"}
```

The keys (case sensitive) used are: idx, Name, Data, LastUpdate.

The jQuery library is used as required for getting the data from the Domoticz server.

The device data is displayed in a bootstrap grid.

In the browser, use F12 to view the console for any errors.

## Source

```

<!DOCTYPE html>
<html lang="en">
<!--
domoticz-home-automation-workbook - explore_custom_ui
File: index.html
Libraries: jQuery, Bootstrap; included via local files stored in subfolders css & js
Folders: ~domoticz/www/cui/; ~domoticz/www/cui/css, ~domoticz/www/cui/js
Browser Access: http://domoticz-ip:port/cui/index.html
The access focus is on using the custom ui on small devices, i.e. smartphones.
Data:
GET = Display the data from 3 devices: Temperature, Thermostat Setpoint, Thermostat Battery Level.
SET = Set the Thermostat Setpoint
The Domoticz API GET request parsed by JavaScript and refreshed every 60 seconds.

Example HTTP API GET request URL to get all devices data in JSON format with key/value pairs:
http://domoticz-ip:port/json.htm?type=devices&used=true&displayhidden=0
From the HTTP API response JSON array, the data for 3 devices is selected using the device idx.

Example HTTP API SET request URL to set the setpoint, in Celsius, of the thermostat with idx=30
http://domoticz-ip:port/json.htm?type=command&param=setsetpoint&idx=30&setpoint=20.5

Note: Prior testing the webpage, submit the URL in a browser and check response:
{"status" : "OK","title" : "SetSetpoint"} OR

The keys (case sensitive) used are: idx, Name, Data, LastUpdate.
The jQuery library is used as required for getting the data from the Domoticz server.
The device data is displayed in a bootstrap grid.
In the browser, use F12 to view the console for any errors.
20200901 by RwbL
-->
<head>
<title>Domoticz Custom UI Explore - Dashboard Room MakeLab</title>
<meta charset="utf-8">
<!-- Ensure proper rendering and touch zooming -->
<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- INCLUDES = REMOTE but with LOCAL Bootstrap & jQuery -->
<link rel="stylesheet" href="css/bootstrap.css">
<script src="js/jquery.js"></script>
<script src="js/bootstrap.js"></script>

<!-- Specific Styles for the device data keys Name, Data, LastUpdate-->
<style type="text/css">
/* The classes for the device keys are just an example but not used as replaced by bootstrap hX */
```

```

/* Class name used for device key "Name" displayed in a table cell */
.name { color: black; font-size: 16px; font-family: "Lucida Console", Courier, monospace; }
/* Class data used for device key "Data" displayed in a table cell */
.data { color: red; font-size: 48px; font-family: "Lucida Console", Courier, monospace; }
/* Class lastupdate used for device key "LastUpdate" displayed in a table cell */
.lastupdate { color: gray; font-size: 12px; font-weight: normal; font-family: "Lucida Console", Courier, monospace; }
/* HR */
.hrdv { min-width:90%; height:1px; margin:2px !important; background-color: lightgray; }
</style>
</head>

<body>
<!-- HTML -->
<!-- Full width container, spanning entire width of the viewport -->
<div class="container-fluid">
<!-- Stack the columns on mobile half-width and desktop half-width -->
<div class="row">
<div class="col-sm-4 col-md-4"><p class="h5">Dashboard Room</p></div>
<div class="col-sm-4 col-md-6 text-right small">Domoticz Explore Custom UI v20200901</div>
<hr class="hrdv" />
</div>

<!-- Stack the columns on mobile full-width and desktop quarter-width -->
<div class="row">
<div id="temperature" class="col-12 col-md-4"></div>

<div class="col-sm-10 col-md-4">
<div id="setpoint"></div>
<form id="formsetpoint">
<div class="form-row">
<div class="col-auto">
<label for="setpointvalue" class="col-form-label text-right">Change to:</label>
</div>
<div class="col-auto">
<input id="setpointvalue" type="number" class="form-control" min="0" max="23" step="0.5">
</div>
<div class="col">
<button id="btnsetsetpoint" type="button" class="btn btn-primary btn-block">Set</button>
</div>
</div>
</form>
</div>

<div class="col-sm-10 col-md-4">
<div id="battery"></div>
<div id="batterystate" class="text-center"></div>
</div>
</div>

<!-- Stack the columns on mobile full-width and desktop full-width -->
<div class="row">
<hr class="hrdv" />
<div id="datetime" class="col-12 small"></div>
</div>

</div>

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">
const URL_DOMOTICZ = "http://domoticz-ip:port/json.htm?";
var URL_DOMOTICZ_DEVICES = URL_DOMOTICZ + "type=devices&used=true&displayhidden=0";
var URL_DOMOTICZ_SETSETPOINT = URL_DOMOTICZ +
"type=command&param=setsetpoint&idx=#IDX#&setpoint=#SP#"
const REFRESHINTERVAL = 10000;           // ms, 60s
// Define the device idx
const IDX_THERMOSTAT_SETPOINT = 30;
const IDX_THERMOSTAT_BATTERY = 32;
const IDX_TEMPERATURE = 31;
const TH_BATTERYLEVEL = 50;

/*
 Set the device item in a grid col.
 The items displayed are Name, Data, LastUpdate
 Example: setDevice(item, "setpoint")

```

```

/*
function setDevice(item, selector)
{
    console.log("Set Device: " + item.Name);
    document.getElementById(selector).innerHTML =
        '<div class="h5 text-center border mt-2 bg-info text-white">' + item.Name + '</div>' +
        '<div class="h1 text-danger text-center">' + item.Data + '</div>' +
        '<div class="small text-center text-secondary">' + item.LastUpdate + '</div>';
}

/*
Set a new setpoint.
Example: setSetPoint(URL_DOMOTICZ_SETSETPOINT, IDX_THERMOSTAT_SETPOINT, "setpointvalue")
*/
function setSetPoint(requestUrl, deviceIdx, inputSelector)
{
    // Get the new setpoint from the input field defined by the inputSelector
    newSetpoint = document.getElementById(inputSelector).value;
    // Build the HTTP API request url to set the new setpoint
    requestUrl = requestUrl.replace("#IDX#", deviceIdx).replace("#SP#", newSetpoint);
    // Set Setpoint: http://domoticz-
    ip:port/json.htm?type=command&param=setsetpoint&idx=30&setpoint=19.5
    console.log("Set Setpoint: " + requestUrl);
    $.getJSON(
        requestUrl,
        {
            format: "json"
        },
        function(data)
    {
        // Expect JSON result: {"status" : "OK","title" : "SetSetpoint"}
        if (typeof data != 'undefined')
        {
            // Set Setpoint Status: OK (SetSetpoint)
            console.log("Set Setpoint Status: " + data.status + " (" + data.title + ")");
            refreshData();
        }
    });
}

/*
Refresh domoticz sensor data for the devices set by the request url
*/
function refreshData()
{
    // Clear the refresh timer interval first
    clearInterval($.refreshTimer);

    // Get current date&time and assign to the selector
    var d = new Date();
    $('#datetimestamp').html(d.toTimeString());

    // Define the HTTP API request url
    var requestUrl = $.urlDomoticz;

    // Request all device data, loop over the HTTP response to get the data for the selected devices.
    $.getJSON(
        requestUrl,
        {
            format: "json"
        },
        function(data)
    {
        if (typeof data.result != 'undefined')
        {
            // Loop over the result array and get the items
            $.each(data.result, function(i,item)
            {
                if (item.idx == IDX_TEMPERATURE) {
                    setDevice(item, "temperature")
                }
                if (item.idx == IDX_THERMOSTAT_SETPOINT) {
                    setDevice(item, "setpoint")
                    // Set the input field setpointvalue if there is no value, i.e. first time
                }
            });
        }
    });
}

```

```
        if (document.getElementById("setpointvalue").value.length == 0) {
            document.getElementById("setpointvalue").value = item.Data;
        }
    }
    if (item.idx == IDX_THERMOSTAT_BATTERY) {
        setDevice(item, "battery")
        // Check if level to set the state OK or LOW
        batterylevel = item.Data.replace("%","");
        batterystate = batterylevel > TH_BATTERYLEVEL ? '<div class="text-success">OK</div>' :
'<div class="text-danger font-weight-bold">LOW</div>';
        document.getElementById("batterystate").innerHTML = '<div class="text-center">State: ' + batterystate + '</div>';
    }
}
);
// Start the refresh timer
$.refreshTimer = setInterval(refreshData, $.refreshInterval);
}

// Init the document with request url, roomplan and energydata info
$(document).ready(function() {
    $.urlDomoticz = URL_DOMOTICZ_DEVICES;
    $.refreshInterval = REFRESHINTERVAL;
    refreshData();

    // Disable enter key on form to set the setpoint
    $("#formsetpoint").keypress(function(e){
        if(e.keyCode == 13) {
            e.preventDefault();
            return false;
        }
    })
}

// assign the click function to the button with id=btnsetsetpoint
$("#btngensetsetpoint").click(function(){
    setSetPoint(URL_DOMOTICZ_SETSETPOINT, IDX_THERMOSTAT_SETPOINT, "setpointvalue")
});
});

</script>

</body>
</html>
```

## Appendix: Solution Options Libraries

### Local Access with External Libraries

The custom UI is built locally (any device) and accessed from the desktop by opening the HTML file *index.html* stored in a dedicated folder, i.e. *cui*.

The required external libraries jQuery & Bootstrap are included via a Content Delivery Network (CDN).

```
<!-- REMOTE CDN Bootstrap & jQuery -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

### Local Access with Local Libraries

Like previous, but the jQuery & Bootstrap libraries are stored also locally in the subfolders *css* & *js* of the dedicated folder, i.e. *cui*, *cui/css* and *cui/js*

Access from the desktop by opening the HTML file **index.html**.

The required external libraries can be downloaded from the respective jQuery and Bootstrap websites download section.

The JQuery library *jquery.min.js* is copied to the subfolder *js* with filename *jquery.js*.

There are 3 files used *bootstrap.css*, *bootstrap.js* and *jquery.js*.

```
<!-- LOCAL Bootstrap & jQuery -->
<link rel="stylesheet" href="css/bootstrap.css">
<script src="js/jquery.js"></script>
<script src="js/bootstrap.js"></script>
```

### Remote Access with Local Libraries

The dashboard is accessed remote via browser.

The dashboard file & required libraries are stored in a dedicated subfolder (f.e. *cui*) on the Domoticz www folder, i.e. */home/pi/domoticz/www/cui*.

Open the dashboard in the browser with URL <http://domoticz-ip:8080/cui/index.html>.

The advantage of a dedicated folder, is that the folder can contain additional files used, like images, modules, stylesheets and other.

The required external libraries can be downloaded from the respective jQuery and Bootstrap websites download section.

The JQuery library *jquery.min.js* is copied to the subfolder *js* with filename *jquery.js*.

There are 3 files used *bootstrap.css*, *bootstrap.js* and *jquery.js*.

```
<!-- Bootstrap & jQuery stored in subfolders css&js -->
<link rel="stylesheet" href="css/bootstrap.css">
<script src="js/jquery.js"></script>
<script src="js/bootstrap.js"></script>
```

### Remote Access with Domoticz Local Libraries

As previous, but the external libraries used are the ones from the Domoticz installation located in the www subfolders *css* & *js*.

The dashboard file & required libraries are stored in a dedicated subfolder (f.e. cui) on the Domoticz www folder, i.e. /home/pi/domoticz/www/cui.

Open the dashboard in the browser with URL <http://domoticz-ip:8080/cui/index.html>.

```
<!-- Bootstrap & jQuery used from the Domoticz subfolders ../css & ../js -->
<link rel="stylesheet" href="../css/bootstrap.css">
<script src="../js/jquery-3.4.1.min.js"></script>
<script src="../js/bootstrap.min.js"></script>
```

### Notes

The Domoticz installation uses some older versions of the external libraries and additional stylesheets overriding some styles.

This results in different behaviour of the custom UI esp. when using the Bootstrap grid component.

Checkout if the versions will change upon further development or Domoticz updates (Status: 2020.2 Build 12283).

## Appendix: Automation Event Simulator

```
--[[[custom_ui_simulator.dzvents
Trigger: timer, device change
To simulate a room (called MakeLab) temperature controlled by a thermostat device.
The temperature and the battery level are set using random values.
Devices
(idx,hardware,name,type,subtype,switchtype,data)
32, VirtualSensors, MakeLab Thermostat Battery, General, Percentage, , 0%
31, VirtualSensors, MakeLab Temperature, Temp, LaCrosse TX3, , 0.0 C
30, VirtualSensors, MakeLab Thermostat Setpoint, Thermostat, SetPoint, , 20.5
20200828 rwbl
]]-- 

-- device idx
local IDX_THERMOSTAT_SETPOINT = 30
local IDX_THERMOSTAT_BATTERY = 32
local IDX_TEMPERATURE = 31

local TH_THERMOSTAT_BATTERY_LOW = 50

return {
    on = {
        timer = {'every minute'},
        devices = {IDX_THERMOSTAT_SETPOINT, IDX_THERMOSTAT_BATTERY, IDX_TEMPERATURE},
    },
    execute = function(domoticz, item)
        -- define locals and assign device values
        local setpoint = domoticz.devices(IDX_THERMOSTAT_SETPOINT).setPoint
        local batterylevel = domoticz.devices(IDX_THERMOSTAT_BATTERY).percentage
        local temperature = domoticz.devices(IDX_TEMPERATURE).temperature
        -- handle initial temperature
        if temperature == 0 then temperature = setpoint end

        -- check the timer
        if item.isTimer then
            domoticz.log(string.format("Timer trigger: %s", item.trigger), domoticz.LOG_INFO)
            -- run randomseed with clock parameter prior random used for the new temperature & battery
            level
            math.randomseed(os.clock()*10000000000)

            -- update the temperature around 5°C plus/minus around the setpoint
            -- math.random with no arguments generates a real number between 0 and 1
            local value = math.random(0,5)
            -- set negative value
            if value < 3 then value = value * -1 end
            -- set new temperature
        end
    end
}
```

```

temperature = setpoint + value
domoticz.devices(IDX_TEMPERATURE).updateTemperature(temperature)
domoticz.log(string.format("New Temperature: %.1f", temperature), domoticz.LOG_INFO)

-- change the battery percentage with random value 0 - 100
batterylevel = math.random(0,100)
domoticz.devices(IDX_THERMOSTAT_BATTERY).updatePercentage(batterylevel)
domoticz.log(string.format("New Battery Level: %d", batterylevel), domoticz.LOG_INFO)
end

-- check device changes
if item.isDevice then
    domoticz.log(string.format("Device changed: %s; nValue=%d, sValue=%s",
        domoticz.devices(item.id).name,
        domoticz.devices(item.id).nValue,
        domoticz.devices(item.id).sValue), domoticz.LOG_INFO)
    -- device battery
    if domoticz.devices(item.id).id == IDX_THERMOSTAT_BATTERY then
        domoticz.log(string.format("Battery Level: %d", batterylevel), domoticz.LOG_INFO)
        if batterylevel <= TH_THERMOSTAT_BATTERY_LOW then
            domoticz.log(string.format("Battery Low: %d (%d)", batterylevel,
TH_THERMOSTAT_BATTERY_LOW), domoticz.LOG_ERROR)
        end
    end
    -- device thermostat setpoint
    if domoticz.devices(item.id).id == IDX_THERMOSTAT_SETPOINT then
        domoticz.log(string.format("Setpoint: %.1f", setpoint), domoticz.LOG_INFO)
    end
    -- device temperature
    if domoticz.devices(item.id).id == IDX_TEMPERATURE then
        domoticz.log(string.format("Temperature: %.1f", temperature), domoticz.LOG_INFO)
    end
end
end
}

```

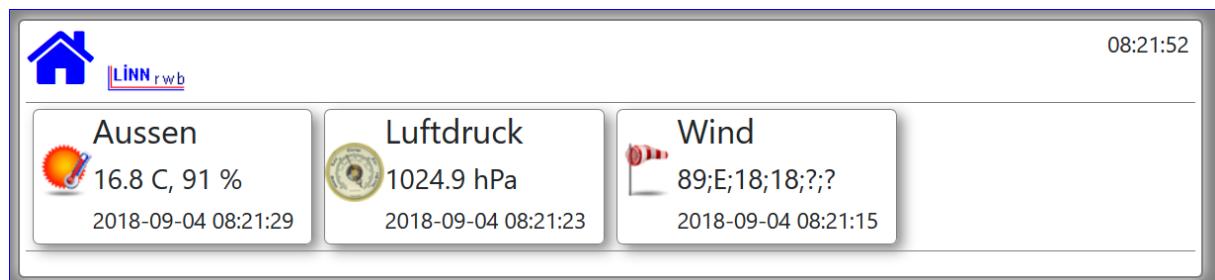
## Appendix: Older Solution

This was the first tryout, to display weather data from three devices and update every 5 seconds. A clock is running every second.

This solution is just kept for reference.

### Domoticz Devices

Temperature & Humidity (Name="Aussen", idx=14, Hardware=RFXtrx433e, Type=Temp + Humidity, SubType=Cresta, TFA TS34C)
Luftdruck (Name="Luftdruck", idx=116, Hardware=VirtualSensors, Type=General, SubType=Barometer)
Wind (Name="Wind", idx=117, Hardware=VirtualSensors, Type=Wind, SubType=WTGR800)





## Concept

The device data is requested (URL) via the Domoticz engine using JSON.  
The Domoticz engine allows to interact with all devices using JSON (nice solution).

The UI uses jQuery, Bootstrap and Domoticz icons.

The data for the favourite (=Dashboard) devices is requested from the Domoticz system via URL.

```
URL: domoticzurl:8080/json.htm?type=devices&used=true&filter=all&favorite=1
```

From the request result (JSON string), selective data for the devices with IDX=14,116,117 is obtained.

### JSON Result Snippet

The URL returns the data in JSON format.

This snippet shows some of the properties for the devices.  
The key `result[]` is an array with the device information.

In this example the results array (made bold) has a length of 3 for the devices with idx,14,116,117. For each of the devices, the Keys “Data” and “Idx” are used.

Using JavaScript, the result array is parsed (see below).

```
{
  "ActTime" : 1536051617,
  ...
  "app_version" : "4.9980",
  "result" : [
    {
      "Data" : "20.9 C, 79 %", "idx" : "14"
    },
    {
      "Data" : "1025 hPa", "idx" : "116"
    },
    {
      "Data" : "89;E;14;21;?;?", "idx" : "117"
    }
  ],
  "status" : "OK",
  "title" : "Devices"
}
```

### JavaScript Snippet

```
// Define the idx to of the devices display
var arrayIdx = [14,116,117];
url = domoticzURL + "json.htm?type=devices&used=true&filter=all&favorite=1"
$.getJSON(url, function(result){
  var idx = 0;
  var sensorInfo = "";
  for (i = 0; i < result.result.length; i++) {
    idx = parseInt(result.result[i].idx);
    if (arrayIdx.indexOf(idx) > -1) {
      sensorInfo += setSensorInfo(result,i);
      console.log("Found idx=" + idx);
    }
  }
  ...
})
```

{}

## HTML & JavaScript

```

<!DOCTYPE html>
<html>
<head>
    <title>Weather Frontend</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8"/>
    <!-- Reference to the external CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
    <!-- Reference to the external JS -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
    <!-- Styling -->
    <style>
        body{background-color: grey; padding: 0px 1% 0% 1%;}

        .container{padding: 0px 20px 20px; border-radius: 5px; background-color: white; border-style: solid; border-color: grey; border-width: 2px; box-shadow: 0px 10px white; margin-top: 2%}

        .row {border-bottom-style: solid; border-color: grey; border-width: 1px; }

        #linn_logo {max-width: 100px; margin: 5px; }

        #top-bar{text-align: left; margin-bottom: 1px; padding: 0px; }

        #clock{float: right; margin: 5px; }

        .v-align {float: none; display: inline-block; vertical-align: middle;} /*class for vertically aligning elements in a row*/
        #house-image{font-size: 9em; vertical-align: middle; display: inline-block; margin-top: 10px; margin-bottom: 10px; }

        #house-status-text{text-align: left; }

        #outside-temperature{text-align: left; }

        #outside-humidity{text-align: left; }

        #outside-humiditystatus{text-align: left; }

        #outside-lastupdate{text-align: left; }

        .sensor-outer-box {padding: 0px; }

        .sensor-box{padding: 0px; margin: 5px; background-color: white; box-shadow: 5px 10px #888888; border-style: solid; border-color: grey; border-width: 1px; border-radius: 5px; }

        .sensor-icon{vertical-align: 80%; margin: 0px; }

        .sensor-data {display: inline-block; margin: 0px; }

    </style>
</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-sm-12" id="top-bar">
                <i class="fa fa-home" style="font-size:60px; color:blue;"></i>
                
                <div id="clock">
                </div>
            </div>
        </div>
        <div class="row" id="sensor-container">
        </div>
    </div>

    <script>
        $(document).ready(function(){
            var domoticzURL = "http://vz-ip:8080/";
            //var domoticzURL = "http://localhost:8080/";
            // Define the idx to of the devices display
            var arrayIdx = [14,116,117];
            // Timer interval polling data (ms)
            var intervalTimer = 5000;

            // Update the sensorinfo
            $(function() {
                updateClock();
                updateSensors();
            });

            // Returns the image of the sensor that is stored in the image folder on the Domoticz system
            // /home/pi/domoticz/www/images
        });
    </script>
</body>

```

```

function getSensorImage(TypeImg){
    var sensorURL = "";
    switch(TypeImg) {
        case "dimmer":
            return domoticzURL + "images/Light48_On.png";
            break;
        case "temperature":
            return domoticzURL + "images/temp-20-25.png";
            break;
        case "wind":
            return domoticzURL + "images/wind48.png";
            break;
        case "gauge":
            return domoticzURL + "images/baro48.png";
            break;
        default:
            return domoticzURL + "images/Light48on.png";
    }
}

function setSensorInfo(result,i){
    var sensorInfo = "";
    sensorInfo += ''+
        '<div class="col-sm-6 col-md-4 col-lg-3 sensor-outer-box">' +
        '<div class="sensor-box">' +
        '  </img>' +
        '<div class="sensor-data">' +
        '  <h4>' + result.result[i].Name + '</h4>' +
        '  <h5>' + result.result[i].Data + '</h5>' +
        '  <h6>' + result.result[i].LastUpdate + '</h6>' +
        '</div>' +
        '</div>' +
        '</div>';
    return sensorInfo;
}

function updateClock() {
    var d = new Date();
    var hours = d.getHours();
    if (hours < 10) { hours = '0' + hours;}
    var minutes = d.getMinutes();
    if (minutes < 10) { minutes = '0' + minutes;}
    var seconds = d.getSeconds();
    if (seconds < 10) { seconds = '0' + seconds;}
    document.getElementById("clock").innerHTML = hours + ":" + minutes + ":" + seconds;
    setTimeout(updateClock, 1000);
}

function updateSensors() {
    // Get data for the favorite (=Dashboard) devices and select the idx=14,116,117
    // URL: domoticzurl:8080/json.htm?type=devices&used=true&filter=all&favorite=1
    url = domoticzURL + "json.htm?type=devices&used=true&filter=all&favorite=1"
    $.getJSON(url, function(result){
        var idx = 0;
        var sensorInfo = "";
        for (i = 0; i < result.result.length; i++) {
            idx = parseInt(result.result[i].idx);
            if (arrayIdx.indexOf(idx) > -1) {
                sensorInfo += setSensorInfo(result,i);
            }
        }
        document.getElementById("sensor-container").innerHTML = sensorInfo;
        //Finally set a timer to repeatedly poll the status of the sensors
        setTimeout(updateSensors,intervalTimer);
    });
}
</script>
</body>
</html>
```

# ESP8266

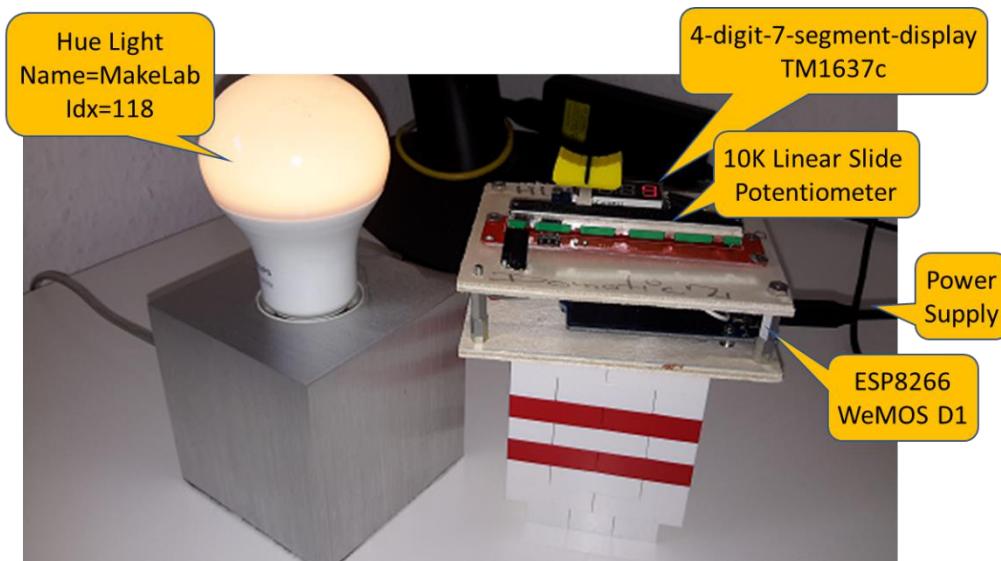
## Purpose

To explore programming an ESP8266 microcontroller in C++ using the Arduino IDE. It is basically a small fun project to tinker with microcontrollers and connected devices, but also an option to develop further integration of microcontrollers in the SMART Home Network.

## Experiment

Set the brightness of the Hue light (Name=MakeLab, IDX=118) connected to Domoticz, by using an ESP8266 microcontroller with a slide potentiometer (10K) and 4-digit-7-segment-display.

## Prototype



The prototype is assembled using wooden bottom plate to hold the microcontroller and a wooden top plate to hold the 4-digit-7-segment-display and the potentiometer slider. The plates are 8x10cm (3mm thick) and connected using 4 3x30mm screws.

## Prepare Arduino IDE

The Arduino IDE requires special ESP8266 setup:

- Download and install the [Arduino IDE](#).
- Start the Arduino IDE.
- Select menu File > Preferences.
- In the field “Additional Board Manager URLs”, set [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) followed by OK.
- Select menu Tools > Boards > Boards Manager...
- Enter esp8266 followed by selecting esp8266 by ESP8266 Community and press Install.
- Connect the ESP8266 board
- Select the board from Tools > Boards – in this case selected “WeMOS D1 R1”

- Install additional libraries required:  
ArduinoJson (used v6.10.0), TM1637 (used v1.2.0) – Thanks to the authors.

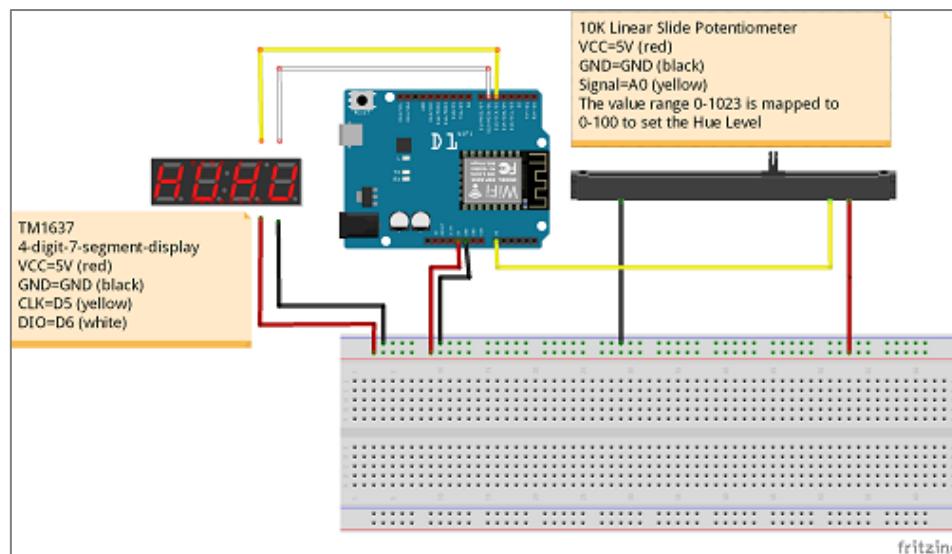
## Parts List

- 1x ESP8266 WeMOS D1 R1
- 1x 4-digit-7-segment-display TM1637
- 1x 10K Linear Slide Potentiometer
- 1x LED [on board]

## Wiring

Potentiometer	ESP8266 WeMOS D1
VCC	5V – power supplied via USB from PC
Signal	A0
GND	GND
4-digit-7-segment-display	ESP8266 WeMOS D1
VCC	5V
CLK	SLK/D5
DIO	MISO/D6
GND	GND
LED	ESP8266 WeMOS D1
Onboard LED used	

## Circuit



## Source Code

See file hueremotecontrol.ino.

### Pseudo Code

- Define constant pins & objects & strings ssid,password,http request url
- Setup
  - Init serial connection
  - Set onboard led mode
  - Setup tm1637 brightness & initial value
  - Setup wifi connection & hide the ssid & wait for connection
- Loop
  - Listen to potentiometer value changes
  - Check absolute value against noise level
  - Define & sent HTTP API Request with new value to Domoticz
  - Wait & handle Domoticz JSON response

### Domoticz Log

Changing the position of the slide potentiometer:

```
2019-04-13 10:44:34.895 (Philips Hue Bridge 1) Light/Switch (Hue MakeLab)
2019-04-13 10:44:34.868 Status: User: Admin initiated a switch command (118/Hue MakeLab/Set Level)
```

# ESP Easy

## Purpose

To explore connecting an ESP (i.e. ESP8266, ESP32 etc.) type microcontroller (like the NodeMCU, WeMOS and many more), running [ESP Easy](#) with associated sensors, communicating (send/receive data) with Domoticz.

### Get Started

If not familiar with ESP Easy, start with the previous link shared.

Furthermore recommend to learn about rules, especially the new [rules engine](#) and the [command reference](#).

## Hardware

- ESP8266 NodeMCU Ver 1.0 DevKit (HW-389) 4M LoLin.
- DFRobot component "Digital Piranha Blue LED Module V2".
- DFRobot component "Digital Push Button Red V2".
- GY-302 BH1750 16-Bit Serial Output Light LUX Sensor Module.
- Breadboard, various wires, resistors (where required).

### Notes

- In circuits using DFRobot LED components, no additional resistors are used ([info](#)).
- Tested the WiFi range of the NodeMCU, which is about 7 meters.

## Software

- ESP Easy Release mega-20200608 - see [GitHub](#).  
Domoticz Beta channel 2020.2 (build 12143).

### Notes

As beta versions are used, check for updates regularly.

## Overview Experiments

Just a few experiments, to test how to connect and read values from sensors.

- Switch an LED connected to the ESP
- Measure ambient light with sensor BH1750 (I2C) connected to the ESP
  - Send Lux value to Domoticz device
  - Trigger switching a connected LED if below darkness threshold
- Potentiometer connected to the ESP, to control the brightness of a Hue light bulb, which is controlled via Domoticz

# Flash ESP

## Archive

1. Download latest ESP82xxarchive.  
Used ESPEasy\_ESP82xx\_mega-20200608.zip (~60MB) from GitHub.
2. Extract the archive to a temp folder, i.e. domoticz-home-automation-workbook\functions\ESPEasy\ distribution\).

## Flashing

Connect the ESP to an USB port of a Windows 10 device.

From the temp folder, start **ESP.Easy.Flasher.exe** (used version 0.04.00).

Set parameter:

- COM Port: COM4 USB Serial CH340
- Firmware: ESP\_Easy\_mega\_20200608\_normal\_ESP8266\_4M1M.bin  
(The NodeMCU has 4M).
- WiFi SSID (main): SSID
- WiFi password (main): \*\*\*\*
- Fixed IP: 192.168.1.92

Press button [Flash ESP Easy FW]

The ESP blue light near the module, starts flashing. The Flasher log shows:

```
(If the post-flash stall, press reset on ESP unit)
Listening to serial for ~10-30s (started at: 09:14:09)...
Serial: WifiConnect
Serial: Save
Serial: subnet 255.255.255.0
Serial: dns 8.8.8.8
Serial: gateway 192.168.1.1
Serial: ip 192.168.1.88
Serial: WifiKey YOURKEY
Serial: WifISSID YOURSSID
Serial: Unit 1
Serial: Name ESPEasy1
Serial: ...
Adding settings using SERIAL commands.
Flashing done (90.6s).
Start flashing on COM3 (approx 102 seconds)
```

Flashing was successful and the ESP can be reached via browser with the fix IP set.

## Error Flashing

If error during flashing, reset the ESP and try flashing again.

Ensure the serial monitor is not open.

If after flashing the serial monitor is available, then check the WiFi status.

If there is an error, i.e. “no valid WiFi settings” connecting, then following steps to resolve:

```
INIT : Booting version: mega-20200608 (ESP82xx Core a5432625, NONOS SDK 2.2.2-dev(38a443e), LWIP: 2.1.2
PUYA support)
89 : Info : INIT : Free RAM:32792
90 : Info : INIT : Warm boot #5 Last Task: Const Interval timer, id: 1 Last systime: 179 - Restart
Reason: External System
91 : Info : FS   : Mounting...
116 : Info : FS   : Mount successful, used 75802 bytes of 957314
133 : Info : CRC  : No program memory checksum found. Check output of crc2.py
140 : Info : CRC  : SecuritySettings CRC    ...OK
144 : Info : WIFI : Start network scan
2439 : Info : INIT : Free RAM:29504
```

```

2440 : Info : INIT : I2C
2440 : Info : INIT : SPI not enabled
2518 : Info : INFO : Plugins: 46 [Normal] (ESP82xx Core a5432625, NONOS SDK 2.2.2-dev(38a443e), LWIP: 2.1.2 PUYA support)
2522 : Info : Webserver: start
2523 : Info : Time set to 179.000 Time adjusted by 176478.00 msec. Wander: 49.02 msec/second
2524 : Info : Current Time Zone: STD time start: 1970-10-25 03:00:00 offset: 0 min
2526 : Info : Local time: 1970-01-01 00:02:59
2540 : Info : WIFI : Scan finished, found: 2
2546 : Error : WIFI : No valid wifi settings
2648 : Info : WiFi : Set WiFi to AP+STA
3638 : Info : WiFi : AP Mode ssid will be ESP_Easy with address 192.168.4.1
3639 : Error : WiFi : Could not prepare WiFi!
3874 : Info : WD : Uptime 0 ConnectFailures 0 FreeMem 20456 WiFiStatus 0
33882 : Info : WD : Uptime 1 ConnectFailures 0 FreeMem 20440 WiFiStatus 0
63883 : Info : WD : Uptime 1 ConnectFailures 0 FreeMem 20440 WiFiStatus 0

```

1. Connect to the network named "ESP\_Easy".
2. Open browser:
  - a. A connection is made to <http://www.msftconnecttest.com/setup> with page title "Welcome to ESP Easy Mega AP", Wifi Setup wizard.
  - b. If no connection is made, enter the IP addresss as stated in the log, i.e. <http://192.168.4.1>.
3. Pick a local network and Submit.
4. If connection failed: Select as network "Other", enter SSID/Password, connect again.
5. If connection ok: In the browser, page "WiFi setup complete" shows up.
6. Select Config to set a fixed IP under "IP Settings" and Submit
7. Switch to the default network.
8. Reset the ESP again.
9. Open browser and connect to the ESP fixed IP.
10. The main page is shown with system info, i.e. Git Build: mega-20200608 and IP address.
11. Under Tools > Advanced Settings set fixed IP address.

## Update ESP

Update the ESP Easy Firmware using the ESP Easy Web Frontend.  
(example below from earlier version)

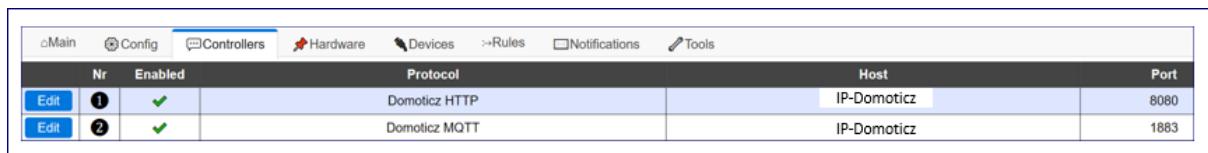
1. Download the latest version of the firmware, i.e. [https://github.com/letscontrolit/ESPEasy/releases/download/mega-20190315/ESPEasy\\_mega-20190315.zip](https://github.com/letscontrolit/ESPEasy/releases/download/mega-20190315/ESPEasy_mega-20190315.zip)
2. Unzip to a temp folder, i.e. `domoticz-home-automation-workbook\functions\ESPEasy\distribution\`.
3. Open a browser
4. Enter the ESP Easy URL
5. Select menu Tools > Update Firmware
6. Select `ESP_Easy_mega-20190315_normal_ESP8266_4M.bin`
7. Press Update
8. Wait for completion which is followed by a reboot of the NodeMCU
9. Refresh the browser window (F5)
10. Check menu Tools > System Info if version is correct updated

## Domoticz Controller

Communication between ESP Easy and Domoticz is possible via Controllers with the protocol's HTTP and MQTT.

In the ESP Easy web interface ESP Easy Mega: ESP\_Easy, changed the Domoticz Controller (Menu Controllers) configuration parameters:

- Locate Controller: Use IP Address
- Controller IP: Domoticz-IP address
- Port: Used the defaults, i.e. HTTP=8080, MQTT: 1883
- Enabled: Ticked
- The other settings, used the defaults



Main	Config	Controllers	Hardware	Devices	Rules	Notifications	Tools
Nr	Enabled	Protocol				Host	Port
Edit	1	✓	Domoticz HTTP			IP-Domoticz	8080
Edit	2	✓	Domoticz MQTT			IP-Domoticz	1883

## ESP Easy Rules

[ESP Easy Rules](#) can be used to create simple flows to control devices on the ESP.

### New Rules

ESP Easy new [rules engine](#) (Tools > Advanced > Rules Settings > Disable "Old Engine").

#### Hints

- The maximum size of a rule is 2048 characters.
- The event name must be the same as the event trigger (device#trigger), i.d.c. PushButton#State.
- ESP Easy converts the rule to a lowercase filename located in rules/devientname/trigger, i.e. rules/pushbutton/state.txt.
- Do not use decimals in the device name. If the event name is not correct, the input field has a red border else green.

### Example Old Rule

The old rule used is "Rules Set #1".

If the Lux value of a BH1750 sensor is below a threshold of 100, an LED connected to pin D5 (GPIO14) is set to ON (= state changes from 0 LOW to 1 HIGH).

### Rule Code

```
// Check ambient light device named ESPEasy_BH1750 value Lux
// (see devices properties).
// If Lux value less threshold (i.e. 100), switch the RED LED ON else OFF.
// The RED LED is connected to NodeMCU pin D5 (GPIO14)
// The command syntax: gpio, pin number, 1|0 - 1=ON,0=OFF

on ESPEasy_BH1750#Lux do

    if [ESPEasy_BH1750#Lux] < 100
        gpio,14,1
    else
        gpio,14,0
    endif

endon
```

### ESP Easy Log

Rules are logged (Menu Tools > System Log).

Logic: The LED state is changed from 0 (OFF) to 1 (ON), because the Light intensity alue (Lux) dropped below 100 to 63.33.

The event ESPEasy\_BH1750#Lux handles the Lux value change.

```
1011480: EVENT: ESPEasy_BH1750#Lux=120.00
1011499: ACT : gpio,14,0
1011502: SW : GPIO 14 Set to 0
1011524: Domoticz: Sensortype: 1 idx: 46 values: 120.00
1021477: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 63.33
1021480: EVENT: ESPEasy_BH1750#Lux=63.33
1021498: ACT : gpio,14,1
1021500: SW : GPIO 14 Set to 1
1021526: Domoticz: Sensortype: 1 idx: 46 values: 63.33
```

## Device Type (Light/Lux – BH1750), Name (ESPEasy\_BH1750) and Value (Lux)

Light/Lux - BH1750	ESPEasy_BH1750		<b>2</b> (46)	GPIO-4 GPIO-5	Lux:	183.33
<b>Values</b>						
#						Name
1	Lux					

## Example New Rule

This example uses two devices

- Task 1: push-button connected to pin D7 (GPIO13)
- Task 2: LED connected to pin D5 (GPIO14)

Logic: If the push-button is pressed, the LED is turned ON until the push-button is presed again to turn the LED OFF.

There is no communication with Domoticz but internal controlled by ESP Easy.

The ESP Easy devices details:

```
(Task,Enabled,Device,Name,Port,Ctr (IDX),GPIO,Values)
1,yes,Switch input - Switch,PushButton,,,GPIO-13,State:0
2,yes,Switch input - Switch,LED,,,GPIO-14,State:0
```

The push-button uses Inversed logic to set state 1 if pressed.

## Rule Code

```
// Rule: pushbutton#state
// Filename: rules/pushbutton/state.txt
// If the pushbutton (D7,GPIO13) pressed, the LED (D5,GPIO14) is turned on
// If pressed again, the LED is turned off
on PushButton#State do
    LogEntry, 'PushButton pressed. LED state: [LED#State]'
    if [LED#State] = 0
        GPIO,14,1
        LogEntry, 'LED state changed to ON'
    else
        GPIO,14,0
        LogEntry, 'LED state changed to OFF'
    endif
endon
```

## ESP Easy Log

```
7059433: EVENT: Clock#Time=Thu,04:52
7062548: SW : GPIO=13 State=1 Output value=1
7062560: EVENT: PushButton#State=1.00
7062602: ACT : LogEntry, 'PushButton pressed. LED state: 0'
7062604: Command: LogEntry
7062605: PushButton pressed. LED state: 0
7062617: ACT : GPIO,14,1
7062618: Command: GPIO
7062619: SW : GPIO 14 Set to 1
7062622: ACT : LogEntry, 'LED state changed to ON'
7062623: Command: LogEntry
7062624: LED state changed to ON
7062648: SW : GPIO=14 State=1 Output value=1
7062654: EVENT: LED#State=1.00
7066948: SW : GPIO=13 State=1 Output value=0
```

```
7066955: EVENT: PushButton#State=0.00
7066997: ACT : LogEntry, 'PushButton pressed. LED state: 1'
7066999: Command: LogEntry
7066999: PushButton pressed. LED state: 1
7067014: ACT : GPIO,14,0
7067015: Command: GPIO
7067017: SW : GPIO 14 Set to 0
7067019: ACT : LogEntry, 'LED state changed to OFF'
7067021: Command: LogEntry
7067022: LED state changed to OFF
7067048: SW : GPIO=14 State=0 Output value=0
7067054: EVENT: LED#State=0.00
```

## GPIO Commands

It is possible by using URL commands (HTTP Request), to control basic GPIO output. This could be used by Domoticz

- Device On/Off action
- Automation event

to trigger f.e. an LED On/Off or other like controlling a servo motor.

### Turn an LED On/Off

#### On

##### HTTP Request

```
http://espeasy-ip/control?cmd=gpio,2,1
```

##### HTTP Response

```
{"log": "GPIO 2 Set to 1", "plugin": 1, "pin": 2, "mode": "output", "state": 0}
```

#### Off

##### HTTP Request

```
http://espeasy-ip/control?cmd=gpio,2,0
```

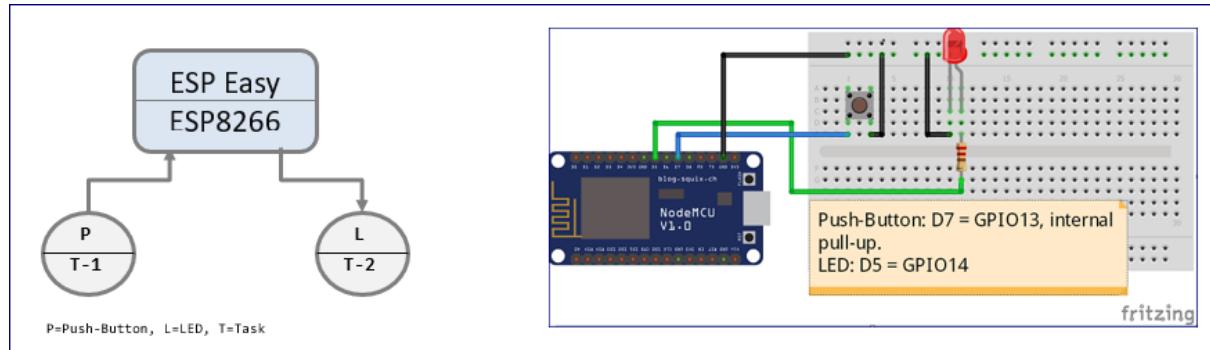
##### HTTP Response

```
{"log": "GPIO 2 Set to 0", "plugin": 1, "pin": 2, "mode": "output", "state": 1}
```

# ESP Push-Button switching ESP LED

## Purpose

To switch an LED On/Off via Push-Button using a rule without Domoticz.



## Wiring

Push-button	ESP
Signal	D7 (GPIO13)
GND	GND
<b>LED</b>	<b>ESP</b>
(+) Anode to R220 Ohm = D5 (GPIO14)	D5 (GPIO14)
(-) Cathode	GND

## ESP Easy

### Devices

Task,Enabled,Device,Name,Port,Ctr (IDX),GPIO,Values
1,YES,Switch input - Switch,PushButton,,,GPIO-13,State:1
2,YES,Switch input - Switch,LED,,,GPIO-14,State:0

### Notes

The port and controller (Ctr) are not used.

The push-button has

- Inversed Logic: True
- Switch Type: Switch
- Switch Button Type: Push Button Active High.

### Rule

```
// Rule: pushbutton#state
// Filename: rules/pushbutton/state.txt
// 20200624 rwbl
on PushButton#State do
    LogEntry, 'PushButton pressed. State: [PushButton#State]'
    if [LED#State] = 0
        GPIO,14,1
        LogEntry, 'LED state changed to ON'
    else
        GPIO,14,0
        LogEntry, 'LED state changed to OFF'
    endif
endon
```

## Log

```
4983049: SW : GPIO=13 State=1 Output value=0
4983055: EVENT: PushButton#State=0.00
4983107: ACT : LogEntry, 'PushButton pressed. State: 0'
4983111: Command: LogEntry
4983112: PushButton pressed. State: 0
4983128: ACT : GPIO,14,1
4983129: Command: GPIO
4983131: SW : GPIO 14 Set to 1
4983134: ACT : LogEntry, 'LED state changed to ON'
4983135: Command: LogEntry
4983136: LED state changed to ON
4983149: SW : GPIO=14 State=1 Output value=1
4983155: EVENT: LED#State=1.00
4986249: SW : GPIO=13 State=1 Output value=1
4986255: EVENT: PushButton#State=1.00
4986289: ACT : LogEntry, 'PushButton pressed. State: 1'
4986291: Command: LogEntry
4986291: PushButton pressed. State: 1
4986306: ACT : GPIO,14,0
4986307: Command: GPIO
4986309: SW : GPIO 14 Set to 0
4986311: ACT : LogEntry, 'LED state changed to OFF'
4986313: Command: LogEntry
4986314: LED state changed to OFF
4986349: SW : GPIO=14 State=0 Output value=0
4986355: EVENT: LED#State=0.00
```

## Domoticz

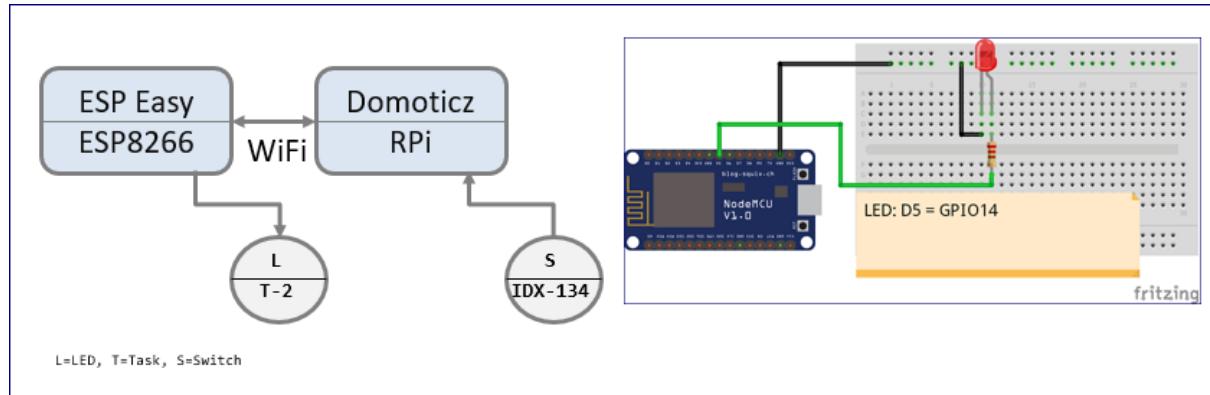
### Devices

Not used.

# Domoticz Switch ESP LED

## Purpose

To switch, via a Domoticz Switch, the LED connected to pin D5 (GPIO14) of the ESP.



## Wiring

LED	ESP
(+) Anode to R220 Ohm	D5 (GPIO14) [green]
(-) Cathode	GND [black]

## ESP Easy

### Devices

No devices required; LED controlled via HTTP commands submitted by Domoticz.

*Note*

In the picture, LED shown as device Task-2, but that is related to the previous experiment.

## Domoticz

### Devices

Create Switch Device (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
134, VirtualDevices, ESPEasy LED Switch, Light/Switch, Switch, On/Off, Off

Define the device action HTTP request:

Switch Type	HTTP Commands to switch GPIO14 On (1) / Off (0)
On Action	http://espeeasy-ip/control?cmd=gpio,14,1
Off Action	http://espeeasy-ip/control?cmd=gpio,14,0

Test: click switch On/Off, check the LED is turned On / Off.

### Domoticz Log

2020-06-25 08:42:44.450 Status: User: Admin initiated a switch command (134/ESPEasy LED Switch/On)
2020-06-25 08:43:25.545 Status: User: Admin initiated a switch command (134/ESPEasy LED Switch/Off)

## Automation Event

Just an Automation Event dzVents example handling the switch device changes, i.d.c. log some text.

```
-- Event: espeasy_ledswitch.dzvents
-- Handle changes of the switch device
local IDX_LED SWITCH = 134
local STATEON = "On"
local STATEOFF = "Off"

return {
  on = {
    devices = {IDX_LED SWITCH}
  },
  execute = function(domoticz, device)
    domoticz.log('Device ' .. device.name .. ' was changed', domoticz.LOG_INFO)
    domoticz.log(string.format('State: %s', domoticz.devices(IDX_LED SWITCH).state))
    local state = domoticz.devices(IDX_LED SWITCH).state
    if (state == STATEON) then
      domoticz.log('State ON, please take action!')
    end
    if (state == STATEOFF) then
      domoticz.log('State OFF, please take action!')
    end
  end
}
```

## Domoticz Log

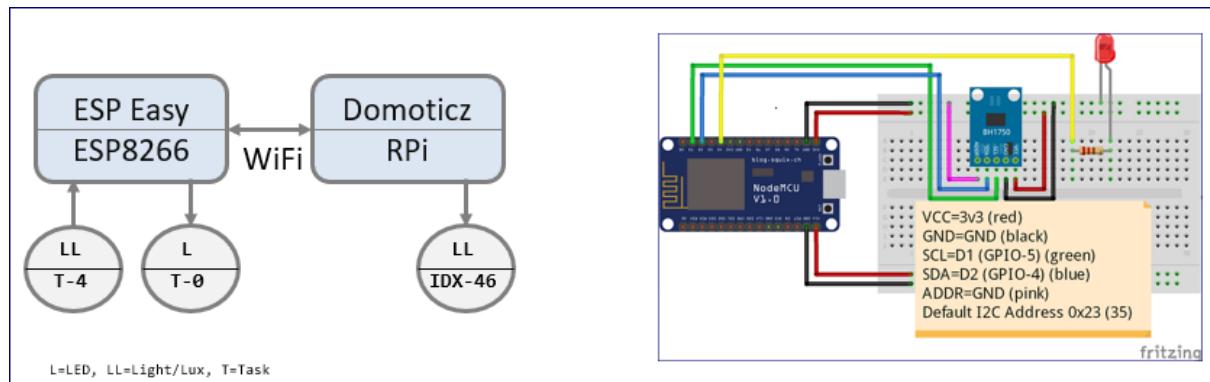
```
2020-06-25 08:42:44.614 Status: dzVents: Info: Handling events for: "ESPEasy LED Switch", value: "On"
2020-06-25 08:42:44.614 Status: dzVents: Info: ----- Start internal script: espeasy-led-domoticz:
Device: "ESPEasy LED Switch (VirtualDevices)", Index: 134
2020-06-25 08:42:44.614 Status: dzVents: Info: Device ESPEasy LED Switch was changed
2020-06-25 08:42:44.614 Status: dzVents: Info: State: On
2020-06-25 08:42:44.614 Status: dzVents: Info: State ON, please act!
2020-06-25 08:42:44.614 Status: dzVents: Info: ----- Finished espeasy-led-domoticz
2020-06-25 08:43:25.551 (VirtualDevices) Light/Switch (ESPEasy LED Switch)

2020-06-25 08:43:25.545 Status: User: Admin initiated a switch command (134/ESPEasy LED Switch/Off)
2020-06-25 08:43:25.709 Status: dzVents: Info: Handling events for: "ESPEasy LED Switch", value: "Off"
2020-06-25 08:43:25.709 Status: dzVents: Info: ----- Start internal script: espeasy-led-domoticz:
Device: "ESPEasy LED Switch (VirtualDevices)", Index: 134
2020-06-25 08:43:25.709 Status: dzVents: Info: Device ESPEasy LED Switch was changed
2020-06-25 08:43:25.709 Status: dzVents: Info: State: Off
2020-06-25 08:43:25.710 Status: dzVents: Info: State OFF, please act!
2020-06-25 08:43:25.710 Status: dzVents: Info: ----- Finished espeasy-led-domoticz
```

# ESP Ambient Light to Domoticz

## Purpose

To measure the ambient light from the GY-302 BH1750 16-Bit Serial Output Light LUX Sensor Module connected to the ESP and send the Lux value to a Domoticz Lux device.



## Wiring

BH1750	ESP [wirecolor]
SCL	D1 (GPIO5) [green]
SDA	D2 (GPIO4) [blue]
ADDR	GND [pink] Default I2C address used: 0x23 (35)
VCC	3V3 [red]
GND	GND [black]
<b>LED</b>	<b>ESP</b>
(+) Anode to R220 Ohm = D5 (GPIO14)	D5 (GPIO14)
(-) Cathode	GND

## ESP Easy

### Devices

Task,Enabled,Device,Name,Port,Ctr (IDX),GPIO,Values
4,True,Light/Lux - BH1750,ESPEasy_BH1750,I2C,2(46),SDA: 4,SCL: 5,Lux:353.33

### Notes

Controller 2 (Domoticz MQTT) is used for the Domoticz device with IDX=46 (see next).

## Domoticz

### Devices

Create Lux Device (Hardware Virtual Sensor) with name ESPEasy Ambient Light.

Idx, Hardware, Name, Type, SubType, SwitchType, Data
46, VirtualDevices, ESPEasy Ambient Light, Lux, Lux, 197 Lux

After creating the device, check the Domoticz Log, if data from the sensor is received.

### Domoticz Log

```
2019-04-26 18:49:27.040 MQTT: Topic: domoticz/in, Message: {"idx":46, "RSSI":10, "nvalue":0, "svalue":"210.00"}
```

## Enhancement

### Purpose

To test using ESP Easy local variables with task Generic - Dummy Device.

If the measured Lux value is less than the threshold set (i.e. 100), switch a red LED ON else OFF. The red LED is connected to pin D5 (GPIO14) of the ESP.

There is no interaction with Domoticz, it is just to understand using variables with Generic – Dummy Device.

### GPIO Command Syntax

The GPIO command is used to switch the LED ON or OFF.

```
gpio, gpio number, 1|0
```

*Note*

State 1=ON, 0=OFF

### Generic – Dummy Device

A dummy device with two values is used to

- Set the threshold
- Track the state of the red LED indicator, to avoid sending GPIO commands permanently.

### Task Settings for the Generic – Dummy Device

- Device: Generic – Dummy Device (with task number 6)
- Name: AmbientLight\_Settings
- Value Index 1: Threshold, Decimals: 0
- Value Index 2: State, Decimals: 1

**Task Settings**

Device: Generic - Dummy Device

Name: AmbientLight\_Var

Enabled:

Simulate Data Type: SENSOR\_TYPE\_SINGLE

Data Acquisition

Send to Controller 1:  IDX: 0

Send to Controller 2:  IDX: 0

Interval: 3600 [sec]

Values

#	Name	Decimals
1	Threshold	0
2	State	0

## Task List

Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
<a href="#">Edit</a>	6	✓	Generic - Dummy Device	AmbientLight_Settings			Threshold: 100 State: 0 ..: 0.00 ..: 0.00

### Extract from the list of devices.

The Generic - Dummy Device named AmbientLight\_Settings, has task number 6 with the 3 variables with index 1 to 2.

This information is required to update a variable in the rule using the command:

```
TaskValueSet TaskNr,VariableIndex,Value
```

## ESP Easy Rule

```
// Check ambient light device named ESPEasy_BH1750 value Lux (see devices properties).
// If Lux value less then threshold (i.e. 100), switch the LED ON else OFF.
// The LED is connected to pin D5 (GPIO14).
// Command syntax: gpio, gpio number, 1|0 - 1=ON;0=OFF
// A generic dummy device is used to set the threshold at boot and track the state of the LED
indicator, to avoid sending GPIO commands permanently
// [AmbientLight_Settings#Threshold], [AmbientLight_Settings#State]

// Init the variables for device AmbientLight_Settings
on System#Boot do
    // Set ambient light threshold setting
    TaskValueSet 6,1,100
    // Turn LED off
    gpio,14,0
    // Update led state setting
    TaskValueSet 6,2,0
endon

// Handleambientlight lux changes
on ESPEasy_BH1750#Lux do

    // Check if Lux below threshold and LED is OFF
    if [ESPEasy_BH1750#Lux]<[AmbientLight_Settings#Threshold] and [AmbientLight_Settings#State]=0
        // Turn LED on and update dummy variable
        gpio,14,1
        TaskValueSet 6,2,1
    endif

    // Check if Lux above/equal threshold and LED is ON
    if [ESPEasy_BH1750#Lux]>=[AmbientLight_Settings#Threshold] and [AmbientLight_Settings#State]=1
        // Turn LED off and update dummy variable
        gpio,14,0
        TaskValueSet 6,2,0
    endif

endon
```

## ESP Easy Log Entries

**LED switched ON triggered by Lux value 6.67 below threshold 100.**

```
531124: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 6.67
531128: EVENT: ESPEasy_BH1750#Lux=6.67
531259: ACT : gpio,14,1
531263: SW : GPIO 14 Set to 1
531266: ACT : TaskValueSet 6,2,1
531273: Command: taskvalueset
531394: Domoticz: Sensortype: 1 idx: 46 values: 6.67
```

**LED switched OFF triggered by Lux value 273.33 above threshold 100.**

```
546131: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 273.33
546135: EVENT: ESPEasy_BH1750#Lux=273.33
546324: ACT : gpio,14,0
546328: SW : GPIO 14 Set to 0
546331: ACT : TaskValueSet 6,2,0
546338: Command: taskvalueset
546431: Domoticz: Sensortype: 1 idx: 46 values: 273.33
```

# ESP controlling Hue Brightness

## Purpose

To control the brightness of the Hue light “MakeLab” with IDX=118 via potentiometer connected to an ESP8266 NodeMCU.

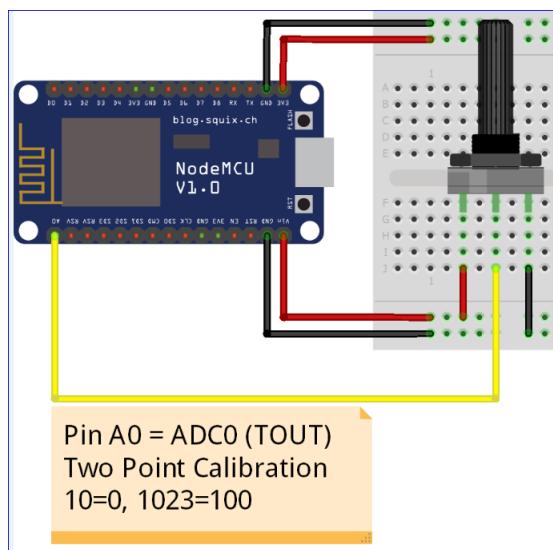
## Parts List

- 1x NodeMCU
- 1x Potentiometer
- 1x Breadboard
- 3x Wires male-male

## Wiring

Potentiometer	NodeMCU
VCC	Vin 5v – power supplied via USB from PC
Signal ADC0 (TOUT)	A0
GND	GND

## Circuit



## Solution 1

This preferred solution uses an ESP Easy Analog input device, ESP Easy Dummy Devices and ESP Easy Rule.

The ESP Easy rule updates the Hue brightness using the SendToHTTP command, which is the preferred method to update Domoticz devices.

### ESP Easy Device Configuration - Analog Input

The screenshot shows the 'Task Settings' tab for an 'Analog input - internal' device. The device is named 'HUE\_Dimmer'. It is enabled and has 'Oversampling' checked. Under 'Two Point Calibration', 'Calibration Enabled' is checked, with 'Point 1' at 10 (0.000) and 'Point 2' at 1023 (100.000). Current values are 9 ± 0.099, Minimum is 0 ± 0.097, Maximum is 1023 ± 100.000, and Step size is 1 ± 0.099. In the 'Data Acquisition' section, 'Send to Controller' is checked for both IDX 154 and IDX 118, with an interval of 2 seconds. The 'Values' section lists an analog value with a formula and 0 decimal places.

### ESP Easy Device Configuration – Dummy Devices

The screenshot shows the 'Task Settings' tab for a 'Generic - Dummy Device' named 'Varstore'. It is enabled and simulates 'SENSOR\_TYPE\_SINGLE' data. In the 'Data Acquisition' section, 'Send to Controller' is checked for both IDX 0 and IDX 1, with an interval of 60 seconds. The 'Values' section lists two entries: 'HUE\_Dimmer' and 'HUE\_Dimmer\_Delta', both with 0 decimal places.

## ESP Easy Device Configuration – Overview Devices

The devices used, with name and value, are

- HUE\_Dimmer with value Analog
- Varstore with values HUE\_Dimmer and HUE\_Dimmer\_Delta

ESP Easy Mega: ESP_Easy							
Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
Edit 1	✓	Analog input - internal	HUE_Dimmer			ADC (TOUT)	Analog: 40
Edit 2	✗	Switch input - Switch	Hue_Light_Switch		1 (154)	GPIO-14	Switch: 0
Edit 3	✗	Display - LCD2004				GPIO-4 GPIO-5	
Edit 4	✓	Light/Lux - BH1750	ESPEasy_BH1750		2 (46)	GPIO-4 GPIO-5	Lux: 46.67
Edit 5	✓	Generic - Dummy Device	Varstore				HUE_Dimmer: 40 HUE_Dimmer_Delta: 1 : 0.00 : 0.00

## ESP Easy Rule

```
// Control Hue light via Potmeter named HUE_Dimmer with value Analog.
// A dummy device named Varstore (TaskNr 5) is used to store 2 values:
// Value 1 = HUE_Dimmer for the previous Hue light value
// Value 2 = HUE_Dimmer_Delta for the absolute delta brightness between Hue light and previous value

// Listen to Hue light analog value changes
on HUE_Dimmer#Analog do

    // Save the delta to dummy variable value 2
    TaskValueSet 5,2,[HUE_Dimmer#Analog]-[Varstore#HUE_Dimmer]
    Delay 5

    // Get the absolute delta for the new dimmer value
    // Update the dummy variable #2
    if [Varstore#HUE_Dimmer_Delta]<0

        TaskValueSet 5,2,[Varstore#HUE_Dimmer_Delta]*-1

    endif

    // Check if delta > 2, this to avoid noise
    // Submit api rest request to domoticz system to set brightness level
    if [Varstore#HUE_Dimmer_Delta]>2

        SendToHTTP domoticz-
        ip,8080,/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=[HUE_Dimmer#Analog]
    endif

    // Update the dummy variable #1
    TaskValueSet 5,1,[HUE_Dimmer#Analog]

endon
```

## Solution 2

This solution uses an ESP Easy Analog input device, Domoticz Text device (Hardware Virtual Sensor) and Domoticz dzVents script.

It is not the preferred way to set the brightness, it is just to show converting the MQTT default command into a value.

### ESP Easy

Hardware	<i>Potentiometer = ESP8266 NodeMCU</i> VCC = Vin (5v) GND = GND Signal = A0
Devices	

#### Notes

When using ESPEasy Controller #2 with a Hue Device (for example IDX=118), the Domoticz log shows:

```
2019-04-26 19:02:24.720 MQTT: Topic: domoticz/in, Message:  
{"idx":118,"RSSI":10,"nvalue":0,"svalue":"75"}
```

With this command, a Hue light cannot be controlled.

To control a Hue light, this command is required:

```
{"command": "switchlight", "idx": 118, "switchcmd": "Set Level", "level": 50}
```

As a workaround a Domoticz Text Device (Hardware Virtual Sensor) with a dzVents script is used.

## Domoticz

Create a Text device (Hardware Virtual Sensor) with name ESPEasy Hue Light.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	154	VirtualSensors	00082154	1	ESPEasy Hue Light	General	Text	51

Create a dzVents Lua script event.

Event name: hue\_control\_espeasy

```
-- [[
hue_control_espeasy.lua
Control a Hue Light via ESP8266 NodeMCU running ESPEasy.
]]-- 

-- Idx of the devices used
local IDX_ESPEASY_HUE_LIGHT = 154
local IDX_HUE_MAKELAB = 118

return {
  on = {
    devices = {
      IDX_ESPEASY_HUE_LIGHT
    },
  },
  data = {
    prevvalue = { initial = -1 }
  },
  execute = function(domoticz, device)
    local currvalue = math.floor(tonumber(device.text))
    if domoticz.data.prevvalue == -1 then
      domoticz.data.prevvalue = currvalue
    end
    domoticz.log('Device ' .. device.name .. ' changed from ' .. tostring(domoticz.data.prevvalue) ..
' to ' .. tostring(currvalue), domoticz.LOG_INFO)
    -- handle noise, i.e. make changes delta > 1
    if (math.abs(domoticz.data.prevvalue - currvalue)) > 1 then
      domoticz.data.prevvalue = currvalue
      if (domoticz.devices(IDX_HUE_MAKELAB).state == 'Off') then
        domoticz.devices(IDX_HUE_MAKELAB).switchOn()
      end
      domoticz.devices(IDX_HUE_MAKELAB).dimTo(currvalue)
    end
  end
}
```

# ESP BMP280 to Domoticz

## Purpose

To measure the temperature and air pressure from the sensor BMP280 connected to the NodeMCU and send the values to a Domoticz device.

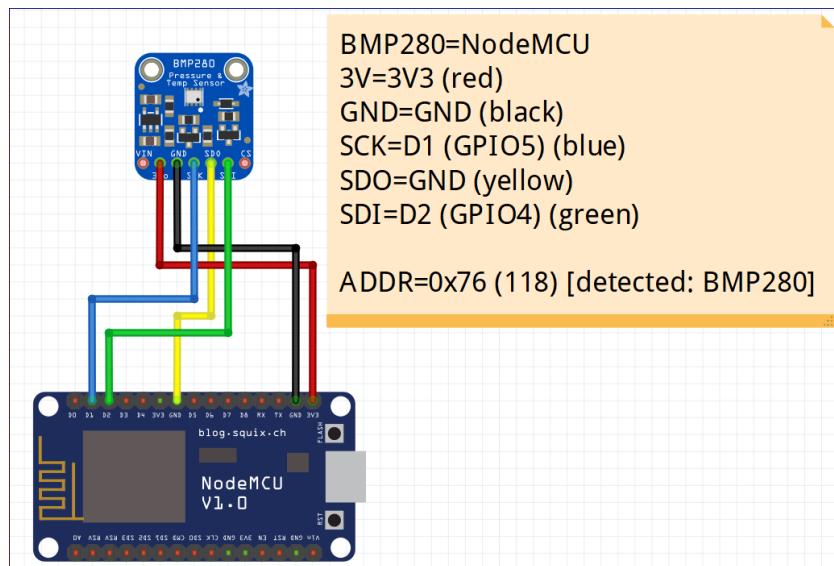
## Parts List

- 1x NodeMCU
- 1x BMP280 sensor
- 1x Breadboard
- 5x Wires female-female

## Wiring

BMP280	NodeMCU [wirecolor]
VIN	n/a
3V	3V [red]
GND	GND [black]
SCK	D1 (GPIO5) [blue]
SDO	GND [yellow]
SDI	D2 (GPIO4) [green]
CS	n/a

## Circuit



## ESP Easy

### Task Settings for Device Environment - BMx280

**Task Settings**

Device: Environment - BMx280

Name: ESPEasy\_BMP280

Enabled:

I2C Address: 0x76 (118) - (default) [Detected BMP280]

Note: SDO Low=0x76, High=0x77

Altitude: 0 [m]

Temperature offset: 0 [x 0.1C]

Note: Offset in units of 0.1 degree Celcius

**Data Acquisition**

Send to Controller

IDX: 60

Interval: 5 [sec]

**Values**

#	Name	Formula	Decimals
1	Temperature		2
2	Humidity		2
3	Pressure		2

#### Note

- The controller 1 is used. This is a Domotocz MQTT controller.
- The Idx of the Domoticz device receiving the data, is taken from next step adding a Domoticz Temp+Baro device.

After adding check, the if the values of the device are updating.

Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
<a href="#">Edit</a>	1 ✓	Environment - BMx280	ESPEasy_BMP280			GPIO:4 GPIO:5	Temperature: Humidity: Pressure:

The data is OK. Note that Humidity is 0 as not supported by this device BMP280.  
To also measure Humidity, use a BME280

## Domoticz

Create a Temp+Baro device (Hardware Virtual Sensor) with name ESPEasy\_BMP280



Idx	Hardware	ID	Unit	Name	Type	SubType	Data
60	VirtualDevices	1408C	1	ESPEasy_BMP280	Temp + Baro	BMP085 I2C	0.0 C, 1038.0 hPa

The Domoticz Idx is used in the ESP Easy device ESPEasy\_BMP280 to set the property IDX for controller 1.

Check the Domoticz Log if data from the sensor is received:

```
2019-07-14 19:00:47.890 MQTT: Topic: domoticz/in, Message:  
{"idx":60,"RSSI":9,"nvalue":0,"svalue":"21.71;1017.74;0;0"}
```

*Note*

The svalue contains:

Temperature;Barometric pressure;Barometer forecast;Altitude (not used, set to 0)

The widgets.

GUI Tab Temperature	GUI Tab Weather
 <b>ESPEasy_BMP280</b> <b>21.71°C</b> Barometer: 1017.78 hPa, Prediction: Stable Last Seen: 2019-07-14 19:05:43     	 <b>ESPEasy_BMP280</b> <b>1017.78 hPa</b> Prediction: Stable Last Seen: 2019-07-14 19:06:39     

The Barometer forecast can be one of:

0 = No Info, 1 = Sunny, 2 = Partly Cloudy, 3 = Cloudy, 4 = Rain

ESP Easy always delivers value 0, means the forecast requires to be adjusted.

<TODO> Find a way using the Domoticz internal calculations.

# Events

## Purpose

To explore the Domoticz Events System and the Event Editor (see GUI Setup > More Options > Events) supporting the interpreter:  
Blockly, Lua, Python, dzVents (next generation Lua Scripting)

*Some notes regarding the various Domoticz Interpreters.*

### Blockly

NOT USED – all events via dzVents. For learning & testing developed a script.

### Lua

NOT USED – all events via dzVents. For learning & testing developed a script.

### Lua dzVents

Target is using the next generation Lua Scripting dzVents - Domoticz Easy Events – with the Domoticz build-in Editor.

Ensure to read [this](#) first.

#### *Note*

The Lua editor is context sensitive, prompt with auto-complete options and show common errors to help with debugging. The editor theme can be changed by pressing the control and comma keys at the same time. Domoticz will save the scheme when events are saved.

### Python

In favour to program using Python instead Lua, **BUT** the new **dzVents** is so great, that for event scripting will use dzVents.

For building Domoticz PlugIns will use Python.

#### *Note*

The Python Event System uses Python3 (same as Python Plugin System). Events are enabled along with plugins; it is no longer possible to enable one without the other.

#### *Note*

The file EXPLORE\_SCRIPTING.md (in folder scripts) contains sample dzVents Lua, Lua, Python and Blockly scripts to explore automation events.

These scripts are not used anymore but kept as learning reference.

## Event Execution Order

The order of events being executed, is:

1. file scripts - stored in the folder /home/pi/domoticz/scripts/...
2. database scripts - stored in tables [EventMaster] and [EventRules]
3. on/off action script - explore usage & storage

#### **Important** (*taken from the Domoticz Wiki*)

Scripts that written on the file system and inside Domoticz using the internal web-editor share the same namespace. If there are two scripts with the same name, only the file system script will be used – this is logged.

## Event Scheduling & Trigger

The scheduling is fixed in Domoticz.

- ALL "device" scripts (scripts with "device" in their name) run everytime ANY device status changes.
- ALL "time" scripts (scripts with "time" in their name) run every MINUTE.

So far, have been using file scripts but considering moving to database solution only for device events.

For event triggered once per day, target to use crontab with Python scripts.

### Important

The execution duration of any event-based script (dzVents, Lua, Python) should be kept to a minimum as the Domoticz event-system is single threaded.

This means that during execution of a script, no other script can start nor runs.

## Event Database Tables

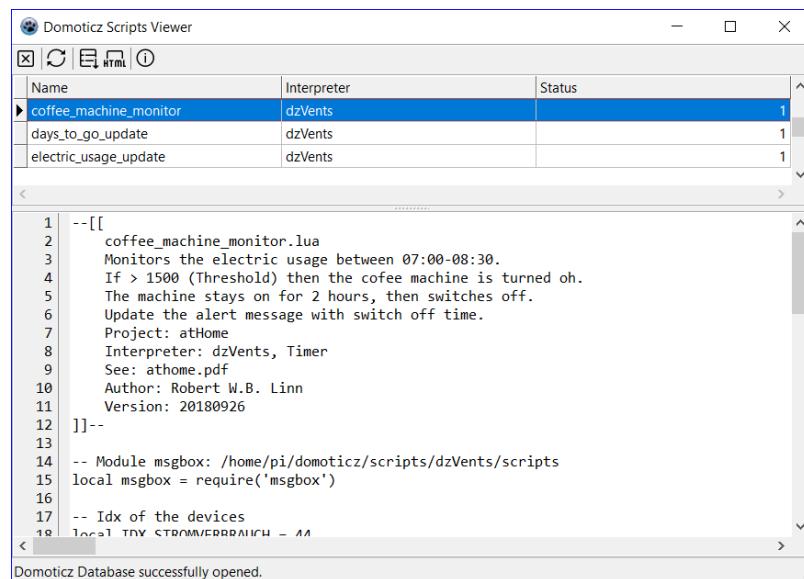
The events which are created using the Domoticz Event Editor, are stored in tables [EventMaster] and [EventRules]

```
CREATE TABLE [EventMaster] ([ID] INTEGER PRIMARY KEY, [Name] VARCHAR(200) NOT NULL, [XMLStatement] TEXT NOT NULL, [Status] INTEGER DEFAULT 0, [Interpreter] VARCHAR(10) DEFAULT 'Blockly', [Type] VARCHAR(10) DEFAULT 'All');
```

```
CREATE TABLE [EventRules] ([ID] INTEGER PRIMARY KEY, [EMID] INTEGER, [Conditions] TEXT NOT NULL, [Actions] TEXT NOT NULL, [SequenceNo] INTEGER NOT NULL, FOREIGN KEY (EMID) REFERENCES EventMaster(ID));
```

## Event Script Viewer

Developed, with Lazarus, an application to view the dzVents scripts.



The screenshot shows a window titled "Domoticz Scripts Viewer". At the top, there are four tabs: "Domoticz", "File", "HTML", and "Edit". Below the tabs is a table with three columns: "Name", "Interpreter", and "Status". The table contains three rows:

Name	Interpreter	Status
coffee_machine_monitor	dzVents	1
days_to_go_update	dzVents	1
electric_usage_update	dzVents	1

Below the table is a large text area containing the script code. The code is as follows:

```

1 --[[[
2   coffee_machine_monitor.lua
3   Monitors the electric usage between 07:00-08:30.
4   If > 1500 (Threshold) then the coffee machine is turned on.
5   The machine stays on for 2 hours, then switches off.
6   Update the alert message with switch off time.
7   Project: athome
8   Interpreter: dzVents, Timer
9   See: athome.pdf
10  Author: Robert W.B. Linn
11  Version: 20180926
12 ]]]-
13
14 -- Module messagebox: /home/pi/domoticz/scripts/dzVents/scripts
15 local messagebox = require('messagebox')
16
17 -- Idx of the devices
18 local idx_startbranch = 11

```

At the bottom of the window, a status message reads "Domoticz Database successfully opened."

## Event Development

Whilst creating scripts using the Event Editor, two browser tabs are open:

- Domoticz Event Editor – to create/modify scripts using Ace Editor, to activate/deactivate a script,
  - Domoticz Log (Domoticz GUI Setup > Log > Tab All) – to check if the script is running OK after save.

# Example Basement Humidity Monitor

## Purpose

To explore the various interpreter using the same event to monitor a device:

Device: Keller (IDX=11)

Property: humidity

Rule:

If the Basement Humidity is  $\geq 70$  (threshold), write a message to the Domoticz Log.

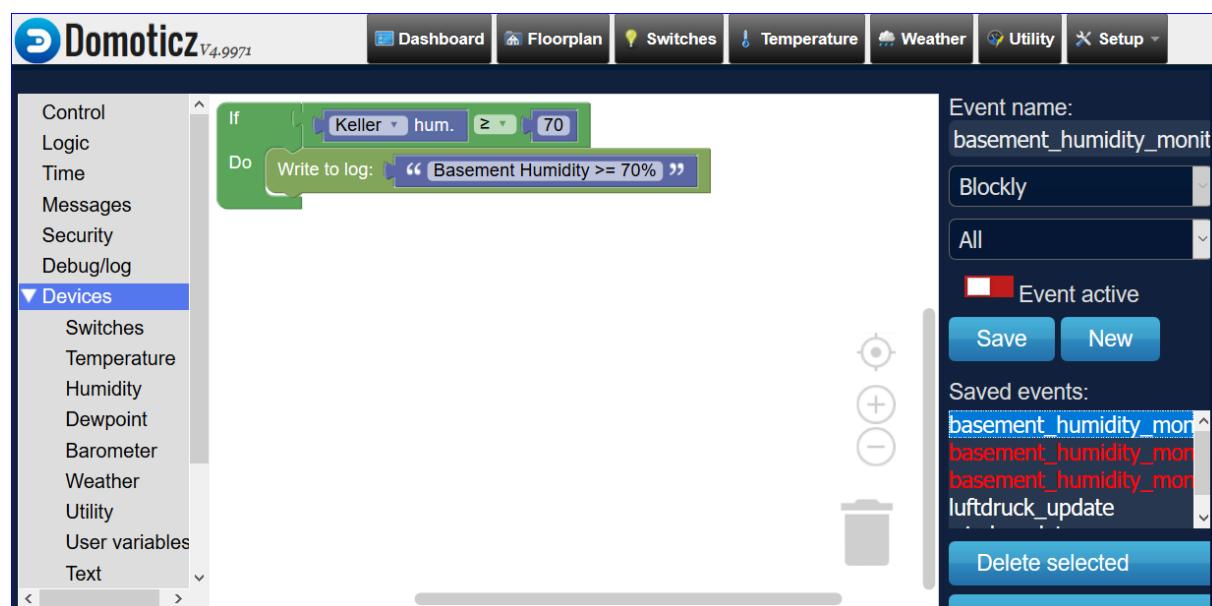
### Note

For the sample scripts, added to the Event Names, a suffix as an interpreter indicator:

\_d (Blockly), \_l (Lua), \_d (dzVents), \_p (Python).

## Blockly

**Event Properties:** basement\_humidity\_monitor\_b, Blockly, All.



## Domoticz Log

```
2018-09-01 10:32:04.478 EventSystem: Event triggered: basement_humidity_monitor_b
2018-09-01 10:32:04.478 Status: Basement Humidity >= 70%
```

## Lua

**Event Properties:** basement\_humidity\_monitor\_1, Lua, Device.

```

1 -- basement_humidity_monitor_1
2 -----
3 -- Monitor the basement humidity if exceeds threshold and write to Domoticz log.
4 -- @project: mysmarthome
5 -- @interpreter: Lua, Device
6 -- @author: Robert W.B. Linn
7 -- @since: 20180901
8 -- @version: 1.0
9 -- @see: mysmarthome.doc
10 -----
11 commandArray = {}
12 -----
13
14 -- Split a string by delimiter
15 function split(s, delimiter)
16     result = {}
17     for match in (s..delimiter):gmatch("(.-)"..delimiter) do
18         table.insert(result, match);
19     end
20     return result;
21 end
22 -----
23 -----
24 -- Domoticz IDX and names of the needed devices
25 -----
26 local KellerIdx = 11;
27 local KellerName = "Keller";
28 -----
29 -- Thresholds

```

## Script

```

commandArray = {}

-- Domoticz IDX and names of the needed devices
local KellerIdx = 11;
local KellerName = "Keller";

local HumidityThreshold = 70;

if devicechanged[KellerName] then
    time = os.date("*t")
    ts = string.format("%02d:%02d:%02d", time.hour, time.min, time.sec);
    print(string.format('Device %s changed @ %s', KellerName, ts));
    print('Device All Values: ' .. otherdevices_svalues[KellerName]);
    -- The svalues contain: TEMP;HUM;HUM_STAT
    -- TEMP = Temperature
    -- HUM = Humidity (0-100 %)
    -- HUM_STAT = Humidity status
    -- Get the humidity
    Humidity = math.floor(otherdevices_humidity[KellerName] * 10) / 10;

    -- Check if the humidity is equal or above threshold - notify
    if Humidity >= HumidityThreshold then
        print('Basement Humidity >= ' .. HumidityThreshold .. ' (' .. Humidity .. ')');
    end
end

return commandArray

```

## Domoticz Log

### *Sample Script – Error*

```
2018-09-01 13:20:29.442 Status: LUA: Device Keller changed @ 13:20:29
2018-09-01 13:20:29.442 Status: LUA: Device All Values: 18.7;74;3
2018-09-01 13:20:29.442 Error: EventSystem: in basement-humiditymonitor_1:
[string "-- script_device_basement-humidity..."]:41: bad argument #2 to 'format' (number expected, got
nil)
```

Caused by using wrong variable name, i.e. humidity instead of Humidity.

### *Sample Script – OK*

```
2018-09-01 13:43:25.433 Status: LUA: Device Keller changed @ 13:43:25
2018-09-01 13:43:25.433 Status: LUA: Device All Values: 18.6;75;3
2018-09-01 13:43:25.433 Status: LUA: Basement Humidity >= 70 (75)
```

## Lua dzVents

**Event Properties:** basement\_humidity\_monitor\_d, dzVents, Device.

```

1 --~/domoticz/scripts/lua/basement_humidity_monitor_d.lua
2 -----
3 -- Monitor the basement humidity if exceeds threshold and write to Domoticz log.
4 -- project: mysmarthome
5 -- @interpreter: dzVents, Device
6 -- @see: readme.txt
7 -- @author: Robert W.B. Linn
8 -- @version: 20180902
9 -----
10 -- Thresholds
11 -- Set the threshold to monitor the humidity (%RH)
12 local HumidityThreshold = 70;
13
14 -- dzVents
15 return {
16   on = {
17     devices = {
18       'Keller'
19     }
20   },
21   execute = function(domoticz, device)
22     if (device.name == 'Keller' and device.humidity >= HumidityThreshold) then
23       domoticz.log('Basement Humidity >= '..HumidityThreshold.. '('..device.humidity..')', domoticz.LOG_INFO)
24       -- domoticz.notify('Fire', 'The room is on fire', domoticz.PRIORITY_EMERGENCY)
25     end
26   end
27 }
28 }
29 <

```

## Script

```

local HumidityThreshold = 70;
return {
  on = {
    devices = {
      'Keller'
    }
  },
  execute = function(domoticz, device)
    if (device.name == 'Keller' and device.humidity >= HumidityThreshold) then
      domoticz.log('Basement Humidity >= '..HumidityThreshold.. '('..device.humidity..')', domoticz.LOG_INFO)
    end
  end
}

```

The script is stored in:

```
/home/pi/domoticz/scripts/dzVents/generated_scripts/basement_humidity_monitor_d.lua
```

With owner root.

### Domoticz Log

#### *Sample Script – Error*

```

2018-09-01 19:05:55.584 Status: dzVents: Error (2.4.7): error loading module '
basement_humidity_monitor_d' from file '/home/pi/domoticz/scripts/dzVents/generated_scripts/
basement_humidity_monitor_d.lua':
2018-09-01 19:05:55.584 .../generated_scripts/ basement_humidity_monitor_d.lua:12: 'then' expected near
''

```

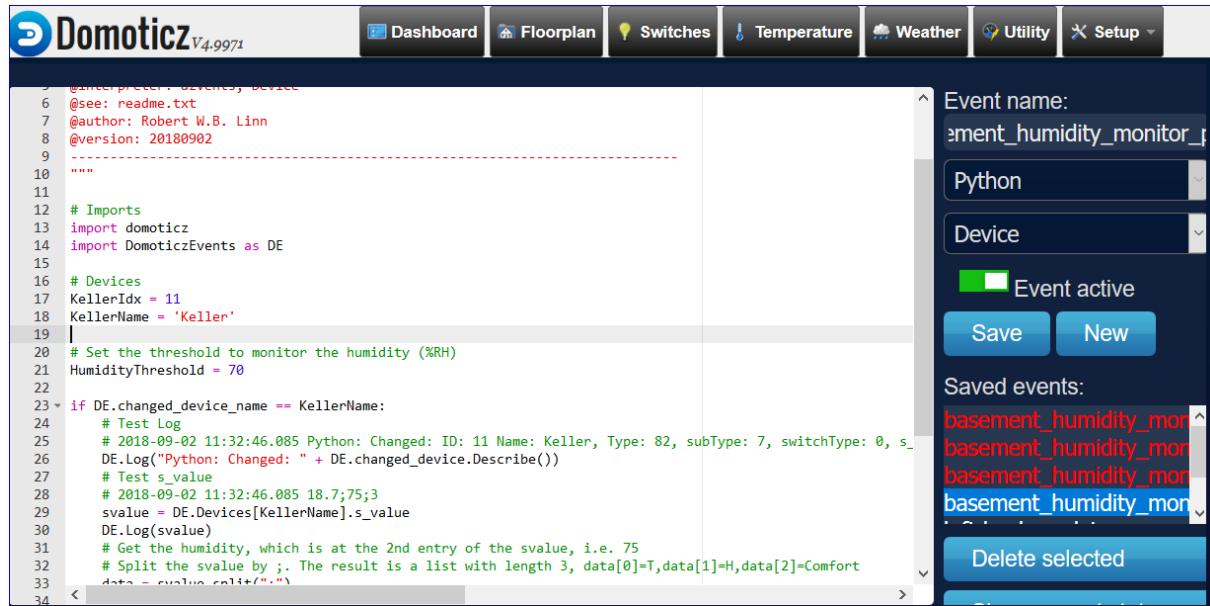
Caused by using double )) prior then.

*Sample Script – OK*

```
2018-09-01 19:09:42.474 Status: dzVents: Write file:  
/home/pi/domoticz/scripts/dzVents/generated_scripts/_basement_humidity_monitor_d.lua  
2018-09-01 19:11:39.449 Status: dzVents: Info: Handling events for: "Keller", value: "18.6;70;1"  
2018-09-01 19:11:39.449 Status: dzVents: Info: ----- Start internal script:  
script_device_basement_humidity_2: Device: "Keller (RFXtrx433e)", Index: 11  
2018-09-01 19:11:39.449 Status: dzVents: Info: Basement Humidity >= 70 (70)  
2018-09-01 19:11:39.449 Status: dzVents: Info: ----- Finished basement_humidity_monitor_d
```

## Python

**Event Properties:** basement\_humidity\_monitor\_p, Python, Device.



## Script

```
"""
Script: basement_humidity_monitor_p
project: AtHome
Monitor the basement humidity if exceeds threshold and write to Domoticz log.
@interpreter: Python, Device
@see: readme.txt
@author: Robert W.B. Linn
@version: 20180902
"""

# Imports
import domoticz
import DomoticzEvents as DE

# Devices
KellerIdx = 11
KellerName = 'Keller'

# Set the threshold to monitor the humidity (%RH)
HumidityThreshold = 70

if DE.changed_device_name == KellerName:
    # Test Log
    # 2018-09-02 11:32:46.085 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType: 0, s_
    DE.Log("Python: Changed: " + DE.changed_device.Describe())
    # Test s_value
    # 2018-09-02 11:32:46.085 18.7;75;3
    svalue = DE.Devices[KellerName].s_value
    DE.Log(svalue)
    # Get the humidity, which is at the 2nd entry of the svalue, i.e. 75
    # Split the svalue by ;. The result is a list with length 3, data[0]=T,data[1]=H,data[2]=Comfort
    data = svalue.split(";")
    # Log all 3 entries
    # for temp in data:
    #     DE.Log(temp)
    # Get the :
    if len(data) == 3:
        Humidity = int(data[1])

```

```
if Humidity >= HumidityThreshold:  
    # Log that the humidity is above threshold. The log string uses new string formatting  
    DE.Log('Basement Humidity >= {}%RH ({}%RH)'.format(HumidityThreshold, Humidity))
```

## Domoticz Log

### *Sample Script – Error*

```
...  
2018-09-02 11:58:34.884 Status: EventSystem: Script event triggered: luftdruck_update  
2018-09-02 11:58:34.107 Error: EventSystem: Failed to execute python event script  
"basement_humidity_monitor_p"  
2018-09-02 11:58:34.108 Error: EventSystem: Traceback (most recent call last):  
2018-09-02 11:58:34.108 Error: EventSystem: File "<string>", line 41, in <module>  
2018-09-02 11:58:34.108 Error: EventSystem: IndexError: tuple index out of range
```

Caused by using wrong string format tuples, i.e. {1} ({2}) instead of 0,1 or leave blank {} ({}).

### *Sample Script – OK*

```
2018-09-02 12:01:26.253 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType: 0,  
s_value: 18.7;74;3, n_value: 0, n_value_string: 18.7;74;3, last_update_string: 2018-09-02 12:01:26  
2018-09-02 12:01:26.253 18.7;74;3  
2018-09-02 12:01:26.253 Basement Humidity >= 70%RH (74%RH)
```

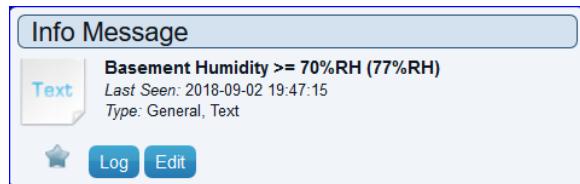
## Python with Device Update

### Extending the previous Python script

Update the Virtual Sensor “Info Message” (IDX=52) (Type: General, Text) with the message logged, if the humidity exceeds the threshold.

52	VirtualSensors	00082051	1	Info Message	General	Text	Basement Humidity >= 70%RH (78%RH)
----	----------------	----------	---	--------------	---------	------	------------------------------------

### Result



**Event Properties:** basement\_humidity\_monitor2\_p, Python, Device.

### Script

```
"""
script: basement_humidity_monitor2_p
Monitor the basement humidity (device=Keller) if exceeds threshold, write to Domoticz log, and update
the virtual sensor text (device=Info Message)
interpreter: Python, Device
see: readme.txt
author: Robert W.B. Linn
version: 20180902
-----
"""

# Imports
import domoticz
import DomoticzEvents as DE
from urllib.request import Request, urlopen
from urllib.error import URLError, HTTPError

# Define the domoticz system ip with port!
domoticzserver="localhost:8080"

# Devices
KellerIdx = 11
KellerName = 'Keller'
TextInfoIdx = 52
TextInfoName = 'Info Message'

# Set the threshold to monitor the humidity (%RH)
HumidityThreshold = 70

def domoticzrequest (url):
    """Send update request to domoticz
    :url: Domoticz JSON/API url
    :result: Result is a json object
    """
    req = Request(url)
    try:
        response = urlopen(req)
    except HTTPError as e:
        # print('Server couldn\'t fulfill the request.')
        # print('Error code: ', e.code)
        return "Error fulfilling the request." + e.code
    except URLError as e:
        # print('Failed to reach a server.')
        # print('Reason: ', e.reason)

```

```

        return "Error reaching the server." + e.code
    else:
        # everything is fine
        return response.read().decode('utf-8')

if DE.changed_device_name == KellerName:
    # Test Log
    # 2018-09-02 11:32:46.085 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType: 0, s_value: 18.7;75;3, n_value: 0, n_value_string: 18.7;75;3, last_update_string: 2018-09-02 11:32:46
    DE.Log("Python: Changed: " + DE.changed_device.Describe())
    # Test s_value
    # 2018-09-02 11:32:46.085 18.7;75;3
    svalue = DE.Devices[KellerName].s_value
    DE.Log(svalue)
    # Get the humidity, which is at the 2nd entry of the svalue, i.e. 75
    # Split the svalue by ;. The result is a list with length 3, data[0]=T,data[1]=H,data[2]=Comfort
    data = svalue.split(";")
    # Log all 3 entries
    # for temp in data:
    #     DE.Log(temp)
    # Get the humidity at pos 2 which is index 1 :
    if len(data) == 3:
        Humidity = int(data[1])
        if Humidity >= HumidityThreshold:
            # Log that the humidity is above threshold. The log string uses new string formatting
            svalueinfo = 'Basement Humidity >= {}%RH ({})'.format(HumidityThreshold, Humidity)
            DE.Log(svalueinfo)

            # Update the Virtual Sensor Info_Message
            # This is not working - not sure why, might be only in plugins
            # DE.Command(TextInfoName, svalueinfo)

            # Require escaping the message string including spaces
            svalueinfo = urllib.parse.quote_plus(svalueinfo)
            # Build the url. Example:
            #
http://localhost:8080/json.htm?type=command&param=udevice&idx=52&nvalue=0&svalue=Basement+Humidity+%3E%3D+70%25RH+%2877%25RH%29
        url = "http://" + domoticzserver + "/json.htm?type=command&param=udevice&idx=" +
str(TextInfoIdx) + "&nvalue=0&svalue=" + svalueinfo
        # DE.Log(url)
        ret = domoticzrequest(url)
        # Log the return . which is a JSON string {"status" : "OK","title" : "Update Device"}
        # DE.Log(ret)

```

## Domoticz Log

### *Sample Script – OK*

```

2018-09-02 19:53:42.963 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType: 0,
s_value: 18.8;78;3, n_value: 0, n_value_string: 18.8;78;3, last_update_string: 2018-09-02 19:53:42
2018-09-02 19:53:42.963 18.8;78;3
2018-09-02 19:53:42.963 Basement Humidity >= 70%RH (78%RH)

```

# Events dzVents Lua

The event system Domoticz Easy Events, *dzVents*, is the next generation of Domoticz Events using [Lua](#), i.e. *dzVents* is a *Lua-based event scripting framework*.

As a general point, be very secure in writing the scripts, also when making changes, i.e.

- ensure all variables have the same name
- do not mix local variables with the same name
- do not use the same variables across scripts which are not local etc.
- the same applies for functions

## Device Properties

To analyse the properties of a device, use `device.dump()`, then select from the list.

## Script

This example checks if the defined devices (see idx of the devices) have changed, then dumps the properties but also shows the device state property.

```
-- Idx of the devices
local IDXMEMORYUSAGE = 1;
local IDXHDDUSAGE = 3;
local IDXTEMPERATURE = 4;
local IDXCONTROLMSG = 52;

return {
  on = {
    devices = {
      IDXMEMORYUSAGE,
      IDXHDDUSAGE,
      IDXIDXTTEMPERATURE
    }
  },
  execute = function(domoticz, device)
    domoticz.log('RPi Monitor Device ' .. device.name .. ',' .. device.idx .. ' was changed:' .. device.state .. '/' .. device.rawData[1], domoticz.LOG_INFO)
  end
}
```

## Domoticz Log

```
2018-09-05 10:01:10.305 Status: dzVents: Info: Handling events for: "RPi Memory Usage", value: "24.20"
2018-09-05 10:01:10.305 Status: dzVents: Info: ----- Start internal script: rpi_monitor: Device: "RPi
Memory Usage (Motherboard)", Index: 1
2018-09-05 10:01:10.305 Status: dzVents: Info: RPi Monitor Device RPi Memory Usage,1 was
changed:24.20/24.20
2018-09-05 10:01:10.305 Status: dzVents: Info: ----- Finished rpi_monitor
```

# Property List

Example properties for device RPi Memory Usage:

ruleIsAtSunrise() wday: 4 isUTC: false ruleIsAfterCivilTwilightEnd() hour: 9 ruleIsAfterCivilTwilightStart() ruleIsBeforeCivilTwilightStart() current: isdst: true min: 49 year: 2018 day: 5 yday: 248 sec: 9 month: 9 wday: 4 hour: 9 secondsAgo: 80 ruleIsOnDate() milliSeconds: 0 millisecondsAgo: 80896 ruleIsInWeek() dDate: 1536133669 ruleIsAtDayTime() ruleIsAfterSunrise() day: 5 isdst: true min: 47 minutesAgo: 1	utils: fileExists() LOG_MODULE_EXEC_INFO: 2 LOG_ERROR: 1 toJSON() log() rgbToHSB() LOG_INFO: 3 osExecute() urlEncode() print() fromJSON() LOG_FORCE: 0.5 LOG_DEBUG: 4 updatePressure() deviceId: 0000044C updateRain() close() updateP1() _adapters: 1: Percentage device adapter disarm() startPlaylist() changed: true setVolume() stop() _data: deviceID: 0000044C data: unit: 1 icon: hardware hardwareTypeValue: 23 _state: 24.15 _nValue: 0 hardwareID: 1	hardwareName: Motherboard protected: false hardwareType: Motherboard sensors lastUpdate: 2018-09-05 09:47:49 subType: Percentage baseType: device changed: true batteryLevel: 255 switchType: On/Off switchTypeValue: 0 deviceType: General rawData: 1: 24.15 id: 1 signalLevel: 12 description: lastLevel: 0 name: RPi Memory Usage timedOut: false switchType: On/Off setWhiteMode() deviceType: General switchTypeValue: 0 protected: false isScene: false state: 24.15 hardwareTypeValue: 23 idx: 1 isTimer: false signalLevel: 12 bState: false isVariable: false hardwareType: Motherboard sensors
--	---	--

## External Modules

### Purpose

To use external modules in dzVents script events created with the Domoticz Events editor.

#### Note

As progressing, the external modules as described here, are replaced by a dzVents script “**global\_data**” (containing shared helper functions), maintained using the Domoticz Event Editor.

This is the recommend way to use common functions.

### Example

Update the virtual device named “Alert Message” (Type General, SubType Alert, IDX=55) by using a function defined in external module “msgbox.lua”.

```
local message = device.name .. ' switched OFF ' -- .. msg.isnowdatetime(domoticz)
msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
```

## Module Access

The module “msgbox.lua” MUST be in folder:

```
/home/pi/domoticz/scripts/dzVents/scripts
```

To access functions from the module, use the syntax **msgbox.functionname**, when the module is defined like:

```
local msgbox = require('msgbox')
```

### Example (screenshot from WinSCP)

/home/pi/domoticz/scripts/dzVents/scripts/*.*			
Name	Size	Changed	
..		08.10.2017 17:17:48	
msgbox.lua	2 KB	06.09.2018 13:15:11	
README.md	1 KB	29.08.2017 16:35:10	

## Module Script

The script defines date & time functions and functions to update a text device & alert device. The idx of the text device is set in the external module, for the alert device as a user variable. This is to show both ways – best would be to have both device idx as user variables to be flexible in assigning devices.

A domoticz object is required to access properties & methods.

```
msgbox.lua: /home/pi/domoticz/scripts/dzVents/scripts
```

```
-- msgbox.lua
-- Update the virtual devices to display a control (Text Sensor) or alert message (Alert Sensor).
-- Requires a user variable IDX_ALERTMSG for the Alert Sensor
-- Robert W.B. Linn
-- 20180906

local msgbox = {}

local IDX_CONTROLMSG = 52

-- Get the current date & time from the domoticz instance with time object
-- Return datetime now, i.e. 2018-09-06 09:09:00
function msgbox.isnowdatetime(domoticz)
    return domoticz.time.rawDate .. ' ' .. domoticz.time.rawTime
end

-- Get the current date from the domoticz instance with time object
-- Return date now, i.e. 2018-09-06
function msgbox.isnowdate(domoticz)
    return domoticz.time.rawDate
end

-- Get the current time from the domoticz instance with time object
-- Return time now, i.e. 09:09:00
function msgbox.isnowtime(domoticz)
    return domoticz.time.rawTime
end

-- Update the alert message with level.
function msgbox.alertmsg(domoticz, level, msg)
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)
end

-- Update the control message.
function msgbox.controlmsg(domoticz, msg)
    domoticz.devices(IDX_CONTROLMSG).updateText(msg)
end

return msgbox
```

## dzVents Script

The Lua script “hue\_control2.lua”, created with the Domoticz Event editor, references the external module “msgbox.lua” as msgbox.

If the script is set to enabled, the scripts is stored in folder

```
/home/pi/domoticz/scripts/dzVents/generated_scripts
```

```
-- Module msgbox.lua: /home/pi/domoticz/scripts/dzVents/scripts
-- Access function using msgbox.functionname
local msgbox = require('msgbox')

-- Idx of the devices
local IDX_HUE_MAKELAB = 118

return {
    -- active = true,
    on = {
        devices = {
            IDX_HUE_MAKELAB
        }
    },
    execute = function(domoticz, device)
        if (device.state == 'On') then
            device.switchOff()
        local message = device.name .. ' = OFF ' -- .. msg.isnowdatetime(domoticz)
        msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
        msgbox.controlmsg(domoticz, message)
        domoticz.log('OK', domoticz.LOG_INFO)
        end
    end
}
```

## Folder Scripts

Scripts located in folder

```
/home/pi/domoticz/scripts/dzVents/scripts
```

are executed every minute if these are not modules.

```
-- Lua script executed every minute in the dzVents scripts folder
print(os.date("today is %A, at %x"))
```

## Domoticz Log

```
-- 2018-09-10 12:15:00.750 Status: dzVents: today is Monday, at 09/10/18
```

# Shared Helper Functions

## Purpose

To define common functions used by dzVents Lua script events i.e. my automation scripts.  
The functions are defined, using the Domoticz Event Editor, in the script “global\_data”.

**This is the recommend way to use common functions.**

## Example Functions

```
return {

    -- global helper functions
    helpers = {

        -- Idx of devices used
        IDX_ALERTMSG = 55,
        IDX_CONTROLMSG = 52,

        -- Update the alert message with level and text
        alertmsg = function(domoticz, level, msg)
            domoticz.devices(domoticz.helpers.IDX_ALERTMSG).updateAlertSensor(level, msg)
        end,

        -- Update the control message with text
        controlmsg = function(domoticz, msg)
            domoticz.devices(domoticz.helpers.IDX_CONTROLMSG).updateText(msg)
        end

    }

    -- end return
}
```

## Usage

```
-- set the alert message
Local message = 'This is an alert message'
domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_ORANGE, message)
```

## Hints

### Clear Device Log

To clear a device log via dzVents Lua Script, use

```
IDXSTATUS = 106          -- text status
domoticz.openURL('http://localhost:8080/json.htm?type=command&param=clearlightlog&idx=' .. IDXSTATUS)
```

## Fast Event Less One Minute

The Domoticz Event System has a restriction of timer updates less than one minute.

### Challenge

How to create an automation event with action(s) being executed with an interval less than one minute or immediate?

### Solutions

1. [Timer Event](#) with a For Loop, acting as a subtimer handled by the function openURL() with extra option delay via the function afterSec()
2. [Device Event](#) with function openURL() triggering another openURL() with delay via the function afterSec()  
This solution is used by the function Music Player.
3. [MQTT Message Trigger](#) from other applications, i.e. Node-RED
4. [HTTP API Trigger](#) from other applications, i.e. Python, Java, triggering a Domoticz Custom Event with Data as JSON key:value pairs

### Timer Event Trigger

This solution uses a Timer as Trigger with a For Loop within the Timer event.

The For Loop runs the function openURL with a delay by the function afterSec(seconds).

### Pseudocode

```
DELAYSECONDS = 20

Timer triggered every minute
Timer event For Loop every 20 seconds
Run openURL(ADDLOGMESSAGE, CALLBACK).afterSec(DELAYSECONDS)

Handle CALLBACK
Any action, i.e. Update device
```

### Automation Event Example

(without Domoticz Log entries)

```
-- explore_fastevent_timer.dzevents
local URL_DOMOTICZ = "http://domoticz-ip:8080"
local CALLBACK_TIMER = 'EXPLOREFASTEVENTTIMER'
local TIMERTRIGGERSECS = 60          -- timer runs every minute (NN seconds)
local TIMERLOOPINTERVALSECS = 20      -- subtimer for loop every NN seconds
-- the subtimer runs 3 times within a minute
local subtimerCounter = 0
math.randomseed(os.time())

return
{
  on = { timer = { 'every minute' }, httpResponses = { CALLBACK_TIMER }, },
  execute = function(domoticz, item)
    if item.isTimer then
      local url = string.format( "%s/json.htm?type=command&param=addlogmessage&message=trigger_%s",
        URL_DOMOTICZ, CALLBACK_TIMER)
      subtimerCounter = 0
      for seconds = 0, TIMERTRIGGERSECS - TIMERLOOPINTERVALSECS, TIMERLOOPINTERVALSECS do
        subtimerCounter = subtimerCounter + 1
        domoticz.openURL({url = url, callback = CALLBACK_TIMER }).afterSec(seconds)
      end
    end
  end
}
```

```

end

if (item.isHTTPResponse) then
    -- Final action to take - hers just logging a random value
    domoticz.log(string.format("New value = %d", math.random(0,9)))
end
end
}

```

## Domoticz Log

The log shows the event triggered at 17:09:00 and running 3 times the subtimer logging the random values 2, 9, 7. The next event starts at 17:10:00.

```

2020-06-03 17:09:00.303 Status: dzVents: Info: ----- Start internal script: explore_fastevent_timer:, trigger: "every minute"
2020-06-03 17:09:00.303 Status: dzVents: Info: ***TIMER TRIGGER START
2020-06-03 17:09:00.303 Status: dzVents: Info: Subtimer trigger: 1 = 0
2020-06-03 17:09:00.304 Status: dzVents: Info: Subtimer trigger: 2 = 20
2020-06-03 17:09:00.304 Status: dzVents: Info: Subtimer trigger: 3 = 40
2020-06-03 17:09:00.304 Status: dzVents: Info: ***TIMER TRIGGER END
2020-06-03 17:09:00.305 Status: dzVents: Info: ----- Finished exploring_fastevent_timer
2020-06-03 17:09:00.306 Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-06-03 17:09:00.330 Status: trigger_EXPLOREFASTEVENTTIMER
2020-06-03 17:09:00.458 Status: dzVents: Info: Handling httpResponse-events for: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:00.458 Status: dzVents: Info: ----- Start internal script: explore_fastevent_timer: HTTPResponse: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:00.459 Status: dzVents: Info: ***HTTP RESPONSE EXPLOREFASTEVENTTIMER Status Code = 200
2020-06-03 17:09:00.459 Status: dzVents: Info: New value = 2
2020-06-03 17:09:00.459 Status: dzVents: Info: ----- Finished explore_fastevent_timer
2020-06-03 17:09:20.338 Status: trigger_EXPLOREFASTEVENTTIMER
2020-06-03 17:09:20.470 Status: dzVents: Info: Handling httpResponse-events for: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:20.470 Status: dzVents: Info: ----- Start internal script: explore_fastevent_timer: HTTPResponse: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:20.470 Status: dzVents: Info: ***HTTP RESPONSE EXPLOREFASTEVENTTIMER Status Code = 200
2020-06-03 17:09:20.470 Status: dzVents: Info: New value = 9
2020-06-03 17:09:20.470 Status: dzVents: Info: ----- Finished explore_fastevent_timer
2020-06-03 17:09:40.308 Status: trigger_EXPLOREFASTEVENTTIMER
2020-06-03 17:09:40.432 Status: dzVents: Info: Handling httpResponse-events for: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:40.432 Status: dzVents: Info: ----- Start internal script: explore_fastevent_timer: HTTPResponse: "EXPLOREFASTEVENTTIMER"
2020-06-03 17:09:40.432 Status: dzVents: Info: ***HTTP RESPONSE EXPLOREFASTEVENTTIMER Status Code = 200
2020-06-03 17:09:40.433 Status: dzVents: Info: New value = 7
2020-06-03 17:09:40.433 Status: dzVents: Info: ----- Finished explore_fastevent_timer
***THE NEXT EVENT ***
2020-06-03 17:10:00.317 Status: dzVents: Info: ----- Start internal script: explore_fastevent_timer:, trigger: "every minute"

```

## Device Event Trigger

Next the pseudo code example of updating devices every 10 seconds.

This is taken from the function [Music Player](#), which updates the Domoticz (virtual) devices playing & progress every 10 seconds. The chapter “Music Player” has more details.

### Pseudocode

```
DELAYSECONDS = 10

Trigger openURL(GETITEM, CALLBACK_GETITEM)

Handle CALLBACK_GETITEM
    Run openURL(GETPROPERTIES, CALLBACK_GETPROPERTIES)

Handle CALLBACK_GETPROPERTIES
    Run openURL(GETITEM, CALLBACK_GETITEM).afterSec(DELAYSECONDS)
```

#### Notes

1. The trigger to start device updates is by switching the device “Remote Control” On,
2. The change of the switch device triggers the first openURL for the JSON-RPC API GETITEM,
3. The GETITEM callback runs openURL for the JSON-RPC API GETPROPERTIES,
4. The GETPROPERTIES callback runs GETITEM again, but after a delay, set by the constant DELAYSECONDS to 10 seconds,
5. After the delay, the loop starts again from the GETITEM callback (3).

## MQTT Message Custom Event Trigger

Trigger, via MQTT an action from another application or system, i.e. RaspberryMatic or Node-RED which is handled by a Domoticz Custom Event.

With this option it is possible to send MQTT messages to Domoticz which are picked up immediately by dzVents scripts.

This avoids using a timer to pull data but use custom events to receive pushed data (packed in the payload key data).

### Example

Node-RED sending in regular intervals, an MQTT message with topic & payload to Domoticz  
The “data” key contains the properties “idx” and “value” to be used by Domoticz.

```
Topic: domoticz/in
Payload: {"command":"customevent","event":"MyCustomEvent","data":{"idx":29,"value":9} }
```

### Automation Event Example

```
-- explore_fastevent_custom.dzvents

return {
  on = {
    customEvents = { 'explore_fastevent_custom' },
  },
  execute = function(domoticz, item)
    -- Log the trigger and the mqtt message key data (which is a string)
    domoticz.log(string.format("%s:%s", item.trigger, item.data))
    -- Convert the data string, as a JSON object to a Lua table
    local data = domoticz.utils.fromJSON(item.data)
    -- Log the key:value pairs from the passed data
    domoticz.log(string.format("IDX:%s, Value:%d", data.idx, data.value))
  end
}
```

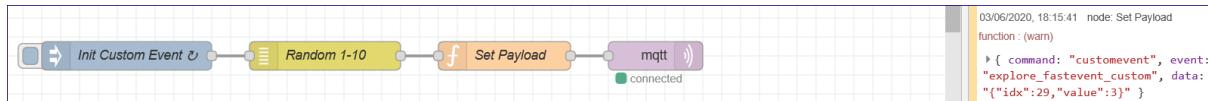
### Domoticz Log

The log shows the event triggered every 10 seconds, 18:18:52, 18:19:02 ...

```
2020-06-03 18:18:52.644 MQTT: Topic: domoticz/in, Message:
{"command":"customevent","event":"explore_fastevent_custom","data":{"idx":29,"value":1}}
2020-06-03 18:18:52.776 Status: dzVents: Info: Handling Domoticz custom event for:
"explore_fastevent_custom"
2020-06-03 18:18:52.776 Status: dzVents: Info: ----- Start internal script: explore_fastevent_custom:
Custom event: "explore_fastevent_custom"
2020-06-03 18:18:52.777 Status: dzVents: Info: explore_fastevent_custom:{"idx":29,"value":1}
2020-06-03 18:18:52.777 Status: dzVents: Info: IDX:29, Value:1
2020-06-03 18:18:52.777 Status: dzVents: Info: ----- Finished explore_fastevent_custom
2020-06-03 18:19:02.655 MQTT: Topic: domoticz/in, Message:
{"command":"customevent","event":"explore_fastevent_custom","data":{"idx":29,"value":10}}
2020-06-03 18:19:02.785 Status: dzVents: Info: Handling Domoticz custom event for:
"explore_fastevent_custom"
2020-06-03 18:19:02.786 Status: dzVents: Info: ----- Start internal script: explore_fastevent_custom:
Custom event: "explore_fastevent_custom"
2020-06-03 18:19:02.786 Status: dzVents: Info: explore_fastevent_custom:{"idx":29,"value":10}
2020-06-03 18:19:02.786 Status: dzVents: Info: IDX:29, Value:10
2020-06-03 18:19:02.786 Status: dzVents: Info: ----- Finished explore_fastevent_custom
... AND MORE ...
```

## Node-RED Flow

Source: explore\_fastevent\_custom.flow



### Nodes

Inject	Trigger sending MQTT message every 10 seconds
random	generate a number between 1-10
function	<p>create the msg with topic and payload</p> <pre> var vvalue = msg.payload; msg.topic = "domoticz/in"; msg.payload = {     "command": "customevent",     "event": "explore_fastevent_custom",     "data": "{\"idx\":29,\"value\":\"" + vvalue + "\"}" }; node.warn(msg.payload); return msg; </pre>
mqtt out	send the msg to the MQTT broker (Domoticz)

## HTTP API Request Custom Event Trigger

Trigger, via a HTTP API request an action, which is handled by a Domoticz Custom Event (dzVents script).

With this option it is possible to send HTTP API requests to Domoticz which are picked up immediately by Custom Events.

This avoids using a timer to pull data but use custom events to receive pushed data.

### Solution Data from Parameter

In below example, a Python script triggers every 5 seconds a HTTP API request to Domoticz with parameter customevent and data as JSON string with properties level & text to update a Domoticz virtual sensor type General, SubType Alert.

The Python script runs on the same system as Domoticz.

#### *Cron Reboot*

The Python script could be included in a cron job to get started as a process at system boot.

Edit crontab

```
$ crontab -e
```

Add the line

```
@reboot python /Path/To/Python_Script.py >>/Path/To/Log/Python_Script.log 2>&1
```

#### *Cron Job*

Initially been thinking of using a Bash script via a *cron* job, BUT *cron* has a resolution of one minute. So, the same as the Domoticz Event system.

With some tricks it would be possible to trigger a job less than one minute, by for example running 12 cron jobs every 5 seconds

```
* * * * * for i in {1..12}; do Path/To/Python_Script.py & sleep 5; done
```

Or as an alternative, use *watch*

```
watch -n5 -x /Path/To/Python_Script.py
```

#### *Python Libraries*

There are many Python libraries available to access sensors or control actuators.

With this solution, it is possible to create own solutions interacting with Domoticz fast via HTTP API.

## Example updating Domoticz Alert Sensor every 10 seconds with random Alert Level

### Pseudo Code

```

Python
DELAYSECONDS = 5
Define JSON data
Submit, every DELAYSECONDS, a HTTP API request to Domoticz with param=customevent and data JSON

dzVents Lua
Trigger customEvent
Parse the JSON data to Lua table
Act, i.e. update an alert sensor level & text

```

### Python Script

The Python script runs from the Command-Line Interface (CLI)

```

# explore_fastevent_http.py
## domoticz-workbook - function explore_events_dzvents
## Submit in regular intervals less than 1 minute data to domoticz to update an alert sensor.
## A domoticz custom event (explore_fastevent_http) handles the incoming data defined as json by
## parsing the properties level & text and update the alert sensor.
## The data parameter is a json string. Example: {"level": 4, "text": "Alert Level set to 4"}
##
## CLI Run Example with output:
##   python3 explore_fastevent_http.py
##   Counter:1,
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 4,
"text": "Alert Level set to 4"}
##   OK Custom Event
##   Counter:2,
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 4,
"text": "Alert Level set to 4"}
##   OK Custom Event
##   Counter:3,
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 4,
"text": "Alert Level set to 4"}
##   OK Custom Event
##   Done
##
## The data received in the domoticz custom event as debug:
## ...Info: SETALERT: [{"isJSON":true, "message":"",
## "isXML":false, "data":{"level": 4, "text": "Alert Level set to 4"}}, {"dump":function, "type":"customEvent", "isHTTPResponse":false,
## "isCustomEvent":true, "isHardware":false, "isSystem":false, "json":{["text"]:"Alert Level set
## to 4", ["level"]=4}, "isTimer":false, "isVariable":false, "isSecurity":false,
## "customEvent"]="setalert", "trigger"]="setalert", "status"]="info", "isDevice":false,
## "isScene":false, "isGroup":false}
## ...Info: SETALERT: Level:4,Text:Alert Level set to 4
##
## Dependencies: explore_fastevent_http.dzvents
## 20201007 rwbl

import json
import requests
import time

# generate random integer values used to set the alert level between 1 - 4
from random import seed
from random import randint
# seed random number generator
seed(1)

## Domoticz ip is local with default port
domoticz_ip = "http://127.0.0.1"
domoticz_port = 8080
custom_event = "setalert"
## Alert device level and text - see function send_request
alert_level = randint(1, 4)
alert_text = "Alert Level set to {}".format(alert_level)
## Loop counter with the delay in seconds between the loop
counter = 0
delay = 10

```

```
## Define the json data with keys level (integer) and text (string)
def set_data(level, text):
    data = {}
    data['level'] = level
    data['text'] = text
    return json.dumps(data)

## Send the HTTP API request to domoticz to update the alert device
## In the custom event the idx of the alert device is defined
def send_request():
    # increase the counter
    global counter
    counter = counter + 1
    # define the json data for the custom event request
    #
    alert_level = randint(1, 4)
    alert_text = "Alert Level set to {}".format(alert_level)
    json_data = set_data(alert_level, alert_text)
    # define the url to signal the domoticz custom event
    url = "{}:{}/.json.htm?type=command&param=customevent&event={}&data={}".format(domoticz_ip,
domoticz_port, custom_event, json_data)
    print("Counter:{}, URL:{}".format(counter, url))
    # submit the url
    try:
        r = requests.get(url, timeout=3)
        r.raise_for_status()                                     # 200
        content_json = json.loads(r.content)
        # print(content_json)                                     # {'status': 'OK', 'title': 'Custom
Event'}
        print(content_json['status'], content_json['title'])   # OK Custom Event
    except requests.exceptions.HTTPError as errh:
        print ("Http Error:",errh)
    except requests.exceptions.ConnectionError as errc:
        print ("Error Connecting:",errc)
    except requests.exceptions.Timeout as errt:
        print ("Timeout Error:",errt)
    except requests.exceptions.RequestException as err:
        print ("OOps: Something Else",err)

## main
if __name__ == '__main__':
    while True:
        send_request()
        if counter == 3:
            print("Done\n")
            quit()
        time.sleep(delay)
```

## Custom Event

```
--[[[  
    fastevent_http.dzvents  
    Trigger: customEvent  
    To update in regular intervals less than 1 minute, the level & text of a virtual alert sensor.  
    The custom event receives data from a Python script running on the Raspberry Pi CLI.  
    The custom data is defined as JSON, i.e. {"level": 4, "text": "Alert Level set to 4"}.  
    The data is converted to a Lua table and used to update the alert device level & text.  
    2020929 rwbl  
]]--  
  
-- Idx of the alert sensor  
local IDX_ALERT = 26  
  
return  
{  
    on =  
    {  
        customEvents =  
        {  
            'setalert',  
        },  
    },  
  
    logging =  
    {  
        level = domoticz.LOG_DEBUG,  
        marker = 'SETALERT',  
    },  
  
    execute = function(domoticz, item)  
        domoticz.log(item)  
        -- Convert item data to lua table  
        local data = domoticz.utils.fromJSON(item.data)  
        domoticz.log(("Level:%d,Text:%s"):format(data.level, data.text))  
        -- Update the alert device  
        domoticz.devices(IDX_ALERT).updateAlertSensor(data.level, data.text)  
    end  
}
```

## Example running Python Script from the CLI

The delay is set to every 10 seconds.

The output of the script is logged to a file x.log and viewed with cat.

```
$python3 explore_fastevent_http.py >> x.log 2>&1  
  
$cat x.log  
Counter:1,  
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 1,  
"text": "Alert Level set to 1"}  
OK Custom Event  
Counter:2,  
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 3,  
"text": "Alert Level set to 3"}  
OK Custom Event  
Counter:3,  
URL:http://127.0.0.1:8080/json.htm?type=command&param=customevent&event=setalert&data={"level": 1,  
"text": "Alert Level set to 1"}  
OK Custom Event  
Done
```

## Domoticz Log

The log lists 3 custom events with Alert levels 1, 3, 1 and debug information as set by the dzVents logging level domoticz.LOG\_DEBUG.

```
2020-10-09 09:12:34.678 Status: dzVents: Info: Handling Domoticz custom event for: "setalert"  
2020-10-09 09:12:34.679 Status: dzVents: Info: SETALERT: ----- Start internal script: fastevent_http:  
Custom event: "setalert"  
2020-10-09 09:12:34.679 Status: dzVents: Info: SETALERT: {[{"isHardware":false,  
["isHTTPResponse":false, ["isXML":false, ["data"]={"level": 1, "text": "Alert Level set to 1"}]}],
```

```
["isGroup"]=false, ["isCustomEvent"]=true, ["isTimer"]=false, ["isSystem"]=false, ["isSecurity"]=false,  
["isScene"]=false, ["isVariable"]=false, ["trigger"]="setalert", ["isDevice"]=false,  
["customEvent"]="setalert", ["type"]="customEvent", ["dump"]=function, ["isJSON"]=true,  
["json"]=[{"level":1, "text":"Alert Level set to 1"}, {"status"}="info", ["message"]=""}  
2020-10-09 09:12:34.679 Status: dzVents: Info: SETALERT: Level:1,Text:Alert Level set to 1  
2020-10-09 09:12:34.690 Status: dzVents: Debug: SETALERT: Processing device-adapter for Alert Message:  
Alert sensor adapter  
2020-10-09 09:12:34.691 Status: dzVents: Info: SETALERT: ----- Finished fastevent_http  
2020-10-09 09:12:34.691 Status: EventSystem: Script event triggered:  
/home/pi/domoticz/dzVents/runtime/dzVents.lua  
2020-10-09 09:12:44.745 Status: dzVents: Info: Handling Domoticz custom event for: "setalert"  
2020-10-09 09:12:44.746 Status: dzVents: Info: SETALERT: ----- Start internal script: fastevent_http:  
Custom event: "setalert"  
2020-10-09 09:12:44.746 Status: dzVents: Info: SETALERT: {[{"message}="", ["isSystem"]=false,  
["customEvent"]="setalert", ["isScene"]=false, ["isVariable"]=false, ["dump"]=function,  
["status"]="info", ["isTimer"]=false, ["json"]=[{"text"}="Alert Level set to 3", ["level"]=3},  
["isCustomEvent"]=true, ["isHardware"]=false, ["data"]={"level": 3, "text": "Alert Level set to 3"},  
["isGroup"]=false, ["isXML"]=false, ["isDevice"]=false, ["trigger"]="setalert",  
["isHTTPResponse"]=false, ["isSecurity"]=false, ["isJSON"]=true, ["type"]="customEvent"}  
2020-10-09 09:12:44.746 Status: dzVents: Info: SETALERT: Level:3,Text:Alert Level set to 3  
2020-10-09 09:12:44.758 Status: dzVents: Debug: SETALERT: Processing device-adapter for Alert Message:  
Alert sensor adapter  
2020-10-09 09:12:44.758 Status: dzVents: Info: SETALERT: ----- Finished fastevent_http  
2020-10-09 09:12:44.758 Status: EventSystem: Script event triggered:  
/home/pi/domoticz/dzVents/runtime/dzVents.lua  
2020-10-09 09:12:54.777 Status: dzVents: Info: Handling Domoticz custom event for: "setalert"  
2020-10-09 09:12:54.777 Status: dzVents: Info: SETALERT: ----- Start internal script: fastevent_http:  
Custom event: "setalert"  
2020-10-09 09:12:54.777 Status: dzVents: Info: SETALERT: {[{"isTimer}=false, ["isXML"]=false,  
["message"]="", ["isCustomEvent"]=true, ["isDevice"]=false, ["isGroup"]=false, ["status"]="info",  
["customEvent"]="setalert", ["json"]=[{"level":1, "text"}="Alert Level set to 1"},  
["type"]="customEvent", ["isScene"]=false, ["isVariable"]=false, ["data"]={"level": 1, "text": "Alert  
Level set to 1"}, ["isHTTPResponse"]=false, ["dump"]=function, ["isJSON"]=true, ["isHardware"]=false,  
["trigger"]="setalert", ["isSystem"]=false, ["isSecurity"]=false}  
2020-10-09 09:12:54.777 Status: dzVents: Info: SETALERT: Level:1,Text:Alert Level set to 1  
2020-10-09 09:12:54.790 Status: dzVents: Debug: SETALERT: Processing device-adapter for Alert Message:  
Alert sensor adapter  
2020-10-09 09:12:54.790 Status: dzVents: Info: SETALERT: ----- Finished fastevent_http  
2020-10-09 09:12:54.790 Status: EventSystem: Script event triggered:  
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

## Bash Script

Example bash script to update the Alert Sensor level & text once.

To regular update the Alert Sensor in less one minute, the bash script could be triggered by a Python script or Node-RED flow.

```
#!/bin/bash
# explore_fastevent_http.sh
# To update the level & text of a virtual alert sensor.
# The custom event receives data from a Python script running on the Raspberry Pi CLI.
# The custom data is defined as JSON, i.e. {"level": 4, "text": "Alert Level set to 4"}.
# The data is converted to a Lua table and used to update the alert device level & text.
# 20201007 rwbl

#
# Define vars
#
## Domoticz ip running local with default port
domoticz_ip=127.0.0.1
domoticz_port=8080
## Alert level and text. The text special characters must be escaped!
level=4
text="The%20alert%20level%20is%20set%20to%20$level"
## Define the JSON string without {} as these are set in the url parameter
json_data=\"level\":$level,\"text\":\"$text\"

#
# Define the custom event name, which must be the same as the Custom Event trigger
# on = {customEvents = { 'setalert', }, },
custom_event=setalert

#
# Send HTTP API Custom Event
#
curl -H 'Content-type: application/json'
http://$domoticz_ip:$domoticz_port/json.htm?type=command&param=customevent&event=$custom_event&data=\{$
json_data\}
```

## Solution Data from File

This solution reads the data, to update the Alert sensor, from a file.  
The file content is a JSON string with key:value pairs. Example:

```
{"level": 4, "text": "Alert Level set to 4"}
```

In the example, the file is generated by the Python script. The Python scripts sends an HTTP API request with a custom event and parameter filename.

The Domoticz custom event read the full content of the file, parses the JSON string to a Lua table which properties are used to update the Alert sensor level & text.

The usage of a file with JSON data is easier to maintain, then as HTTP API request parameter.

In the example, the Python script is stored in folder

```
/home/pi/python
```

The files used are:

explore_fastevent_http_2.py	Python script generating in regular intervals less one minute the JSON data file
explore_fastevent_http_2.json	Text file with JSON key:value pairs generated by the Python script
explore_fastevent_http_2.dzvents	Domoticz automation event script triggered by the Python script

## Pseudo Code

```
Python
DELAYSECONDS = 5
Create file with JSON data
Submit, every DELAYSECONDS, a HTTP API request to Domoticz with param=customevent and data filename

dzVents Lua
Trigger customEvent
Get the filename from the data property, amend the fullname
Read the file content to JSON data
Parse the JSON data to Lua table
Act, i.e. update an alert sensor level & text
```

## Python Script

```

# explore_fastevent_http_2.py
## domoticz-workbook - function explore_events_dzvents
## Submit in regular intervals less than 1 minute data to domoticz to update an alert sensor.
## A domoticz custom event (explore_fastevent_http) handles the incoming data defined as json by
## parsing the properties level & text and update the alert sensor.
## The data parameter is a filename.
## The file contains a json string. Example: {"level": 4, "text": "Alert Level set to 4"}
## 20201009 rwbl

import json
import requests
import time

# generate random integer values used to set the alert level between 1 - 4
from random import seed
from random import randint
# seed random number generator
seed(1)

## Domoticz ip is local with default port
domoticz_ip = "http://127.0.0.1"
domoticz_port = 8080
custom_event = "setalert2"
file_name = "explore_fastevent_http_2.json"
## Alert device level and text - see function send_request
alert_level = randint(1, 4)
alert_text = "Alert Level set to {}".format(alert_level)
## Loop counter with the delay in seconds between the loop
counter = 0
maxcount = 3
delay = 10

## Define the json data with keys level (integer) and text (string)
def set_data(level, text):
    data = {}
    data['level'] = level
    data['text'] = text
    return json.dumps(data)

def create_file(level, text):
    f = open(file_name,"w+")
    f.write(set_data(level, text))
    f.close()
    print(set_data(level, text))

## Send the HTTP API request to domoticz to update the alert device
## In the custom event the idx of the alert device is defined
def send_request():
    # increase the counter
    global counter
    counter = counter + 1
    # define the json data for the custom event request
    #
    alert_level = randint(1, 4)
    alert_text = "Alert Level set to {}".format(alert_level)
    create_file(alert_level, alert_text)
    json_data = file_name
    # define the url to signal the domoticz custom event
    url = "{}:{}//json.htm?type=command&param=customevent&event={}&data={}".format(domoticz_ip,
domoticz_port, custom_event, json_data)
    print("Counter:{}, URL:{}".format(counter, url))
    # submit the url
    try:
        r = requests.get(url, timeout=3)
        r.raise_for_status()                                     # 200
        content_json = json.loads(r.content)
        # print(content_json)                                     # {'status': 'OK', 'title': 'Custom
Event'}
        print(content_json['status'], content_json['title'])   # OK Custom Event
    except requests.exceptions.HTTPError as errh:
        print ("Http Error:",errh)

```

```

except requests.exceptions.ConnectionError as errc:
    print ("Error Connecting:",errc)
except requests.exceptions.Timeout as errt:
    print ("Timeout Error:",errt)
except requests.exceptions.RequestException as err:
    print ("OOps: Something Else",err)

## main
if __name__ == '__main__':
    while True:
        send_request()
        if counter == maxcount:
            print("Done\n")
            quit()
        time.sleep(delay)

```

## Custom Event

```

--[[[

fastevent_http_2.dzvents
Trigger: customEvent
To update in regular intervals less than 1 minute, the level & text of a virtual alert sensor.
The custom event receives data from a Python script running on the Raspberry Pi CLI.
The custom data contains the filename without full path.
The data is stored in a text file with JSON key:value pairs. Example:
 {"level": 4, "text": "Alert Level set to 4"}
The filename full path is set, and the file content is read as data.
The data is converted to a Lua table and used to update the alert device level & text.
Dependencies: explore_fastevent_http_2.py
20201009 rwbl
]]--
-- Idx of the alert sensor
local IDX_ALERT = 26
-- Data File path - must end with /
local DATA_FILE_PATH = "/home/pi/python/"

-- Read the content of a text file
local function readFile(f)
    local file = io.open (f, 'r')    -- open read mode
    local data =  file:read('*a')   -- read all
    file:close()                  -- close
    return data                    -- return all filecontent
end

return
{
    on = { customEvents = { 'setalert2', }, },
    logging = { level = domoticz.LOG_DEBUG, marker = 'SETALERT2', },

    execute = function(domoticz, item)
        domoticz.log(("Item Data: %s"):format(item.data))
        -- Get the filename from the property item.data and amend the full filepath
        local filename = ("%s%s"):format(DATA_FILE_PATH, item.data)
        -- Read the file content and convert item data to lua table
        local data = domoticz.utils.fromJSON(readFile(filename))
        -- domoticz.utils.dumpTable(data)
        domoticz.log(("Level:%d,Text:%s"):format(data.level, data.text))
        -- Update the alert device
        domoticz.devices(IDX_ALERT).updateAlertSensor(data.level, data.text)
    end
}

```

## Asynchronous HTTP requests

With dzVents it is possible to make asynchronous HTTP requests and handle the results. The advantage of making asynchronous HTTP requests, is that the Domoticz system is not blocked while waiting for the response.

An example of using an asynchronous HTTP request is function Indoor Air Quality (Tinkerforge, Script). See the [Pseudo Code](#) and the Python + dzVents Lua scripts.

The concept is to submit, via dzVents Lua script, an OS command with the “&” character added. This will start the command as a process in the background.

The Python process triggers, when done, a custom event with data, which is handled by the same dzVents Lua script.

1. dzVents Custom Event - Trigger Timer  
Execute Python script as background process (to not block Domoticz)
2. Python Script  
Send Domoticz HTTP API request to the Custom Event with JSON string as parameter holding the bricklet data
3. dzVents Custom Event – Trigger Custom Events  
Get Item Data JSON and Parse JSON to Lua Table  
*Note*  
This step is not required in Domoticz 2020.2 but in BETA 2020.2 (build 12467 or higher) it is required  
Handle Data

# Grafana & InfluxDB

## Purpose

To explore how to create simple charts using Grafana with the InfluxDB data source supplied by Domoticz data, running on a Raspberry Pi 4 with Raspberry Pi OS (32bit).

As a beginner applying Grafana & InfluxDB, this chapter focus on how to create a Dashboard gauges & line charts.

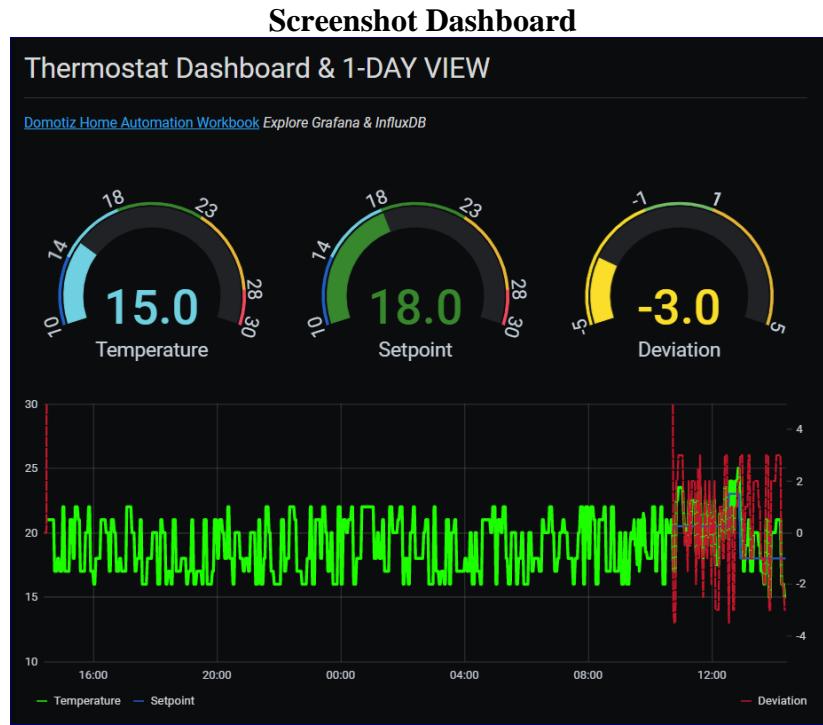
The approach is to begin with one panel dashboards for a single device line chart followed by multi-devices line chart & gauge chart to finally the combine gauge & line charts in a two panel dashboard.

**Versions:** Grafana 7.3.4, InfluxDB 1.8.3, Domoticz 2020.2 (build 12709).

### Notes

There could be other (better) ways to setup the solution or create charts – *information shared might change over time when getting to know Grafana & InfluxDB better.*

Grafana offers many features – these to describe goes beyond this chapter – please refer the docs. The shared examples are briefly described.



## Test Scenario

Used the same as described in chapter *Explore Custom Pages Charts* [Test Scenario](#):

- Two Domoticz devices: Thermostat and Temperature.
- Temperature changes every 5 minutes randomly +/-3% from the Thermostat Setpoint.
- Deviation is calculated as temperature – setpoint.

## InfluxDB

The InfluxDB stores data from Domoticz - either via

- Domoticz option Data Push  
(GUI > Setup > More Options > Data push > InfluxDB)  
or
- dzVents scripts executing
  - curl with InfluxQL (Influx Query Language) statements or
  - function openURL with postData

The Grafana charts use the InfluxDB data source by querying with the Domoticz Device IDX (or use the Device Name is also possible) as WHERE clause in the SELECT statement.

## Installation

Summary of the CLI commands used to install the InfluxDB on the Raspberry Pi.  
Using username "pi".

```
$ wget -qO- https://repos.influxdata.com/InfluxDB.key | sudo apt-key add -
$ echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee
/etc/apt/sources.list.d/InfluxDB.list
$ sudo apt update
$ sudo systemctl unmask InfluxDB
$ sudo systemctl enable InfluxDB
$ sudo systemctl start InfluxDB
```

## Create Database

The test database named *domoticz* is created using the CLI *influx*.

```
$ influx
Connected to http://localhost:8086 version 1.8.3
InfluxDB shell version: 1.8.3
> create database domoticz
> use domoticz
Using database domoticz
```

*Note*

For the ease of use, no username and password set.

## Time Series

InfluxDB stores data as time series (“Time series database”).  
Each series has a name with one or more columns holding values.

In SQLite terms it is like:

- Measurement = Table
- Time = Column as primary index
- Tags and Fields = Column names

### Example CLI Influx

For the Thermostat scenario, *insert* & *select* the temperature for a location using the CLI *influx* to execute *InfluxQL* statements:

- Measurement name = temperature
- Tag key = location - set as MakeLab
- Field key = value – set as 21.0

The time is set by InfluxDB in UNIX format.

#### *InfluxQL Statements*

```
INSERT temperature, location=MakeLab value=21.0
```

```
SELECT * FROM temperature
name: temperature
time           location      value
----           -----        -----
1574055049844513350 MakeLab      21.0
```

### Example HTTP API Request Response (snippet)

Using HTTP API Select Requests gives an insight how data is stored (see InfluxDB API).

```
"series": [
  {
    "name": "Temperature",
    "columns": [ "time", "idx", "name", "value"],
    "values": [["2020-12-01T17:49:42.860238Z", "-30", "MakeLab-Deviation", 3]]
  }
]
```

## InfluxDB API

The *InfluxDB API* enables to write or select data by submitting HTTP POST requests via *curl* (command line tool for sending HTTP requests) with *InfluxQL* statements.

The HTTP API for InfluxDB v1.8.3 are used.

*Curl* can be used in Domoticz via dzVents or Python scripts (see example TODO).

Next an example for inserting and selecting data via curl submitted from the CLI.

### HTTP Request Insert

Insert data for measurement Temperature with tags idx, name and a value.

```
curl -i -XPOST 'http://localhost:8086/write?db=domoticz' --data-binary 'Temperature, idx=-30, name=MakeLab-Deviation value=3'
```

#### Notes

The idx is negative to indicate this is a device with a calculated value and not a Domoticz regular device stored in the Domoticz database.

This is just an idea on how indicate the device is not a regular Domoticz device.

If the name tag is enclosed in quotes, these values are stored in the InfluxDB as well.

Example:

name="MakeLab-Deviation" is stored as name="\\"MakeLab-Deviation\\\"".

So be careful setting quotes.

### HTTP Request Select

Check if the previous inserted data has been added to the database *domoticz*.

```
curl -G "http://localhost:8086/query?db=domoticz" --data-urlencode "q=select * from Temperature where idx='-30'"
```

#### Note

The IDX value is stored as a string, there for put in brackets in the WHERE clause.

### HTTP Response Select

The results array contains a single row only.

```
{
  "results": [
    {
      "statement_id": 0,
      "series": [
        {
          "name": "Temperature",
          "columns": [ "time", "idx", "name", "value" ],
          "values": [ ["2020-12-01T17:49:42.860238Z", "-30", "MakeLab-Deviation", 3] ]
        }
      ]
    }
  ]
}
```

## Other InfluxQL examples

### Delete

Delete all rows where time > 2020-12-02

```
curl -G "http://localhost:8086/query?db=domoticz" --data-urlencode "q=delete from Temperature where time > '2020-12-02'"
```

OR with timestamp 00:00:00Z

```
curl -G "http://localhost:8086/query?db=domoticz" --data-urlencode "q=delete from Temperature where time > '2020-12-02T00:00:00Z'"
```

# Grafana Installation

## Summary

Summary of the CLI commands used to install Grafana on the Raspberry Pi.  
Using username "pi".

```
$ sudo apt-get update && sudo apt-get upgrade
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
$ sudo apt-get install -y grafana
Tried but got grafana is not available message.
**WORKAROUND**
$ wget https://dl.grafana.com/oss/release/grafana_6.3.3_armhf.deb
$ sudo dpkg -i grafana_6.3.3_armhf.deb
$ sudo apt-get update
$ sudo apt --fix-broken install
$ sudo /bin/systemctl daemon-reload
$ sudo /bin/systemctl enable grafana-server
$ sudo /bin/systemctl start grafana-server
$ sudo update-rc.d grafana-server defaults
$ sudo apt-get update && sudo apt-get upgrade
This command upgraded Grafana to 7.3.4
```

Started exploring Grafana by following [this](#) tutorial.

From a browser, opened the Grafana UI and logged in with username admin & password:

```
http://domoticz-ip:3000/login
```

Changing the password first.

## Data Source InfluxDB

The data source *InfluxDB* is added as required to create the charts with Domoticz data.  
Configuration:

- HTTP > URL: http://localhost:8086
- InfluxDB Details > Database: domoticz
- HTTP method: GET

All other settings left as default.

## JSON Model

Each dashboard has a JSON model which can be edited / viewed / saved.

Select the dashboard settings > JSON Model.

For each of the examples, the JSON model is stored

### Screenshots JSON Model Example Makelab Temperature Trend

The screenshot shows two side-by-side views of a Grafana dashboard. On the left is the 'General' settings page for a dashboard named 'Temperature Trend'. It includes fields for Name, Description, Tags, Folder, and Editable status. Below that are Time Options for Timezone, Auto-refresh (set to 5s, 10s, 30s, 1m, 5m, 15m, 30m, 1h, 2h, 1d), Now delay now (0m), and Hide time picker. At the bottom are Panel Options for Graph Tooltip (Default) and a Delete Dashboard button. On the right is a 'JSON Model' view showing the JSON structure for the dashboard. The JSON code defines annotations, editability, panel IDs, and a list of panels, each with specific properties like alias colors, bars, dash length, and data source.

```
{
  "annotations": {
    "list": [
      {
        "builtin": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ],
    "editable": true,
    "gnetId": null,
    "graphTooltip": 0,
    "id": 6,
    "links": [],
    "panels": [
      {
        "aliasColors": {},
        "bars": false,
        "dashLength": 10,
        "dashes": false,
        "datasource": "InfluxDB",
        "description": ""
      }
    ]
  }
}
```

**Source:**  
makelab-temperature-trend.json

As mentioned, the JSON model can be edited and saved .. but needs to have a good knowledge of the key:value properties.

Next step is to configure Domoticz to be able to store data in the InfluxDB.

# Domoticz Configuration

## Data Push InfluxDB

Domoticz supports writing data to a InfluxDB database via the *Data push* option. This is called *InfluxLink*.

Go to GUI > Setup > More Options > Data push > InfluxDB.

The screenshot shows the 'Edit link' configuration page in the Domoticz GUI. At the top, there is a table listing three entries:

Idx	Device name	Value to send	Target type	Enabled
32	MakeLab Thermostat Battery	Percentage	On Value Change	Yes
31	MakeLab Temperature	Temperature	On Value Change	Yes
30	MakeLab Thermostat Setpoint	Temperature	On Value Change	Yes

Below the table, the 'Edit link' configuration form is displayed for the 'MakeLab Thermostat Setpoint' entry. The 'General settings' section contains the following fields:

- Device name: MakeLab Thermostat Setpoint
- Value to send: Temperature
- Target type: On Value Change
- Link active: ✓
- Enabled: ✓
- Remote Address: 127.0.0.1
- Port: 8086
- Path: (Only when using a remote proxy)
- Database: domoticz
- Username:
- Password:
- Debug to logfile: ✓

At the bottom of the form are 'Save' and 'Cancel' buttons.

### General settings

- Enabled: Selected
- Remote Address: 127.0.0.1  
*Note:* InfluxDB running on same system as Domoticz
- Port: 8086
- Path: Left empty
- Database: domoticz  
*Note:* as created previously in InfluxDB
- Username: Left empty  
*Note:* For the ease of use, no username and password set
- Password: Left empty
- Debug to logfile: Selected  
*Note:* The entries are tagged with InfluxLink in the Domoticz log

### Edit Link

From the test scenario, three devices with an *InfluxLink* are added.

The data is written to the InfluxDB when the value is changed, i.e. trigger “On Value Change”.

#### *Note*

For the Thermostat Setpoint, the database needs to be populated first time.

This can be done by either changing the value with target type “On Value Change” or wait till Domoticz sends the value to the InfluxDB with target type “Direct”.

The target type “Direct” sends data to the InfluxDB by for example an automation event dzVents triggered by a timer.

Example:

```

EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
... Status: dzVents: Info: Handling httpResponse-events for: "U6000E6201"
... Status: dzVents: Info: ----- Start internal script: heating_system_control: HTTPResponse:
"U6000E6201"
... Status: dzVents: Info: {[{"setpoint":21.0, "iseid":1541, "level":47.0, "temperature":22.0}]
... Status: dzVents: Info: Updating Device: 1541, SP: 21.0, PV: 22.0, L: 47.0
... Status: dzVents: Info: ----- Finished heating_system_control
... Status: EventSystem: Script event triggered: /home/pi/domoticz/dzVents/runtime/dzVents.lua
... Sending SetPoint to device....
... (VirtualSensors) Thermostat (TRC-6201-SP)
... InfluxLink: value Temperature,idx=52,name=TRC-6201-SP value=21
... InfluxLink: value Temperature,idx=51,name=TRC-6201-PV value=22
... InfluxLink: value Percentage,idx=53,name=TCV-6201-PV value=47
... Status: dzVents: Info: ----- Start internal script: heating_system_control:, trigger: "every 5
minutes"
... Status: dzVents: Info: ----- Finished heating_system_control

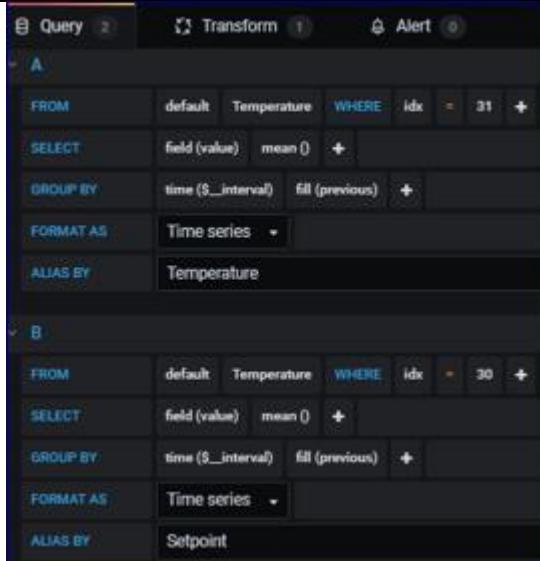
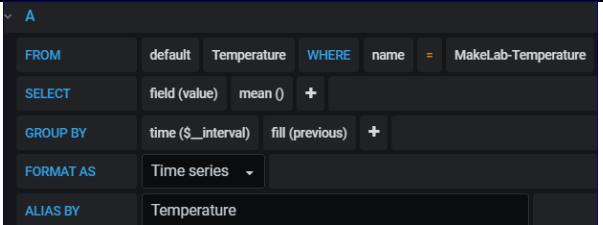
```

The Domoticz Log gives an insight on how Domoticz writes the InfluxDB Measurement name, Tag & Field keys.

The marker is InfluxLink with measurement Temperature, the tags idx & name, and a value.

```
2020-11-30 15:15:00.642 InfluxLink: value Temperature,idx=31,name=MakeLab-Temperature value=19.5
```

The tag idx or name is used in Grafana for the Query to select the data for the chart(s).

WHERE idx	WHERE name
	

### Example InfluxDB Selection using CLI influx with InfluxQL

InfluxDB selecting data from the Measurement names Temperature & Percentage.

```

> select * from Temperature
name: Temperature
time          idx name          value
-----        --- ---
1606744740000000000 31 MakeLab-Temperature 22.5
> select * from Percentage
name: Percentage
time          idx name          value
-----        --- ---
1606744860000000000 32 MakeLab-Thermostat-Battery 49
1606744920000000000 32 MakeLab-Thermostat-Battery 27

```

## Automation Script dzVents

Another option to write to the InfluxDB is by running automation scripts dzVents Lua either executing *curl* commands or function openURL with post data (of course would also be possible via Python scripts – not covered here)

... BUT ... it is better to stick to the Domoticz Data push InfluxDB feature (by either sending direct or on value change) and let Domoticz handle the communication with the InfluxDB.

... NEXT IS JUST TO EXPLORE other mechanisms ... AND NOT RECOMMENDED.

Possible usage

- Regular updates of Domoticz devices via timer trigger instead of InfluxLink options “direct” or “On Value Change”.
- Write calculated data to the InfluxDB.  
The data is not assigned to a Domoticz device.  
The insert pattern could be the same as used by the InfluxLink but with, for example a negative IDX to indicate this is not a regular Domoticz device.  
Grafana does not need to perform the calculations, i.e. Transform as the data is available via Query WHERE clause on the idx.
- Write data in regular intervals to get trending data with aligned values, i.e. time series (even if values have not changed).

### Example

Used for the Line Chart Multiple Devices.

If the thermostat temperature changes, write the deviation to the InfluxDB.

```
deviation = temperature - setpoint
-- NOT USED but for future enhancements
deviationpct = (deviation / temperature) * 100
```

The InfluxDB HTTP API v1 is used to write the data.

A successful write returns

```
HTTP/1.1 204 No Content.
```

Write the **deviation** as temperature to the InfluxDB measurement Temperature.

Write by following the same pattern as Domoticz, but with a negative IDX as the device does not exist in the Domoticz database.

Example Domoticz pattern InfluxLink:

```
Temperature,idx=31,name=MakeLab-Temperature value=19.5
```

Example Deviation with idx=-30 and name set

```
Temperature,idx=-30,name=MakeLab-Deviation value=3
```

### Insert New Row using curl

HTTP Request InfluxDB 1.8:

```
curl -i -XPOST 'http://localhost:8086/write?db=domoticz' --data-binary 'Temperature,idx=-30,name=MakeLab-Deviation value=3'
```

HTTP Response OK:

```
HTTP/1.1 204 No Content
Content-Type: application/json
Request-Id: 3c9880ce-3487-11eb-85ea-dca63200bcfe
X-InfluxDB-Build: OSS
X-InfluxDB-Version: 1.8.3
X-Request-Id: 3c9880ce-3487-11eb-85ea-dca63200bcfe
Date: Wed, 02 Dec 2020 10:15:00 GMT
```

## Option curl

Insert the Domoticz data in the InfluxDB by executing curl command.

Source: makelab\_thermostat\_InfluxDB\_curl.dzvents

```
-- Device idx
local IDX_SETPOINT = 30
local IDX_TEMPERATURE = 31
local IDX_DEVIATION = -30
local IDX_DEVIATIONPCT = -31

-- Post data to InfluxDB using curl
-- A successful write returns HTTP/1.1 204 No Content
-- Example data: Temperature,idx=-30,name=MakeLab-Deviation value=-3.0
local function postCurl(domoticz, data)
    local CODEWRITEOK = "204";
    local curlCmd = string.format("%s '%s'",
        "curl -i -XPOST 'http://localhost:8086/write?db=domoticz' --data-binary ", data)
    local returnValue, returnCode = domoticz.utils.osCommand(curlCmd)
    if string.match(returnValue, CODEWRITEOK) then
        domoticz.log(string.format('[OK] InfluxDB data insert: %s', data))
    else
        domoticz.log(string.format('[ERROR] InfluxDB data insert: %s, %s', data,
        returnValue),domoticz.LOG_ERROR)
    end
end

return {
    on = {
        devices = { IDX_TEMPERATURE }
    },
    logging = {
        level = domoticz.LOG_INFO, marker = 'THERMOSTATINFLUXDB'
    },
    execute = function(domoticz, triggeredItem)
        if triggeredItem.trigger == isDevice then
            if triggeredItem.idx == IDX_TEMPERATURE then
                -- domoticz.log(string.format('Device Trigger: %d', IDX_TEMPERATURE))
                -- Calculate the Deviation
                local temperatureDeviation = domoticz.devices(IDX_TEMPERATURE).temperature -
domoticz.devices(IDX_SETPOINT).setPoint
                local temperatureDeviationPct = (temperatureDeviation /
domoticz.devices(IDX_TEMPERATURE).temperature) + 100
                -- Define the URL data
                local insertData = string.format('Temperature,idx=%d,name=MakeLab-Deviation value=%.1f',
                    IDX_DEVIATION, temperatureDeviation)
                postCurl(domoticz, insertData)
            end
        end
    end
}
```

## Option openURL

Insert the Domoticz data in the InfluxDB using function openURL().

### ISSUE

Using openURL() with callback gives an error (in the Domoticz log) after completing isHTTPResponse:

```
Error parsing json to LUA table: (invalid json string)
```

This is because the post data is not a proper json string, i.e.

```
Temperature,idx=-30,name=MakeLab-Deviation value=3
```

BUT the data is successfully written to the InfluxDB.

Which means the error message is correct and expected because a JSON string with a single quote in it is invalid.

### Question

<TODO> How to define the body as a proper Lua table because measurement has no key:value. The body is a string.

### Domoticz Log Example

See error at the end = (invalid json string).

```
THERMOSTATINFLUXDB: ----- Start internal script: thermostat_InfluxDB_http: Device: "MakeLab
Temperature (VirtualSensors)", Index: 31
THERMOSTATINFLUXDB: Device Trigger: 31
THERMOSTATINFLUXDB: openURL: Temperature,idx=-30,name=MakeLab-Deviation value=1.0
THERMOSTATINFLUXDB: ----- Finished thermostat_InfluxDB_http

Handling httpResponse-events for: "THERMOSTATINFLUXDB"
THERMOSTATINFLUXDB: ----- Start internal script: thermostat_InfluxDB_http: HTTPResponse:
"THERMOSTATINFLUXDB"
THERMOSTATINFLUXDB: [OK] Insert InfluxDB row.
THERMOSTATINFLUXDB: ----- Finished thermostat_InfluxDB_http
(3.0.17) THERMOSTATINFLUXDB: Error parsing json to LUA table: (invalid json string)
```

### WORKAROUND

If the callback is not used, no error message BUT then no check if the InfluxDB has been updated (i.e. code 204 check not possible).

### Source: thermostat\_InfluxDB\_http.dzvents

```
local HTTPAPIRESPONSE = 'THERMOSTATINFLUXDB'
local HTTPAPIRESPONSECODEOK = 204

-- Device idx
local IDX_SETPOINT = 30
local IDX_TEMPERATURE = 31
local IDX_DEVIATION = -30
local IDX_DEVIATIONPCT = -31

-- Post data to InfluxDB using HTTP
-- Example:
-- url: http://localhost:8086/write?db=domoticz&precision=ms
-- type :POST
-- body: Temperature,idx=-30,name=MakeLab-Deviation value=3
local function postHTTP(domoticz, data)
    -- Submit http request
    domoticz.openURL({
        url = 'http://127.0.0.1:8086/write?db=domoticz&precision=ms',
        method = 'POST',
```

```
        callback = HTTPAPIRESPONSE,
        postData = data
    })
    domoticz.log('openURL: ' .. data)
    return
end

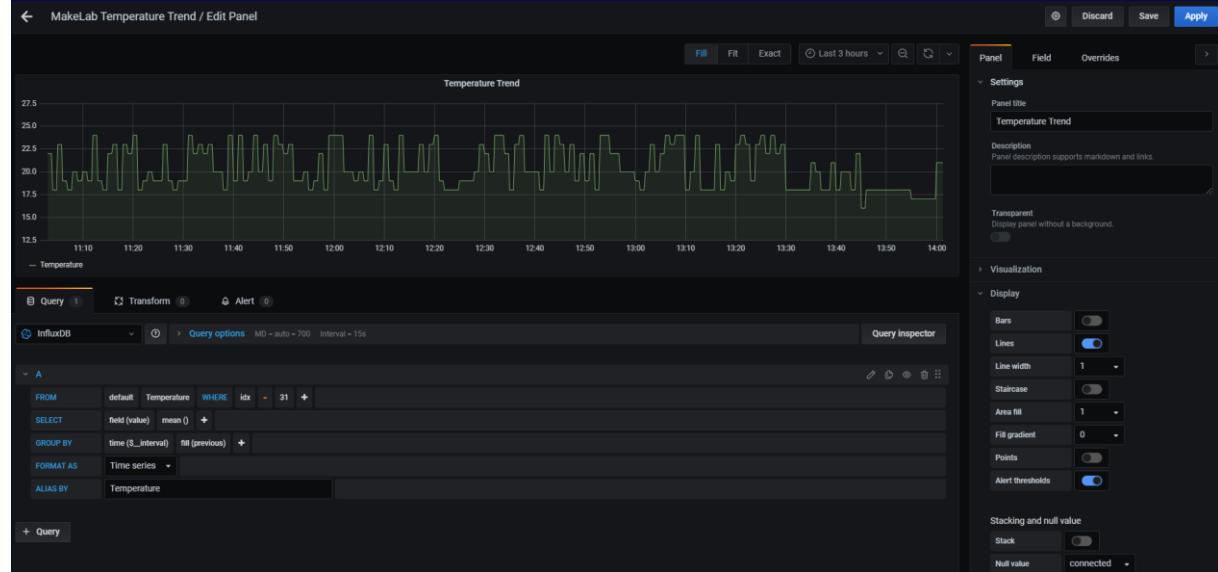
return {
    on = {
        devices = { IDX_TEMPERATURE },
        httpResponses = { HTTPAPIRESPONSE }
    },
    logging = {
        level = domoticz.LOG_INFO,
        marker = 'THERMOSTATINFLUXDB'
    },
    execute = function(domoticz, triggeredItem)
        -- Trigger device temperature
        if triggeredItem.trigger == isDevice then
            if triggeredItem.idx == IDX_TEMPERATURE then
                domoticz.log(string.format('Device Trigger: %d', IDX_TEMPERATURE))
                -- Calculate the Deviation
                local temperatureDeviation = domoticz.devices(IDX_TEMPERATURE).temperature -
domoticz.devices(IDX_SETPOINT).setPoint
                local temperatureDeviationPct = (temperatureDeviation /
domoticz.devices(IDX_TEMPERATURE).temperature) + 100
                -- domoticz.log(string.format('Deviation: %.1f, %.1f', temperatureDeviation,
temperatureDeviationPct))
                -- Define the URL data
                local insertData = string.format('Temperature,idx=%d,name=MakeLab-Deviation
value=%.1f',
                                                IDX_DEVIATION, temperatureDeviation)
                postHTTP(domoticz, insertData)
            end
        end

        -- HTTP response trigger. Check the status code on 204 (OK).
        if triggeredItem.isHTTPResponse then
            if triggeredItem.statusCode == HTTPAPIRESPONSECODEOK then
                domoticz.log(string.format('[OK] Insert InfluxDB row.'))
            else
                domoticz.log(string.format('[ERROR] Insert InfluxDB row. Code: %d',
triggeredItem.statusCode))
            end
        end
    end
}
```

## Line Chart Single Device

Summary creating single device line chart *MakeLab Temperature Trend* logging the temperature of the Domoticz device with IDX=31 (see previous Domoticz InfluxLink).

### Screenshot Temperature Trend / Edit Panel



### Create Dashboard with single Panel

From the Grafana UI, select + > Create Dashboard > Add new panel > Visualisation: Graph.

### Configure Data Query (#1)

Select Query > Database > InfluxDB

After selecting the InfluxDB, a new query is listed name A. Set the query A properties:

Query A		
FROM default Temperature WHERE idx = 31 SELECT field(value) mean() GROUP BY time (\$__interval) fill (previous) FORMAT AS Time series ALIAS BY Temperature		
<i>Option</i> Use WHERE name=MakeLab-Temperature		
Settings		
Panel title MakeLab Temperature Trend	Field No changes = left defaults	Time Range Last 24-Hours
Stacking and null value Null value: connected		Refresh 5m

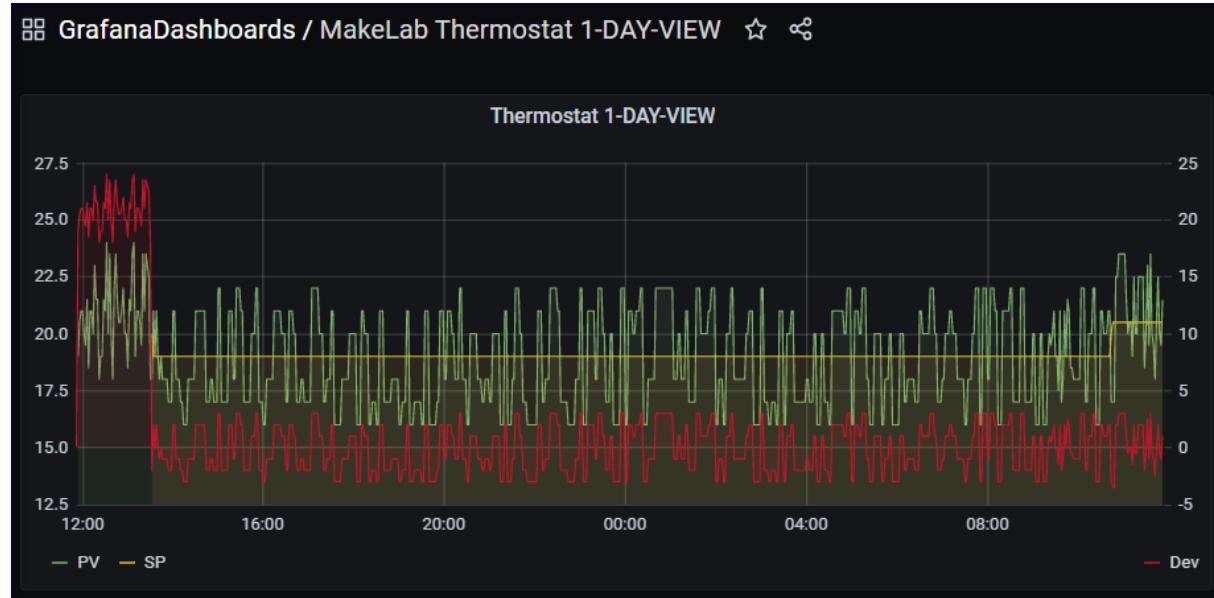
**General:** Name: MakeLab Temperature Trend

**JSON Model:** Source: makelab-temperature-trend.json

## Line Chart Multiple Devices

Summary creating multi device line chart *Thermostat 1-DAY-VIEW* logging the setpoint (idx=30), temperature (idx=31) and deviation (idx=-30).

### Screenshot



### Create Dashboard with single Panel

From the Grafana UI, select + > Create Dashboard > Add new panel > Visualisation: Graph.

### Configure Data Queries (#2) and Transform (#1)

Select Query > Database > InfluxDB > Create 2 Queries A, B, and 1 Transform

Query A	Query B	Transform
<pre>FROM default Temperature WHERE name = MakeLab- Temperature SELECT field(value) mean() GROUP BY time (\$__interval) fill (previous) FORMAT AS Time series ALIAS BY PV</pre>	<pre>FROM default Temperature WHERE name = MakeLab- Thermostat-Setpoint SELECT field(value) mean() GROUP BY time (\$__interval) fill (previous) FORMAT AS Time series ALIAS BY SP</pre>	Add field from Calculation Mode: Binary Operation Operation: PV - SP Alias: Dev
<b>Settings</b>		
<b>Panel title</b> Thermostat 1-DAY-VIEW	<b>Field</b> No changes = left defaults	<b>Time Range</b> Last 24-Hours
<b>Stacking and null value</b> Null value: connected		<b>Refresh</b> 5m

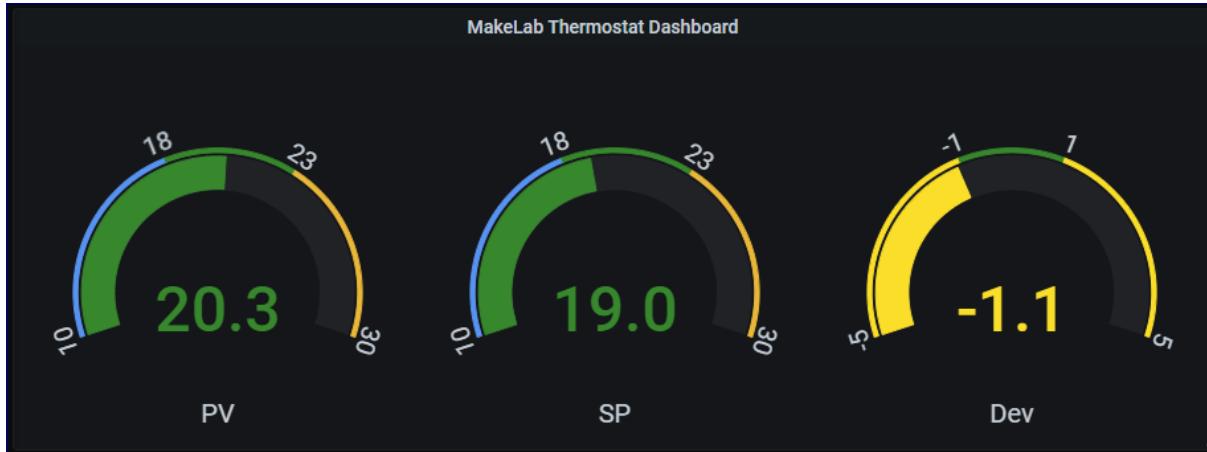
**General:** Name: MakeLab Thermostat 1-DAY-VIEW

**JSON Model:** Source: makelab-thermostat-trend.json

## Gauge Chart Multiple Devices

Summary creating a multi device gauges chart *Thermostat Dashboard* showing the last values of the setpoint (idx=30), temperature (idx=31) and deviation (calculated).

### Screenshot



### Create Dashboard with single Panel

From the Grafana UI, select + > Create Dashboard > Add new panel > Visualisation: Gauge.

### Configure Data Queries (#2) and Transform (#1)

Query A	Query B	Transform
<pre>FROM default Temperature WHERE name = MakeLab- Temperature SELECT field(value) mean() GROUP BY time (\$__interval) fill (previous) FORMAT AS Time series ALIAS BY PV</pre>	<pre>FROM default Temperature WHERE name = MakeLab- Thermostat-Setpoint SELECT field(value) mean () GROUP BY time (\$__interval) fill (previous) FORMAT AS Time series ALIAS BY SP</pre>	Add field from Calculation Mode: Binary Operation Operation: PV - SP Alias: Dev
		<b>Settings Override 1</b> Field with name: Dev Min: -5, Max: 5 Thresholds <ul style="list-style-type: none"> <li>• Base: Yellow</li> <li>• 1: Green</li> <li>• -1: Yellow</li> </ul>
<b>Settings</b>		
<b>Panel</b> <b>Panel title</b> Thermostat SP PV Dev	<b>Field</b> Unit: none Min: 10, Max: 30 Thresholds <ul style="list-style-type: none"> <li>• Base: Blue</li> <li>• 18: Green</li> <li>• 23: Yellow</li> </ul>	<b>Time Range</b> Today  <b>Refresh</b> 5m

**General:** Name: MakeLab Thermostat Dashboard

**JSON Model:** Source: makelab-thermostat-dashboard.json

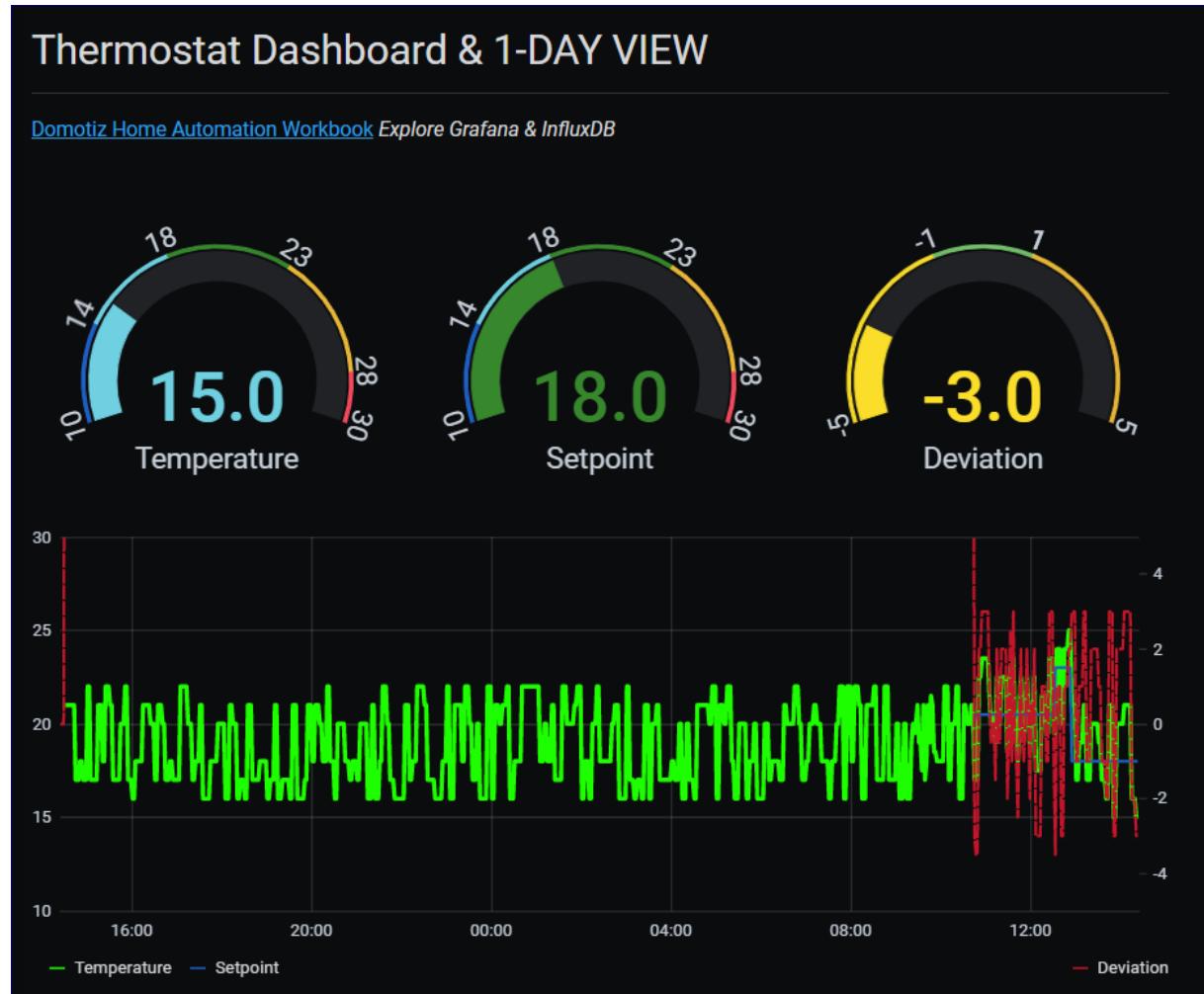
## Gauge & Line Chart

This dashboard is the target of this chapter.  
It is based upon the previous created dashboards.

There are three panels:

- **Top Visualization: Text**  
The text is defined with text mode Markup.
- **Middle Visualization: Gauges**  
Latest values for the Thermostat Temperature, Setpoint and the deviation.  
Changes previous gauges chart: Thresholds added, Panel title removed
- **Bottom Visualization: Line**  
24-hours data updated every 5 minutes.  
Changes previous line chart: Panel title removed

### Screenshot Dashboard



**General:** Name: MakeLab Dashboard

**JSON Model:** Source: makelab-dashboard.json

# GPIO

## Purpose

- To explore how to use Raspberry Pi General Purpose Input Output (GPIO) ports
- To define Python scripts with functions to control GPIO via the CLI
- To trigger Domoticz device actions with the previous defined scripts via script://
- To create Python Automation Events handling GPIO
- To develop Python Plugins handling GPIO
- To test with the Python libraries
  - [RPi.GPIO](#) low level - scripts, device actions, automation events, plugins
  - [GPIO Zero](#) - plugins only
  - Other like LCDI2C

The **initial focus** is on how to use

- GPIO output with an LED and
- GPIO input with a push-button

with interaction and Domoticz device integration.

The next step is to explore other components, like displays, sensors etc..

### Possible Usage

- (output) LED as indicator for hardware state, notifications ...
- (input) Push-button to switch lights on, set setpoints, read sensor ...

There are loads of Python libraries available for all kind of sensors & actuators.

### Plugin Thoughts

Some thoughts on using a Python Plugin with the RPi.GPIO or GPIO Zero library:

Developing a plugin, enables to create a proper solution for handling GPIO.

The GPIO is setup once when starting the plugin and the pins are cleanup when stopping the plugin.

The plugin can act as a base with Automation Events dzVents Lua to enhance functionality.

For example, the plugin handles immediate button callbacks which sets the state of a Domoticz device. The state change is then handled by a dzVents script device change.

There is no need to change the plugin in case requirements change on the functionality - (just) enhance the dzVents script.

The use of dzVents is rather powerful and easier to use.

For further plugin development, the newer GPIO Zero library will be used.

### Why not using Domoticz Hardware Raspberry Pi GPIO?

Although Domoticz has built in hardware for GPIO, decided to use the Raspberry Pi libraries RPi.GPIO and GPIOZero.

The (simple) experiments are tested first interactive via the command line interface to then test either as Domoticz standalone scripts or as Domoticz automation events. To close out developed some plugins.

## Use Arduino

As stated by the [project](#):

*This RPi.GPIO module is unsuitable for real-time or timing critical applications. This is because you cannot predict when Python will be busy garbage collecting. It also runs under the Linux kernel which is not suitable for real time applications - it is multitasking O/S, and another process may be given priority over the CPU, causing jitter in the program.*

Its recommended for true real-time performance and predictability, to use an [Arduino](#). Basically, use the RPi to control the Arduino, whereas the Arduino does the mechanics as having the sensors and actuators connected.

Hint: Instead, using an Arduino, ESP controller is an alternative (see function ESPEasy).

## Automation Python Events

The use of Domoticz Automation Python events to handle or trigger GPIO actions is probably not the best use of resources.

An automation event is running every minute. A GPIO output action is triggered by device. For an input action, the event is waiting for callbacks.

But anyhow, it is just to explore options - let us see if of use for future workbook functions.

## Notes

### Specify Interpreter

The system running Domoticz (Raspberry Pi with Raspberry Pi OS), has several Python interpreters installed.

To ensure which interpreter to use in the scripts, the shebang is used at the beginning of the script:

```
#!/usr/bin/python3
```

### Specify Path Packages

If external packages are used and are not found by the Python interpreter, the path can be specified. Example:

```
# Amend the import path to enable using the Tinkerforge libraries
# Alternate (ensure to update in case newer Python API bindings):
# create folder tinkerforge and copy the binding content, i.e.
# /home/pi/domoticz/plugins/tfrgbledv2
from os import path
import sys
sys.path
sys.path.append('/usr/local/lib/python3.7/dist-packages')
```

### Automation Event Helper

In folder

```
/home/pi/domoticz/scripts/python
```

there is a helper Python script

```
domoticz.py
```

**Issue (state: 20200612)**

The logging functions in the Python helper code are giving arguments error and therefore modified.

<TODO> Check if the helper is modified in newer Domoticz versions.

**Modify Devices**

In the helper, how to modify Domoticz devices other the switches.

<TODO> Have not found a solution and need to explore further.

**Modified Helper**

The modified helper script is called **domoticz2.py**.

This helper is imported in the Automation Events scripts:

```
import domoticz2
```

and used like

```
domoticz2.log(msg)
```

## GPIO Pin Usage

Ensure the GPIO pins used are not used by other applications or plugins. Even if a plugin uses the same (or some) of the GPIO pins and the plugin is disabled, the active plugin cannot use the GPIO pins – error setting up GPIO when starting Domoticz, check the log.

When stopping a script or plugin, cleanup the GPIO pins.

## Plugin Unique Key

Ensure each of the plugins have a unique key across all plugins:

```
<plugin key="UNIQUE KEY" name="UNIQUENAME"
```

## Plugin Development Iteration

During development, restart Domoticz after every change and check the Domoticz log.

```
sudo service domoticz.sh restart
```

If the plugin behaves not as expected, then reinstall the plugin, i.e.

1. Delete hardware
2. Restart Domoticz
3. Add hardware

## Push-Button Bounce Time

Depending on the behaviour of the push-button, need to set the bounce\_time (float or None) – If None (the default), no software bounce compensation will be performed. Otherwise, this is the length of time (in seconds) that the component will ignore changes in state after an initial change.

# RPi.GPIO

## Prepare

### Version & Path & Update

The package RPi.GPIO is installed as part of the Raspberry Pi OS. Located in folder /usr/bin.  
Command for help:

```
gpio -h
```

Find the RPi.GPIO packages and version:

```
Option 1
find /usr | grep -i rpi gpio

/usr/lib/python2.7/dist-packages/RPi.GPIO-0.7.0.egg-info
/usr/lib/python3/dist-packages/RPi.GPIO-0.7.0.egg-info

Option 2
dpkg -l python-rpi gpio

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name          Version       Architecture Description
+====-
ii  python-rpi gpio  0.7.0-0.1~bpo10+1 armhf      Module to control Raspberry Pi
```

Another way is to use a small Python script from the CLI:

```
python3
>>> import RPi.GPIO as GPIO
>>> GPIO.VERSION
'0.7.0'
>>> quit()
```

To get the Revision number:

```
python3
>>> import RPi.GPIO as GPIO
>>> GPIO.RPI_REVISION
3
>>> quit()

0 = Compute Module, 1 = Rev 1, 2 = Rev 2, 3 = Model B+/A+
```

Update the package

```
sudo apt-get update && sudo apt-get install python-rpi gpio python3-rpi gpio

Hit:1 http://archive.raspberrypi.org/debian buster InRelease
Hit:2 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:3 https://deb.nodesource.com/node_12.x buster InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-rpi gpio is already the newest version (0.7.0-0.1~bpo10+1).
python3-rpi gpio is already the newest version (0.7.0-0.1~bpo10+1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

## GPIO Pinout

gpio readall

Pi 3B+											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1    2			5v			
2	8	SDA.1	ALT0	1	3    4			5V			
3	9	SCL.1	ALT0	1	5    6			0V			
4	7	GPIO. 7	IN	1	7    8	1	IN	TxD	15	14	
		0v			9    10	1	IN	RxD	16	15	
17	0	GPIO. 0	IN	0	11    12	0	OUT	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13    14			0V			
22	3	GPIO. 3	IN	0	15    16	0	IN	GPIO. 4	4	23	
		3.3v			17    18	0	IN	GPIO. 5	5	24	
10	12	MOSI	ALT0	0	19    20			0V			
9	13	MISO	ALT0	0	21    22	0	IN	GPIO. 6	6	25	
11	14	SCLK	ALT0	0	23    24	1	OUT	CE0	10	8	
		0v			25    26	1	OUT	CE1	11	7	
0	30	SDA.0	IN	1	27    28	1	IN	SCL.0	31	1	
5	21	GPIO.21	IN	1	29    30			0V			
6	22	GPIO.22	IN	1	31    32	0	IN	GPIO.26	26	12	
13	23	GPIO.23	IN	0	33    34			0v			
19	24	GPIO.24	IN	0	35    36	0	IN	GPIO.27	27	16	
26	25	GPIO.25	IN	0	37    38	0	IN	GPIO.28	28	20	
		0v			39    40	0	IN	GPIO.29	29	21	

## Experiments

The next basic experiments are used to test the GPIO with input and output components.

1. GPIO Cleanup
  2. LED On|Off via Command Line
  3. LED On|Off via Domoticz Switch
  4. LED On|Off via Automation Event Domoticz Switch
  5. LED On|Off via Push-Button handled by Automation Event
  6. LED On|Off via Push-Button controlled by Python Plugin
  7. Push-button trigger by Python Plugin

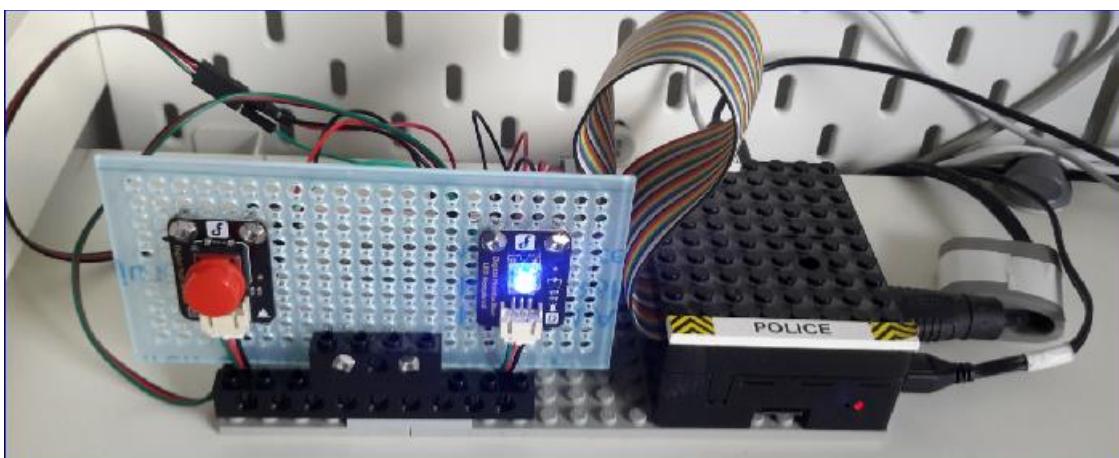
## Software

- Raspberry Pi OS - Raspbian GNU/Linux 10 (buster)
  - Python 3.7.3, GCC 8.3.0
  - RPi.GPIO library 0.7.0
  - Domoticz Beta channel 2020.2 (build 12143)

### Note

Versions are subject to change.

## Test Setup



The Raspberry Pi in a LEGO case with the DFRobot components Push-Button and LED connected. The DFRobot components are attached to a Tinkerforge Mounting Plate 12x6cm ([info](#)) with 3mm screws. *[LEGO is a trademark of the LEGO Group]*

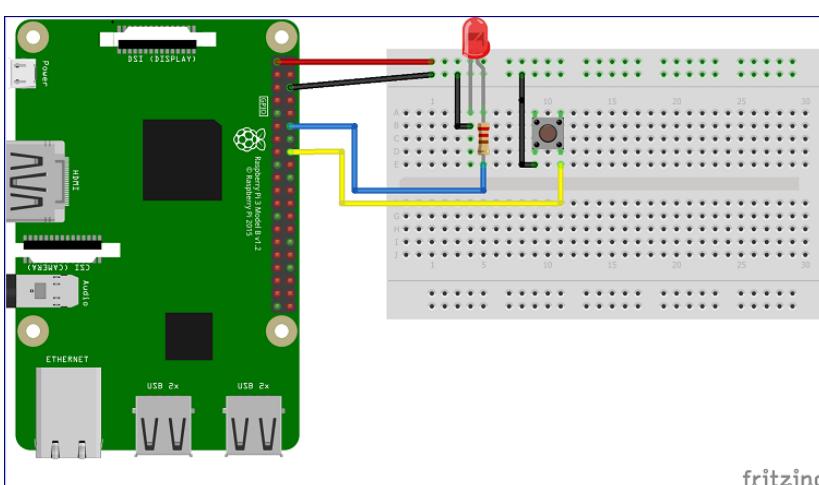
## Wiring

<b>LED</b>	<b>Raspberry Pi</b>
Signal	Pin #12 GPIO18
GND	Pin #6 GND
<b>Push-Button</b>	<b>Raspberry Pi</b>
Signal	Pin #16 GPIO23
GND	Pin #6 GND

### Note

In the scripts, the pulldown resistor is enabled, therefore no additional resistor used in the circuit.

## Circuit



## GPIO Cleanup

### Important Note

The experiments make exclusive use of the GPIO ports. The ports are cleaned up by using:

```
GPIO.cleanup()
```

If an LED is turned ON and GPIO.cleanup is used, the LED is turned OFF.

The same would apply for Button events.

If the ports should not be used, then another event could be used, to clean up the ports triggered by a switch or when Domoticz stops.

Next is an Automation Event, but it is also possible to create a script in the Python/script folder which is triggered by a Lua dzVents script handling onStop event which executes the script.

### Python Script

Source: gpio\_cleanup.python\_event

```
#!/usr/bin/python3
"""

Push On switch to cleanup GPIO ports. All ports are released. Any LEDs are turned off.
This scrip only works if the ports have been setup by other automation event scripts.
Trigger: Device
131,VirtualDevices,GPIO Cleanup,Light/Switch,Switch,Off
Domoticz Log:
2020-06-10 09:57:31.752 EVENT: explore_gpio_cleanup
2020-06-10 09:57:31.752 131,GPIO Cleanup,1,0
2020-06-10 09:57:31.752 RPi.GPIO ports cleaned up.
20200610 rwbl
"""

import domoticz2
import DomoticzEvents as DE
import RPi.GPIO as GPIO

# Define the idx of the switch turning led on/off
IDX_SWITCH = 131

def tolog(msg):
    domoticz2.log(msg)

# print ("This will only show up in the shell where you start domoticz");
# print("EVENT: explore_gpio_led")
msg = "EVENT: explore_gpio_cleanup"
tolog(msg)

deviceobj = DE.changed_device
devicename = deviceobj.name
deviceid = deviceobj.id

if deviceid == IDX_SWITCH:
    # domoticz.log("Device changed: Name={}, idx={}".format(devicename, deviceid))
    # get the values
    msg = "{} ,{} ,{} ,{}".format(deviceobj.id, deviceobj.name, deviceobj.n_value, deviceobj.s_value)
    tolog(msg)
    try:
        GPIO.cleanup()
        msg = "RPi.GPIO ports cleaned up."
        tolog(msg)
    except:
        msg = "RPi.GPIO ports cannot be cleaned up. Possibly not used."
        tolog(msg)
```

## LED On|Off Command Line

### Purpose

Turn an LED ON or OFF via script with command line argument ON | OFF.  
This experiment purpose is to explore how to use a GPIO output port.

### Python Script

Source: gpio\_ledonoff.py

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
"""

gpio_ledonoff.py
Turn an LED, connected to GPIO18, on & off via the command line.

Wiring:
LED = RPi
Signal = Pin #12 - GPIO18
GND = Pin #6 - GND
Dependencies:
RPi.GPIO library
Run:
python3 gpio_ledonoff.py ON
Number of arguments: 2 arguments.
Argument List: ['gpio_ledonoff.py', 'ON']
LED State: ON
"""

import RPi.GPIO as GPIO
import sys

# GPIO pin number
GPIOPIN = 18
# LED object
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(GPIOPIN,GPIO.OUT)

def switch_led(argv):
    print('Number of arguments:', len(sys.argv), 'arguments.')
    if (len(sys.argv) < 2):
        print("Wrong number of arguments.")
        return
    print('Argument List:', str(sys.argv))
    state = sys.argv[1].upper()
    if (state == "ON"):
        GPIO.output(GPIOPIN,GPIO.HIGH)
    elif (state == "OFF"):
        GPIO.output(GPIOPIN,GPIO.LOW)
    else:
        state = "UNKNOWN"
    print("LED State: ",state)

if __name__ == "__main__":
    switch_led(sys.argv[2:])
    quit()
```

## LED On|Off Device Action

### Purpose

A Domoticz switch (Virtual Sensor) to turn an LED On or Off via device On|Off actions.

### Domoticz Device

Create Switch Device (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
128, VirtualDevices, GPIO-LED, Light/Switch, Switch, On/Off, Off

### Python Script

The previous script **gpio\_ledonoff.py** is used.

The script **gpio\_ledonoff.py** is placed in the RPi folder:

```
/home/pi/domoticz/scripts/python
```

The switch properties actions defined, to turn the LED On|Off:

On Action	script://python/gpio_ledonoff.py ON
Off Action	script://python/gpio_ledonoff.py OFF

#### Note

URL scheme **script://** refers to folder **/home/pi/domoticz/scripts/** on the RPi (for user pi).

The script is owned by user pi and executable.

If not executable, might need to set "sudo chmod +x gpio\_ledonoff.py".

### Domoticz Log

```
2020-06-09 09:26:05.087 (VirtualDevices) Light/Switch (GPIO-LED)
2020-06-09 09:26:05.081 Status: User: Admin initiated a switch command (128/GPIO-LED/Off)
2020-06-09 09:26:05.300 Status: Executing script: /home/pi/domoticz/scripts/python/gpio_ledonoff.py
2020-06-09 09:26:06.007 (VirtualDevices) Light/Switch (GPIO-LED)
2020-06-09 09:26:06.000 Status: User: Admin initiated a switch command (128/GPIO-LED/On)
2020-06-09 09:26:06.224 Status: Executing script: /home/pi/domoticz/scripts/python/gpio_ledonoff.py
```

## LED On|Off Automation Event Device Trigger

### Purpose

A Domoticz switch to turn an LED on or off triggered by a device change in a Python Automation Event.

The previous script **gpio\_ledonoff.py** has been modified to an Automation Event type Python. The trigger is a device, i.e. Switch device with idx 130.

### Python Script

Source: **gpio\_ledonoff.python\_event**

```
#!/usr/bin/python3
"""
Switch an LED connected to GPIO 18 on/off by using a Domoticz Switch Type On/Off.
The event is running every minute.
If a device has changed, the device id is greater 0
If a device has not changed, the device id is 0
Device object properties.
self.id = id
self.name = name
self.type = type
self.sub_type = sub_type
self.switch_type = switch_type
self.n_value = n_value
self.n_value_string = n_value_string
self.s_value = s_value
self.last_update_string = last_update
"""

import domoticz2
import DomoticzEvents as DE
import RPi.GPIO as GPIO

# Define the idx of the switch turning led on/off
IDX_SWITCH = 130

# GPIO pin number
GPIOPIN = 18
# LED object
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(GPIOPIN,GPIO.OUT)

def tolog(msg):
    domoticz2.log(msg)

def switch_led(state):
    state = state.upper()
    if (state == "ON"):
        GPIO.output(GPIOPIN,GPIO.HIGH)
    elif (state == "OFF"):
        GPIO.output(GPIOPIN,GPIO.LOW)
    else:
        state = "UNKNOWN"
    msg = "LED State: {}".format(state)
    tolog(msg)

# print ("This will only show up in the shell where you start domoticz");
# print("EVENT: explore_gpio_led")
msg = "EVENT: explore_gpio_ledonoff"
tolog(msg)

deviceobj = DE.changed_device
devicename = deviceobj.name
deviceid = deviceobj.id

if deviceid == IDX_SWITCH:
    # domoticz.log("Device changed: Name={}, idx={}".format(devicename, deviceid))
    # get the values
    msg = "{}{},{}{},{}".format(deviceobj.id, deviceobj.name, deviceobj.n_value, deviceobj.s_value)
```

```
tolog(msg)
# state = on (n_value = 0): 2020-06-05 11:17:53.439 128 GPIO-LED 0 0
# state = off (n_value = 1): 2020-06-05 11:18:04.563 128 GPIO-LED 1 0

devicestate = deviceobj.n_value
if devicestate == 0:
    switch_led("OFF")
if devicestate == 1:
    switch_led("ON")

# NOT USED else led will turn off: led.close()
msg = "Device {} turned {}".format(devicename, devicestate)
tolog(msg)
```

## Domoticz Log

```
2020-06-09 10:42:51.883 (VirtualDevices) Light/Switch (GPIO LED ONOFF)
2020-06-09 10:42:51.877 Status: User: Admin initiated a switch command (130/GPIO LED ONOFF/On)
2020-06-09 10:42:52.053 EVENT: explore_gpio_ledonoff
2020-06-09 10:42:52.053 130,GPIO LED ONOFF,1,0
2020-06-09 10:42:52.054 LED State: ON
2020-06-09 10:42:52.054 Device GPIO LED ONOFF turned 1.
2020-06-09 10:42:52.746 (VirtualDevices) Light/Switch (GPIO LED ONOFF)
2020-06-09 10:42:52.906 EVENT: explore_gpio_ledonoff
2020-06-09 10:42:52.906 130,GPIO LED ONOFF,0,0
2020-06-09 10:42:52.906 LED State: OFF
2020-06-09 10:42:52.906 Device GPIO LED ONOFF turned 0.
2020-06-09 10:42:52.739 Status: User: Admin initiated a switch command (130/GPIO LED ONOFF/Off)
2020-06-09 10:42:55.449 (VirtualDevices) Light/Switch (GPIO LED ONOFF)
```

## LED On|Off Push-Button Automation Event

### Purpose

When the push-button is pressed or released, a Python Automation Event turns the LED On|Off and updates a Domoticz text device.

### Python Script

Source: [gpio\\_btnledonoff.python\\_event](#)

```
#!/usr/bin/python3
"""

Switch an LED connected to GPIO 18 on/off by using a push-button connected to GPIO 23.
Wait till the event has run first time to setup, then press button.
If the button is pressed or released, the text device GPIO Status is updated.
The event is running every minute - but doing nothing.
"""

import domoticz2
import DomoticzEvents as DE
import datetime
# urllib to update device(s)
import urllib.request
# parse values into the url
import urllib.parse
import RPi.GPIO as GPIO

# GPIO pin numbers
LEDPIN = 18
BUTTONPIN = 23

IDX_GPIO_STATUS = 129

"""

Write a string to the domoticz log
"""

def tolog(msg):
    domoticz2.log(msg)

"""

Get the current date & time in format "YYYYMMDD HH:MM:SS"
"""

def getnow():
    now = datetime.datetime.now()
    return now.strftime("%Y%m%d %X")

"""

Update the text of the device named "GPIO_STATUS"
"""

def updateGPIOStatus(state):
    msg = "Update GPIO Status: {}".format(state)
    tolog(msg)
    # Update GPIO Status: ON
    data = urllib.parse.quote("LED turned {} @ {}".format(state, getnow()))
    # LED%20turned%20ON.
    req = "http://domoticz-
ip:8080/json.htm?type=command&param=udevice&idx={}&nvalue=0&svalue={}".format(IDX_GPIO_STATUS, data)
    tolog(req)
    # http://domoticz-
    ip:8080/json.htm?type=command&param=udevice&idx=129&nvalue=0&svalue=LED%20turned%20OFF%20%40%2020200609
    %2014%3A52%3A14.
    resp = urllib.request.urlopen(req)
    respData = resp.read().decode('utf-8')
    msg = "Update GPIO Status: {}".format(respData)
    tolog(msg)
    # Update GPIO Status: b'\\n\\t"status" : "OK", \\n\\t"title" : "Update Device"\\n\\n'
    msg = "Update GPIO Status: done"
    tolog(msg)
    return

# Define handling button pressed & released
def button_callback(channel):
```

```

state = "UNKNOWN"
if GPIO.input(BUTTONPIN) == GPIO.HIGH:
    GPIO.output(LEDPIN,GPIO.HIGH)
    state = "ON"
if GPIO.input(BUTTONPIN) == GPIO.LOW:
    GPIO.output(LEDPIN,GPIO.LOW)
    state = "OFF"
msg = "Button was pressed. LED changed: {}".format(state)
tolog(msg)
updateGPIOStatus(state)

def setup():
    # GPIO mode & warnings
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    # LED and BUTTON objects
    try:
        GPIO.setup(LEDPIN,GPIO.OUT)
        GPIO.setup(BUTTONPIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        GPIO.add_event_detect(BUTTONPIN,GPIO.BOTH,callback=button_callback)
        GPIO.output(LEDPIN,GPIO.LOW)
    except:
        # do nothing
        # tolog("Setup already done")
        dummy = 1

# Event is running every minute
# Setup is only required firsttime, this is handled by try-except block in function setup
# Consider using GPIO.cleanup()
setup()

```

## Domoticz Log

```

2020-06-10 08:51:19.117 Button was pressed. LED changed: ON
2020-06-10 08:51:19.117 Update GPIO Status: ON
2020-06-10 08:51:19.126 http://domoticz-
ip:8080/json.htm?type=command&param=udevice&idx=129&nvalue=0&svalue=LED%20turned%20ON%20%40%2020200610%
2008%3A51%3A19.
2020-06-10 08:51:19.222 Update GPIO Status: {
2020-06-10 08:51:19.222 "status" : "OK",
2020-06-10 08:51:19.222 "title" : "Update Device"
2020-06-10 08:51:19.222 }
2020-06-10 08:51:19.222
2020-06-10 08:51:19.222 Update GPIO Status: done
2020-06-10 08:51:19.420 Button was pressed. LED changed: OFF
2020-06-10 08:51:19.420 Update GPIO Status: OFF
2020-06-10 08:51:19.421 http://domoticz-
ip:8080/json.htm?type=command&param=udevice&idx=129&nvalue=0&svalue=LED%20turned%20OFF%20%40%2020200610%
2008%3A51%3A19.
2020-06-10 08:51:19.436 Update GPIO Status: {
2020-06-10 08:51:19.436 "status" : "OK",
2020-06-10 08:51:19.436 "title" : "Update Device"
2020-06-10 08:51:19.436 }
2020-06-10 08:51:19.436
2020-06-10 08:51:19.436 Update GPIO Status: done

```

## LED On|Off Push-Button Plugin

### Purpose

When a push-button (GPIO 23) is pressed or released, a Domoticz plugin turns an LED On|Off (GPIO 18) and updates a Domoticz text device.

### The plugin

- is installed in folder /home/pi/domoticz/plugins/gpiobtnledonoff, file plugin.py.
- is listed as Domoticz hardware "RPi.GPIO Push-Button LED OnOff".
- creates a text device to log the LED state changes.

### Installing

- Create folder:  
/home/pi/domoticz/plugins/gpiobtnledonoff
- Copy file gpio\_btnledonoff.plugin to  
/home/pi/domoticz/plugins/gpiobtnledonoff/plugin.py
- Restart Domoticz:  
sudo service domoticz.sh restart
- Domoticz GUI > Setup > Hardware > Type: "RPi.GPIO Push-Button LED OnOff" >  
Name: RPiGPIO (as an example) > Check LED & Button GPIO pins > Debug: True >  
Add.
- Check the Domoticz log for errors.

### Hardware Configuration

The screenshot shows the Domoticz Hardware Configuration page. At the top, there is a table listing a single entry:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
7	RPiGPIO	Yes	RPi.GPIO Push-Button LED OnOff			Disabled

Below the table, it says "Showing 1 to 1 of 1 entries (filtered from 3 total entries)". There are "Update" and "Delete" buttons. The main configuration area for the RPi.GPIO plugin includes:

- Enabled:**
- Name:** RPiGPIO
- Type:** RPi.GPIO Push-Button LED OnOff
- Data Timeout:** Disabled
- Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.  
Do not enable this option for devices that do not receive data!
- RPi.GPIO Push-Button LED OnOff**  
Explore the Raspberry Pi RPi.GPIO library.  
This plugin
  - explores how to use the GPIO input (including callback handling) and output.
  - is a base template for exploring further plugin solutions.
- Features**
  - Components: LED (output) connected to GPIO18 (default), Push-Button (input) connected to GPIO23 (default).
  - Creates the GPIO pins and set the GPIO numbering mode to BCM.
  - By pressing the push-button, the LED state changes to ON and a Domoticz Text Device gets updated.
  - By releasing the push-button, the LED state changes to OFF and a Domoticz Text Device gets updated.
- Devices**
  - GPIO Status - Text device logging the LED state change.
- Configuration**  
Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use BCM format, i.e. GPIOnn (checkout for the board used).
  - LED GPIO Pin Number (default: 18)
  - Button GPIO Pin Number (default: 23)
- LED GPIO:** 18
- Button GPIO:** 23
- Debug:** True
- Add** button

### Text Device created by the plugin and logging the LED state change

The screenshot shows the Domoticz Text Device list. It lists one device:

Idx	Hardware	ID	Unit	Name	Type	SubType
132	RPiGPIO	00070001	1	RPiGPIO - GPIO Status	General	Text

In the "Text" column, it shows "LED turned OFF @ 20200611 09:50:25."

## Python Script

Source: gpio\_btnledonoff.plugin

```
#!/usr/bin/python3
"""

<plugin key="gpiobtnledonoff" name="RPi.GPIO Push-Button LED OnOff" author="rwbl" version="0.5.0"
wikilink="" externallink=""
<description>
    <h2>RPi.GPIO Push-Button LED OnOff</h2>
    Explore the Raspberry Pi RPi.GPIO library.<br/> This plugin
    <ul style="list-style-type:square">
        <li>explores how to use the GPIO input (including callback handling) and output.</li>
        <li>is a base template for exploring further plugin solutions.</li>
    </ul>
    <h3>Features</h3>
    <ul style="list-style-type:square">
        <li>Components: LED (output) connected to GPIO18 (default), Push-Button (input) connected
        to GPIO23 (default).</li>
            <li>Creates the GPIO pins and set the GPIO numbering mode to BCM.</li>
            <li>By pressing the push-button, the LED state changes to ON and a Domoticz Text Device
        gets updated.</li>
            <li>By releasing the push-button, the LED state changes to OFF and a Domoticz Text Device
        gets updated.</li>
    </ul>
    <h3>Devices</h3>
    <ul style="list-style-type:square">
        <li>GPIO Status - Text device logging the LED state change.</li>
    </ul>
    <h3>Configuration</h3>
    Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use
    BCM format, i.e. GPIOnn (checkout for the board used).<br/>
    <ul style="list-style-type:square">
        <li>LED GPIO Pin Number (default: 18)</li>
        <li>Button GPIO Pin Number (default: 23)</li>
    </ul>
</description>
<params>
    <param field="Mode1" label="LED GPIO" width="75px" required="true" default="18"/>
    <param field="Mode2" label="Button GPIO" width="75px" required="true" default="23"/>
    <param field="Mode6" label="Debug" width="75px">
        <options>
            <option label="True" value="Debug" default="true"/>
            <option label="False" value="Normal"/>
        </options>
    </param>
</params>
</plugin>
"""

import Domoticz
import os
import datetime
import RPi.GPIO as GPIO

# Set the plugin version
PLUGINVERSION = "v0.5.0"
PLUGINSHORTDESCRIPTON = "RPi.GPIO Push-Button LED OnOff"

# Units for the devices used
UNITGPIOSTATUS = 1 # text device

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2
        self.ledpin = 0
        self.buttonpin = 0
        return

    def onStart(self):
        Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
```

```

Domoticz.Debug("onStart called")
Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

#Check debug mode
if Parameters["Mode6"] == "Debug":
    self.debug = True
    Domoticz.Debugging(1)
    dump_config_to_log()

#Setting up GPIO, i.e. get the pin numbers, set mode BCM, warnings off, set the pin mode and
events
    self.ledpin = int(Parameters["Mode1"])
    self.buttonpin = int(Parameters["Mode2"])
    # Domoticz.Debug("Setting up GPIO:LED=" + Parameters["Mode1"] + ", Button=" +
Parameters["Mode2"])
    Domoticz.Debug("Setting up GPIO:LED=" + str(self.ledpin) + ", Button=" + str(self.buttonpin))
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
try:
    GPIO.setup(self.ledpin, GPIO.OUT)
    GPIO.setup(self.buttonpin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    # Add the event handling self-object.
    # Ensure onButtonCallback is in the same class and has self, channel as signature
    GPIO.add_event_detect(self.buttonpin,GPIO.BOTH,callback=self.onButtonCallback)
    GPIO.output(self.ledpin,GPIO.LOW)
    Domoticz.Debug("Setting up GPIO: Done")
except:
    Domoticz.Error("Error setting up GPIO pins.")

#Create device if needed
# Domoticz.Debug("Devices:" + str(len(Devices)) )
if (len(Devices) == 0):
    Domoticz.Debug("Creating device(s)")
    Domoticz.Device(Name="GPIO Status", Unit=UNITGPIOSTATUS, TypeName="Text", Used=1).Create()
    Domoticz.Debug("Device created: "+Devices[UNITGPIOSTATUS].Name)

def onStop(self):
    Domoticz.Debug("onStop called: Cleaning up GPIO (IMPORTANT)")
    GPIO.cleanup()

def onConnect(self, Connection, Status, Description):
    Domoticz.Debug("onConnect called")

def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called")

def onCommand(self, Unit, Command, Level, Hue):
    Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Parameter '" + str(Command) + "'",
Level: " + str(Level))

def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
    Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," +
str(Priority) + "," + Sound + "," + ImageFile)

def onDeviceModified(self, Unit):
    Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))

def onDisconnect(self, Connection):
    Domoticz.Debug("onDisconnect called")

def onHeartbeat(self):
    Domoticz.Debug("onHeartbeat called")

"""

Handle button changes. Button pressed = LED on, released = LED off.
This function must be in the same class as onStart because it uses the self-object to access the
pins.

"""
def onButtonCallback(self, channel):
    Domoticz.Debug("onButtonCallback called")
    Domoticz.Debug("Pins:" + str(self.ledpin) + "," + str(self.buttonpin))
    state = "UNKNOWN"
    if GPIO.input(self.buttonpin) == GPIO.HIGH:
        GPIO.output(self.ledpin,GPIO.HIGH)
        state = "ON"

```

```

if GPIO.input(self.buttonpin) == GPIO.LOW:
    GPIO.output(self.ledpin,GPIO.LOW)
    state = "OFF"
Domoticz.Debug("Button was pressed. LED changed: {}".format(state))
isnow = datetime.datetime.now().strftime("%Y%m%d %X")
msg = "LED turned {} @ {}".format(state, isnow)
Devices[UNITGPIOSTATUS].Update(nValue=0, sValue=msg)
Domoticz.Log(msg)
Domoticz.Debug("onButtonCallback done")

global _plugin
_plugin = BasePlugin()

def onStart():
    global _plugin
    _plugin.onStart()

def onStop():
    global _plugin
    _plugin.onStop()

def onConnect(Connection, Status, Description):
    global _plugin
    _plugin.onConnect(Connection, Status, Description)

def onMessage(Connection, Data):
    global _plugin
    _plugin.onMessage(Connection, Data)

def onCommand(Unit, Command, Level, Hue):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Hue)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Generic helper functions
"""
Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:
        if Parameters[x] != "":
            Domoticz.Log( " " + x + ":" + str(Parameters[x]) + " ")
    Domoticz.Debug("Device count: " + str(len(Devices)))
    for x in Devices:
        Domoticz.Log("Device: " + str(x) + " - " + str(Devices[x]))
        Domoticz.Log("Device ID: " + str(Devices[x].ID) + " ")
        Domoticz.Log("Device Name: " + Devices[x].Name + " ")
        Domoticz.Log("Device nValue: " + str(Devices[x].nValue))
        Domoticz.Log("Device sValue: " + Devices[x].sValue + " ")
        Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
    return

```

## Domoticz Log

```
2020-06-11 09:50:24.745 (RPiGPIO) onButtonCallback called
2020-06-11 09:50:24.745 (RPiGPIO) Pins:18,23
2020-06-11 09:50:24.745 (RPiGPIO) Button was pressed. LED changed: ON
2020-06-11 09:50:24.747 (RPiGPIO - GPIO Status) Updating device from 0:'Hello World!' to have values
0:'LED turned ON @ 20200611 09:50:24.'.
2020-06-11 09:50:24.786 (RPiGPIO) LED turned ON @ 20200611 09:50:24.
2020-06-11 09:50:24.786 (RPiGPIO) onButtonCallback done
2020-06-11 09:50:25.029 (RPiGPIO) onButtonCallback called
2020-06-11 09:50:25.029 (RPiGPIO) Pins:18,23
2020-06-11 09:50:25.030 (RPiGPIO) Button was pressed. LED changed: OFF
2020-06-11 09:50:25.030 (RPiGPIO - GPIO Status) Updating device from 0:'LED turned ON @ 20200611
09:50:24.' to have values 0:'LED turned OFF @ 20200611 09:50:25.'.
2020-06-11 09:50:25.051 (RPiGPIO) LED turned OFF @ 20200611 09:50:25.
2020-06-11 09:50:25.051 (RPiGPIO) onButtonCallback done
```

## Push-Button Plugin

### Purpose

When a push-button (GPIO23) is pressed, the Domoticz plugin checks its alert level and actions accordingly.

If the alert level is

- 1 (green), the alert level is set to 4 (red) and an LED (GPIO18) is turned on
- 4 (red), the alert level is set to 1 (green) and an LED (GPIO18) is turned off
- a dzVents Automation script logs the alert device changes

The plugin

- is installed in folder /home/pi/domoticz/plugins/gpiobtntrigger, file plugin.py.
- is listed as Domoticz hardware "RPi.GPIO Push-Button LED OnOff".
- creates a text device to log the LED state changes.

### Installing

- Create folder:  
/home/pi/domoticz/plugins/gpiobtntrigger
- Copy file gpio\_btndriver.plugin to  
/home/pi/domoticz/plugins/gpiobtntrigger/plugin.py
- Restart Domoticz:  
sudo service domoticz.sh restart
- Domoticz GUI > Setup > Hardware > Type: "RPi.GPIO Push-Button Trigger" >  
Name: RPiGPIO (as an example) > Check LED & Button GPIO pins > Debug: True >  
Add.
- Check the Domoticz log for errors.

### Hardware Configuration

The screenshot shows the Domoticz hardware configuration page for the RPi.GPIO Push-Button Trigger device. The device is listed in the table with the following details:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
7	RPiGPIO	Yes	RPi.GPIO Push-Button Trigger			Disabled

Below the table, the device details are shown:

- Enabled:**
- Name:** RPiGPIO
- Type:** RPi.GPIO Push-Button Trigger
- Data Timeout:** Disabled

A note states: "Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Do not enable this option for devices that do not receive data!"

**RPi.GPIO Push-Button Trigger**  
 Explore the Raspberry Pi RPi.GPIO library.  
 This plugin
 

- explores how to use the GPIO input (including callback handling) and output.
- is a base template for exploring further plugin solutions.

**Features**

- Components: Push-Button (input) connected to GPIO23 (default), LED (output) connected to GPIO18 (default).
- Creates the GPIO pins and set the GPIO numbering mode to BCM.
- By pressing the push-button, the LED state changes to ON or OFF (depending LED state) and a Domoticz Alert Device gets updated.
- When the Domoticz Alert Device gets updated, a dzVents script is executed.

**Devices**

- GPIO Status - Text device logging the LED state change.

**Configuration**  
 Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use BCM format, i.e. GPIOnn (checkout for the board used).
 

- LED GPIO Pin Number (default: 18)
- Button GPIO Pin Number (default: 23)

Configuration settings:
 

- LED GPIO: 18
- Button GPIO: 23
- Debug: True

**Add**

## Alert Device created by the plugin and logging the LED state change

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
<input type="checkbox"/>	132	RPIGPIO	00070001	1	RPIGPIO - GPIO Alert	General	Alert	LED state ON (Alert level 4) @ 20200611 11:19:04.

## Python Script

Source: gpio\_btntrigger.plugin

```
#!/usr/bin/python3
"""

<plugin key="gpiobtntrigger" name="RPi.GPIO Push-Button Trigger" author="rwbl" version="0.5.0"
wikilink="" externallink=""
<description>
    <h2>RPi.GPIO Push-Button Trigger</h2>
    Explore the Raspberry Pi RPi.GPIO library.<br/> This plugin
    <ul style="list-style-type:square">
        <li>explores how to use the GPIO input (including callback handling) and output.</li>
        <li>is a base template for exploring further plugin solutions.</li>
    </ul>
    <h3>Features</h3>
    <ul style="list-style-type:square">
        <li>Components: Push-Button (input) connected to GPIO23 (default), LED (output) connected
        to GPIO18 (default).</li>
        <li>Creates the GPIO pins and set the GPIO numbering mode to BCM.</li>
        <li>By pressing the push-button, the LED state changes to ON or OFF (depending LED state)
        and a Domoticz Alert Device gets updated.</li>
        <li>When the Domoticz Alert Device gets updated, a dzVents script is executed.</li>
    </ul>
    <h3>Devices</h3>
    <ul style="list-style-type:square">
        <li>GPIO Status - Text device logging the LED state change.</li>
    </ul>
    <h3>Configuration</h3>
    Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use
    BCM format, i.e. GPIOnn (checkout for the board used).<br/>
    <ul style="list-style-type:square">
        <li>LED GPIO Pin Number (default: 18)</li>
        <li>Button GPIO Pin Number (default: 23)</li>
    </ul>
</description>
<params>
    <param field="Mode1" label="LED GPIO" width="75px" required="true" default="18"/>
    <param field="Mode2" label="Button GPIO" width="75px" required="true" default="23"/>
    <param field="Mode6" label="Debug" width="75px">
        <options>
            <option label="True" value="Debug" default="true"/>
            <option label="False" value="Normal"/>
        </options>
    </param>
</params>
</plugin>
"""

import Domoticz
import os
import datetime
import RPi.GPIO as GPIO

# Set the plugin version
PLUGINVERSION = "v0.5.0"
PLUGINSHORTDESCRIPTON = "RPi.GPIO Push-Button LED OnOff"

# Units for the devices used
UNITALERT = 1 # alert device

# Alert Levels
ALERTON = 4      # red
ALERTOFF = 1     # green

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2 (see below).
        self.ledpin = 0      # default = 18
```

```

    self.buttonpin = 0 # default = 23
    # Define the alert state as set by the alertlevels
    # This will be used to turn the led on (alert level ALERTON) or off (alert level ALERTOFF)
    self.alertlevel = ALERTOFF
    return

def onStart(self):
    Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
    Domoticz.Debug("onStart called")
    Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

    #Check debug mode
    if Parameters["Mode6"] == "Debug":
        self.debug = True
        Domoticz.Debugging(1)
        dump_config_to_log()

    #Setting up GPIO, i.e. get the pin numbers, set mode BCM, warnings off, set the pin mode and
events
    self.ledpin = int(Parameters["Mode1"])
    self.buttonpin = int(Parameters["Mode2"])
    # Domoticz.Debug("Setting up GPIO:LED=" + Parameters["Mode1"] + ", Button=" +
Parameters["Mode2"])
    Domoticz.Debug("Setting up GPIO:LED=" + str(self.ledpin) + ", Button=" + str(self.buttonpin))
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    try:
        GPIO.setup(self.ledpin, GPIO.OUT)
        GPIO.setup(self.buttonpin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        # Add the event handling self-object.
        # Ensure onButtonCallback is in the same class and has self, channel as signature
        GPIO.add_event_detect(self.buttonpin,GPIO.BOTH,callback=self.onButtonCallback)
        GPIO.output(self.ledpin,GPIO.LOW)
        self.alertlevel = ALERTOFF
        Domoticz.Debug("Setting up GPIO: Done")
    except:
        Domoticz.Error("Error setting up GPIO pins.")

    #Create device if needed
    # Domoticz.Debug("Devices:" + str(len(Devices)) )
    if (len(Devices) == 0):
        Domoticz.Debug("Creating device(s)")
        Domoticz.Device(Name="GPIO Alert", Unit=UNITALERT, TypeName="Alert", Used=1).Create()
        # nValue = Level(0=gray, 1=green, 2=yellow, 3=orange, 4=red), sValue = TEXT (to display)
        Devices[UNITALERT].Update(nValue=ALERTOFF, sValue="LED OFF")
        Domoticz.Debug("Device created: "+Devices[UNITALERT].Name)

def onStop(self):
    Domoticz.Debug("onStop called: Cleaning up GPIO (IMPORTANT)")
    GPIO.cleanup()

def onConnect(self, Connection, Status, Description):
    Domoticz.Debug("onConnect called")

def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called")

def onCommand(self, Unit, Command, Level, Hue):
    Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Parameter '" + str(Command) + "',"
Level: " + str(Level))

def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
    Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," +
str(Priority) + "," + Sound + "," + ImageFile)

def onDeviceModified(self, Unit):
    Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))

def onDisconnect(self, Connection):
    Domoticz.Debug("onDisconnect called")

def onHeartbeat(self):
    Domoticz.Debug("onHeartbeat called")

...

```

```

Handle button changes.
Button pressed (GPIO.HIGH) = if LED on, then LED off v.v.
Button released is not used.
This function must be in the same class as onStart because it uses the self-object to access the
pins.
"""
def onButtonCallback(self, channel):
    Domoticz.Debug("onButtonCallback called")
    if GPIO.input(self.buttonpin) == GPIO.HIGH:
        # Check the alert level and change the led state to on or off
        # The alert level is off > set to on and turn LED ON
        if self.alertlevel == ALERTOFF:
            GPIO.output(self.ledpin,GPIO.HIGH)
            self.alertlevel = ALERTON
            ledstate = "ON"
        elif self.alertlevel == ALERTON:
            # The alert level is on > set to off and turn LED OFF
            GPIO.output(self.ledpin,GPIO.LOW)
            self.alertlevel = ALERTOFF
            ledstate = "OFF"
        else:
            ledstate = "UNKNOWN"
    Domoticz.Debug("Button was pressed. LED changed: {}, Alert Level: {}".format(ledstate,
self.alertlevel))
    isnow = datetime.datetime.now().strftime("%Y%m%d %X")
    msg = "LED state {} (Alert level {}) @ {}".format(ledstate, self.alertlevel, isnow)
    Devices[UNITALERT].Update(nValue=self.alertlevel, sValue=msg)
    Domoticz.Log(msg)
    Domoticz.Debug("onButtonCallback done")

global _plugin
_plugin = BasePlugin()

def onStart():
    global _plugin
    _plugin.onStart()

def onStop():
    global _plugin
    _plugin.onStop()

def onConnect(Connection, Status, Description):
    global _plugin
    _plugin.onConnect(Connection, Status, Description)

def onMessage(Connection, Data):
    global _plugin
    _plugin.onMessage(Connection, Data)

def onCommand(Unit, Command, Level, Hue):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Hue)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Generic helper functions
"""
Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:

```

```

if Parameters[x] != "":
    Domoticz.Log( "''' + x + "'':" + str(Parameters[x]) + '''")
    Domoticz.Debug("Device count: " + str(len(Devices)))
for x in Devices:
    Domoticz.Log("Device:      " + str(x) + " - " + str(Devices[x]))
    Domoticz.Log("Device ID:   '" + str(Devices[x].ID) + "'")
    Domoticz.Log("Device Name: '" + Devices[x].Name + "'")
    Domoticz.Log("Device nValue: " + str(Devices[x].nValue))
    Domoticz.Log("Device sValue: '" + Devices[x].sValue + "'")
    Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
return

```

## Domoticz Log

```

2020-06-11 11:30:52.310 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:52.310 (RPiGPIO) Button was pressed. LED changed: OFF, Alert Level: 1
2020-06-11 11:30:52.310 (RPiGPIO - GPIO Alert) Updating device from 4:'LED state ON (Alert level 4) @ 20200611 11:19:04.' to have values 1:'LED state OFF (Alert level 1) @ 20200611 11:30:52.'.
2020-06-11 11:30:52.327 (RPiGPIO) LED state OFF (Alert level 1) @ 20200611 11:30:52.
2020-06-11 11:30:52.327 (RPiGPIO) onButtonCallback done
2020-06-11 11:30:52.459 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:52.460 (RPiGPIO) onButtonCallback done
2020-06-11 11:30:53.832 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:53.832 (RPiGPIO) Button was pressed. LED changed: ON, Alert Level: 4
2020-06-11 11:30:53.832 (RPiGPIO - GPIO Alert) Updating device from 1:'LED state OFF (Alert level 1) @ 20200611 11:30:52.' to have values 4:'LED state ON (Alert level 4) @ 20200611 11:30:53.'.
2020-06-11 11:30:53.849 (RPiGPIO) LED state ON (Alert level 4) @ 20200611 11:30:53.
2020-06-11 11:30:53.850 (RPiGPIO) onButtonCallback done

```

## dzVents Script

Example of a dzVents script acting if the alert level of the Alert Level device changes.

### Source: gpio\_btntrigger.dzvents

```

-- Handle changes of the alert level device
local IDX_ALERT = 132
local ALERTON = 4
local ALERTOFF = 1

return {
    on = {
        devices = {
            IDX_ALERT
        }
    },
    execute = function(domoticz, device)
        domoticz.log('Device ' .. device.name .. ' was changed', domoticz.LOG_INFO)
        domoticz.log(string.format('Alert Level: %d, Text: %s', domoticz.devices(IDX_ALERT).color,
        domoticz.devices(IDX_ALERT).text))
        -- Action depending Level
        level = domoticz.devices(IDX_ALERT).color
        if (level == ALERTON) then
            domoticz.log('Alert is ON, please take action!')
        end
        if (level == ALERTOFF) then
            domoticz.log('Alert is OFF, No action required.')
        end
    end
}

```

## Domoticz Log

The alert state was ALERTON. Pressing the button, changes the state to ALERTOFF. By pressing again, the state changes to ALERTON.

```

2020-06-11 11:43:40.801 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.801 (RPiGPIO) Button was pressed. LED changed: OFF, Alert Level: 1

```

```
2020-06-11 11:43:40.801 (RPiGPIO - GPIO Alert) Updating device from 4:'LED state ON (Alert level 4) @ 20200611 11:39:31.' to have values 1:'LED state OFF (Alert level 1) @ 20200611 11:43:40.'.
2020-06-11 11:43:40.820 (RPiGPIO) LED state OFF (Alert level 1) @ 20200611 11:43:40.
2020-06-11 11:43:40.820 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.968 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED state OFF (Alert level 1) @ 20200611 11:43:40."
2020-06-11 11:43:40.968 Status: dzVents: Info: ----- Start internal script: gpio_btntrigger: Device: "RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-11 11:43:40.968 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-11 11:43:40.968 Status: dzVents: Info: Alert Level: 1, Text: LED state OFF (Alert level 1) @ 20200611 11:43:40.
2020-06-11 11:43:40.968 Status: dzVents: Info: Alert is OFF, No action required.
2020-06-11 11:43:40.968 Status: dzVents: Info: ----- Finished gpio_btntrigger
2020-06-11 11:43:42.319 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:42.320 (RPiGPIO) Button was pressed. LED changed: ON, Alert Level: 4
2020-06-11 11:43:42.320 (RPiGPIO - GPIO Alert) Updating device from 1:'LED state OFF (Alert level 1) @ 20200611 11:43:40.' to have values 4:'LED state ON (Alert level 4) @ 20200611 11:43:42.'.
2020-06-11 11:43:42.343 (RPiGPIO) LED state ON (Alert level 4) @ 20200611 11:43:42.
2020-06-11 11:43:42.343 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:42.492 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:42.492 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:42.487 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED state ON (Alert level 4) @ 20200611 11:43:42."
2020-06-11 11:43:42.488 Status: dzVents: Info: ----- Start internal script: gpio_btntrigger: Device: "RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-11 11:43:42.488 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-11 11:43:42.488 Status: dzVents: Info: Alert Level: 4, Text: LED state ON (Alert level 4) @ 20200611 11:43:42.
2020-06-11 11:43:42.488 Status: dzVents: Info: Alert is ON, please act!
2020-06-11 11:43:42.488 Status: dzVents: Info: ----- Finished gpio_btntrigger
```

## GPIO Zero

To explore how to use the Raspberry Pi GPIO Zero library in Domoticz Python plugins.

This chapter builds upon the chapter [RPi.GPIO](#) with the focus on creating plugins.

[GPIO Zero](#) is a

- simple interface to GPIO devices with Raspberry Pi
- wrapper around the low-level RPi.GPIO library.

Abbreviations: RPi = Raspberry Pi, CLI = Command Line Interface, Python = Python 3

## Prepare

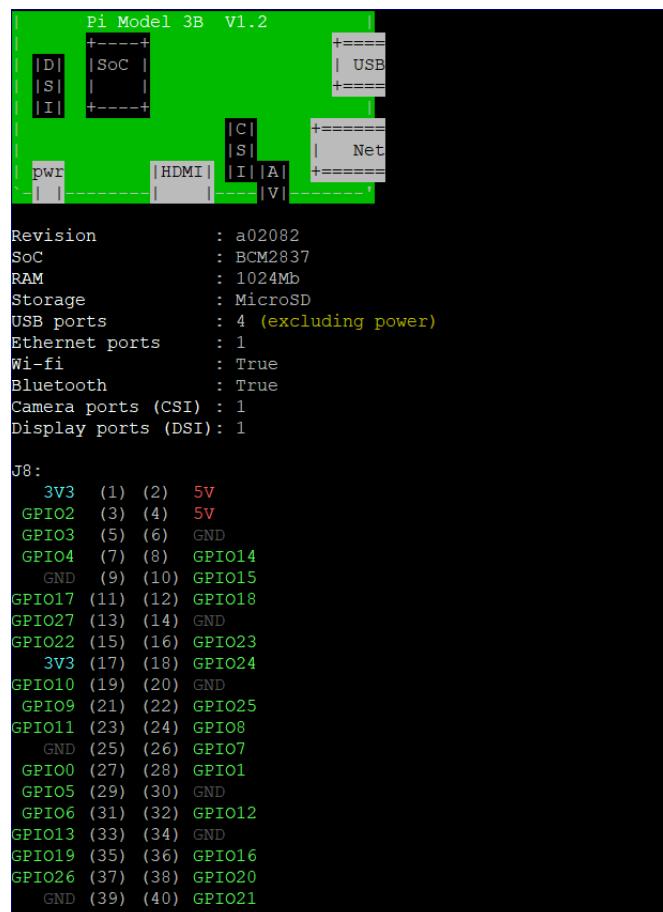
### Version & Path & Update

```
sudo pip3 install gpiozero

Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: gpiozero in /usr/lib/python3/dist-packages (1.5.1)
```

## GPIO pinout

pinout



For further information, please refer to <https://pinout.xyz/>

## Experiments

The GPIOZero experiments are enabled through Python Plugins.

1. Push-Button Plugin  
Switch an LED On|Off via Push-Button.
2. Push-Button Device Plugin  
Switch an LED On|Off via Push-Button & Domoticz Switch.
3. RGB LED Plugin  
Set the color of a RGB LED to Green, Yellow, Orange, Red or White via Domoticz Switch.

## Parts List

- 1x Raspberry Pi 3B V1.2
- 1x DFRobot Digital Piranha Blue LED Module V2
- 1x DFRobot Digital Push Button Red V2
- 1x DFRobot RGB LED Breakout Board (3528)
- Breadboard & various wires

*Note*

This is an overview – for details checkout the experiment description.

## Software

- Raspberry Pi OS - Raspbian GNU/Linux 10 (buster)
- Python 3.7.3, GCC 8.3.0
- GPIOZero library 1.5.1
- Domoticz Beta channel 2020.2 (build 12143)

*Note*

Software versions are subject to change.

## Push-Button Plugin

### Purpose

When a push-button is pressed, the Domoticz plugin checks its alert level and act accordingly.

If the alert level is

- 1 (green), the alert level is changed to 4 (red) and the LED is turned on
- 4 (red), the alert level is changed to 1 (green) and the LED is turned off
- a dzVents Automation script logs the alert device changes

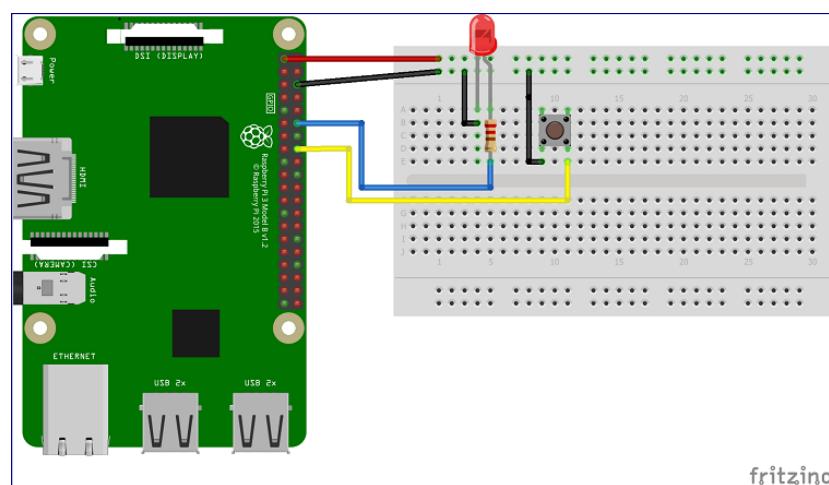
### Parts List

- 1x Raspberry Pi 3B V1.2
- 1x DFRobot Digital Piranha Blue LED Module V2
- 1x DFRobot Digital Push Button Red V2
- 1x Tinkerforge Mounting Plate 12x6cm & 3mm screws
- 1x Breadboard & various wires

### Wiring

<b>LED</b>	<b>Raspberry Pi</b>
Signal (long leg)	Pin #12 GPIO18
GND	Pin #6 GND
<b>Push-Button</b>	<b>Raspberry Pi</b>
Signal	Pin #16 GPIO23
GND	Pin #6 GND
<i>Note</i>	The pulldown resistor is enabled, therefore no additional resistor used in the circuit.

### Circuit



## Plugin

- Installed in folder /home/pi/domoticz/plugins/gpiozeroobtntrigger, file plugin.py.
- Listed as Domoticz hardware "GPIOZero Push-Button Trigger".
- Creates an alert device to indicate & log the LED state changes.

### Installing

Create folder:

```
/home/pi/domoticz/plugins/gpiozeroobtntrigger
```

### Copy file

```
gpiozero_btndtrigger.plugin to /home/pi/domoticz/plugins/gpiozeroobtntrigger/plugin.py
```

Restart Domoticz:

```
sudo service domoticz.sh restart
```

Domoticz GUI > Setup > Hardware > Type: "GPIOZero Push-Button Trigger" > Name: RPiGPIO (as an example) > Check LED & Button GPIO pins > Debug: True > Add.

Check the Domoticz log for errors.

### Python Script

Source: gpiozero\_btndtrigger.plugin

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# domoticz-home-automation-workbook - explore GPIOZero
# WARNING: ensure the plugin key is unique across all plugins.
# author: rwbl
"""

<plugin key="gpiozeroobtntrigger" name="GPIOZero Push-Button Trigger" author="rwbl" version="0.5.0"
wikilink="" externallink="">
    <description>
        <h2>GPIOZero Push-Button Trigger</h2>
        Explore the Raspberry Pi GPIOZero library.<br/> This plugin
        <ul style="list-style-type:square">
            <li>explores how to use the GPIO input (including callback handling) and output.</li>
            <li>is a base template for exploring further plugin solutions.</li>
        </ul>
        <h3>Features</h3>
        <ul style="list-style-type:square">
            <li>Components: Push-Button (input) connected to GPIO23 (default), LED (output) connected
to GPIO18 (default).</li>
                <li>Creates the GPIO pins and set the GPIO numbering mode to BCM.</li>
                <li>By pressing the push-button, the LED state changes to ON or OFF (depending LED state)
and a Domoticz Alert Device gets updated.</li>
                    <li>When the Domoticz Alert Device gets updated, a dzVents script is executed.</li>
                </ul>
            <h3>Devices</h3>
            <ul style="list-style-type:square">
                <li>GPIO Status - Text device logging the LED state change.</li>
            </ul>
            <h3>Configuration</h3>
            Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use
BCM format, i.e. GPIOnn (checkout for the board used).<br/>
            <ul style="list-style-type:square">
                <li>LED GPIO Pin Number (default: 18)</li>
                <li>Button GPIO Pin Number (default: 23)</li>
            </ul>
        </description>
        <params>
            <param field="Mode1" label="LED GPIO" width="75px" required="true" default="18"/>
            <param field="Mode2" label="Button GPIO" width="75px" required="true" default="23"/>
            <param field="Mode6" label="Debug" width="75px">
                <options>
```

```

        <option label="True" value="Debug" default="true"/>
        <option label="False" value="Normal"/>
    </options>
</param>
</params>
</plugin>
"""

import Domoticz
import os
import datetime
from gpiozero import LED, Button

# Set the plugin version
PLUGINVERSION = "v0.5.0"
PLUGINSHORTDESCRIPTON = "GPIOZero Push-Button LED OnOff"

# Units for the devices used
UNITALERT = 1 # alert device

# Alert Levels
ALERTON = 4      # red
ALERTOFF = 1     # green

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2 (see below).
        self.ledpin = 0      # default = 18
        self.buttonpin = 0   # default = 23
        # Define the alert state as set by the alertlevels
        # This will be used to turn the led on (alert level ALERTON) or off (alert level ALERTOFF)
        self.alertlevel = ALERTOFF
        return

    def onStart(self):
        Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
        Domoticz.Debug("onStart called")
        Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

        #Check debug mode
        if Parameters["Mode6"] == "Debug":
            self.debug = True
            Domoticz.Debugging(1)
            dump_config_to_log()

        #Setting up GPIO, i.e. get the pin numbers, set mode BCM, warnings off, set the pin mode and events
        self.ledpin = int(Parameters["Mode1"])
        self.buttonpin = int(Parameters["Mode2"])
        # Domoticz.Debug("Setting up GPIO:LED=" + Parameters["Mode1"] + ", Button=" + Parameters["Mode2"])
        Domoticz.Debug("Setting up GPIO:LED=" + str(self.ledpin) + ", Button=" + str(self.buttonpin))
        try:
            # GPIOZero define led (output) & push-button (input)
            self.led = LED(self.ledpin)
            # define the button as pull down else pressed & released are reversed
            self.button = Button(self.buttonpin, pull_up = False, hold_time = 1)
            # assign the callback functions without }()
            # ensure onButtonCallback is in the same class and has self, channel as signature
            self.button.when_pressed = self.onButtonCallback
            self.led.off()
            self.alertlevel = ALERTOFF
            Domoticz.Debug("Setting up GPIO: Done")
        except:
            Domoticz.Error("Error setting up GPIO pins.")

        #Create device if needed
        # Domoticz.Debug("Devices:" + str(len(Devices)) )
        if (len(Devices) == 0):
            Domoticz.Debug("Creating device(s)")
            Domoticz.Device(Name="GPIO Alert", Unit=UNITALERT, TypeName="Alert", Used=1).Create()
            # nValue = Level(0=gray, 1=green, 2=yellow, 3=orange, 4=red), sValue = TEXT (to display)
            Devices[UNITALERT].Update(nValue=ALERTOFF, sValue="LED OFF")
            Domoticz.Debug("Device created: "+Devices[UNITALERT].Name)

```

```

def onStop(self):
    Domoticz.Debug("onStop called: Shut down the device and release all associated resources
(IMPORTANT)")
    self.button.close()
    self.led.close()

def onConnect(self, Connection, Status, Description):
    Domoticz.Debug("onConnect called")

def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called")

def onCommand(self, Unit, Command, Level, Hue):
    Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Parameter '" + str(Command) + "' ,
Level: " + str(Level))

def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
    Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," +
str(Priority) + "," + Sound + "," + ImageFile)

def onDeviceModified(self, Unit):
    Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))

def onDisconnect(self, Connection):
    Domoticz.Debug("onDisconnect called")

def onHeartbeat(self):
    Domoticz.Debug("onHeartbeat called")

"""
Handle button changes.
Button pressed (button.is_pressed) = if LED on, then LED off v.v.
Button released is not used.
This function must be in the same class as onStart because it uses the self-object to access the
objects.
"""
def onButtonCallback(self):
    Domoticz.Debug("onButtonCallback called")
    if self.button.is_pressed:
        # Check the alert level and change the led state to on or off
        # The alert level is off > set to on and turn LED ON
        if self.alertlevel == ALERTOFF:
            self.led.on()
            self.alertlevel = ALERTON
            ledstate = "ON"
        elif self.alertlevel == ALERTON:
            # The alert level is on > set to off and turn LED OFF
            self.led.off()
            self.alertlevel = ALERTOFF
            ledstate = "OFF"
        else:
            ledstate = "UNKNOWN"
        Domoticz.Debug("Button was pressed. LED changed: {}, Alert Level: {}".format(ledstate,
self.alertlevel))
        isnow = datetime.datetime.now().strftime("%Y%m%d %X")
        msg = "LED state {} (Alert level {}) @ {}".format(ledstate, self.alertlevel, isnow)
        Devices[UNITALERT].Update(nValue=self.alertlevel, sValue=msg)
        Domoticz.Log(msg)
    Domoticz.Debug("onButtonCallback done")

global _plugin
_plugin = BasePlugin()

def onStart():
    global _plugin
    _plugin.onStart()

def onStop():
    global _plugin
    _plugin.onStop()

def onConnect(Connection, Status, Description):
    global _plugin
    _plugin.onConnect(Connection, Status, Description)

```

```

def onMessage(Connection, Data):
    global _plugin
    _plugin.onMessage(Connection, Data)

def onCommand(Unit, Command, Level, Hue):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Hue)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Generic helper functions
"""
Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:
        if Parameters[x] != "":
            Domoticz.Log( "'''" + x + ":" + str(Parameters[x]) + "''''")
    Domoticz.Debug("Device count: " + str(len(Devices)))
    for x in Devices:
        Domoticz.Log("Device: " + str(x) + " - " + str(Devices[x]))
        Domoticz.Log("Device ID: '" + str(Devices[x].ID) + "'")
        Domoticz.Log("Device Name: '" + Devices[x].Name + "'")
        Domoticz.Log("Device nValue: " + str(Devices[x].nValue))
        Domoticz.Log("Device sValue: '" + Devices[x].sValue + "'")
        Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
    return

```

## Domoticz Log

```

2020-06-11 11:30:52.310 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:52.310 (RPiGPIO) Button was pressed. LED changed: OFF, Alert Level: 1
2020-06-11 11:30:52.310 (RPiGPIO - GPIO Alert) Updating device from 4:'LED state ON (Alert level 4) @ 20200611 11:19:04.' to have values 1:'LED state OFF (Alert level 1) @ 20200611 11:30:52.'.
2020-06-11 11:30:52.327 (RPiGPIO) LED state OFF (Alert level 1) @ 20200611 11:30:52.
2020-06-11 11:30:52.327 (RPiGPIO) onButtonCallback done
2020-06-11 11:30:52.459 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:52.460 (RPiGPIO) onButtonCallback done
2020-06-11 11:30:53.832 (RPiGPIO) onButtonCallback called
2020-06-11 11:30:53.832 (RPiGPIO) Button was pressed. LED changed: ON, Alert Level: 4
2020-06-11 11:30:53.832 (RPiGPIO - GPIO Alert) Updating device from 1:'LED state OFF (Alert level 1) @ 20200611 11:30:52.' to have values 4:'LED state ON (Alert level 4) @ 20200611 11:30:53.'.
2020-06-11 11:30:53.849 (RPiGPIO) LED state ON (Alert level 4) @ 20200611 11:30:53.
2020-06-11 11:30:53.850 (RPiGPIO) onButtonCallback done

```

## dzVents Script

Example of a dzVents script acting if the alert level of the Alert Level device changes.

Source: [gpio\\_btntrigger.dzvents](#)

```

-- Handle changes of the alert level device
local IDX_ALERT = 132
local ALERTON = 4
local ALERTOFF = 1

return {
    on = {

```

```

devices = {
    IDX_ALERT
}
},
execute = function(domoticz, device)
    domoticz.log('Device ' .. device.name .. ' was changed', domoticz.LOG_INFO)
    domoticz.log(string.format('Alert Level: %d, Text: %s', domoticz.devices(IDX_ALERT).color,
domoticz.devices(IDX_ALERT).text))
    -- Action depending Level
    level = domoticz.devices(IDX_ALERT).color
    if (level == ALERTON) then
        domoticz.log('Alert is ON, please take action!')
    end
    if (level == ALERTOFF) then
        domoticz.log('Alert is OFF, No action required.')
    end
end
}
}

```

## Domoticz Log

The alert state was ALERTON. Pressing the button, changes the state to ALERTOFF. By pressing again, the state changes to ALERTON.

```

2020-06-11 11:43:40.801 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.801 (RPiGPIO) Button was pressed. LED changed: OFF, Alert Level: 1
2020-06-11 11:43:40.801 (RPiGPIO - GPIO Alert) Updating device from 4:'LED state ON (Alert level 4) @
20200611 11:39:31.' to have values 1:'LED state OFF (Alert level 1) @ 20200611 11:43:40.'.
2020-06-11 11:43:40.820 (RPiGPIO) LED state OFF (Alert level 1) @ 20200611 11:43:40.
2020-06-11 11:43:40.820 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:40.965 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:40.968 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED
state OFF (Alert level 1) @ 20200611 11:43:40."
2020-06-11 11:43:40.968 Status: dzVents: Info: ----- Start internal script: gpio_btntrigger: Device:
"RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-11 11:43:40.968 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-11 11:43:40.968 Status: dzVents: Info: Alert Level: 1, Text: LED state OFF (Alert level 1) @
20200611 11:43:40.
2020-06-11 11:43:40.968 Status: dzVents: Info: Alert is OFF, No action required.
2020-06-11 11:43:40.968 Status: dzVents: Info: ----- Finished gpio_btntrigger
2020-06-11 11:43:42.319 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:42.320 (RPiGPIO) Button was pressed. LED changed: ON, Alert Level: 4
2020-06-11 11:43:42.320 (RPiGPIO - GPIO Alert) Updating device from 1:'LED state OFF (Alert level 1) @
20200611 11:43:40.' to have values 4:'LED state ON (Alert level 4) @ 20200611 11:43:42.'.
2020-06-11 11:43:42.343 (RPiGPIO) LED state ON (Alert level 4) @ 20200611 11:43:42.
2020-06-11 11:43:42.343 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:42.492 (RPiGPIO) onButtonCallback called
2020-06-11 11:43:42.492 (RPiGPIO) onButtonCallback done
2020-06-11 11:43:42.487 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED
state ON (Alert level 4) @ 20200611 11:43:42."
2020-06-11 11:43:42.488 Status: dzVents: Info: ----- Start internal script: gpio_btntrigger: Device:
"RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-11 11:43:42.488 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-11 11:43:42.488 Status: dzVents: Info: Alert Level: 4, Text: LED state ON (Alert level 4) @
20200611 11:43:42.
2020-06-11 11:43:42.488 Status: dzVents: Info: Alert is ON, please act!
2020-06-11 11:43:42.488 Status: dzVents: Info: ----- Finished gpio_btntrigger

```

## Push-Button Device Plugin

### Purpose

This plugin builds upon the previous experiment and is extended with a Domoticz Switch to turn the LED On|Off.

### Python Script

Source: `gpiozero_btndevtrigger.plugin`

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# domoticz-home-automation-workbook - explore GPIOZero
# WARNING: ensure the plugin key is unique across all plugins.
# author: rwbl
"""

<plugin key="gpiozerobtndevtrigger" name="GPIOZero Push-Button Device Trigger" author="rwbl"
version="0.5.0" wikilink="" externallink="">
    <description>
        <h2>GPIOZero Push-Button Device Trigger</h2>
        Explore the Raspberry Pi GPIOZero library.<br/> This plugin
        <ul style="list-style-type:square">
            <li>explores how to use the GPIO input (including callback handling) and output.</li>
            <li>is a base template for exploring further plugin solutions.</li>
        </ul>
        <h3>Features</h3>
        <ul style="list-style-type:square">
            <li>Components: Push-Button (input) connected to GPIO23 (default), LED (output) connected
to GPIO18 (default).</li>
                <li>Creates the GPIO pins and set the GPIO numbering mode to BCM.</li>
                <li>By pressing the push-button, the LED state changes to ON or OFF (depending LED state)
and a Domoticz Alert Device gets updated.</li>
                    <li>By switching a Domoticz switch device, the LED state can be set to ON or OFF</li>
                    <li>When the Domoticz Alert Device gets updated, a dzVents script is executed.</li>
                </ul>
            <h3>Devices</h3>
            <ul style="list-style-type:square">
                <li>GPIO Status - Text device logging the LED state change.</li>
                <li>GPIO LED - Switch device to change LED state ON or OFF.</li>
            </ul>
            <h3>Configuration</h3>
            Caution: Make sure the GPIO pin numbers are not used by other applications. The pin numbers use
BCM format, i.e. GPIOnn (checkout for the board used).<br/>
            <ul style="list-style-type:square">
                <li>LED GPIO Pin Number (default: 18)</li>
                <li>Button GPIO Pin Number (default: 23)</li>
            </ul>
        </description>
        <params>
            <param field="Mode1" label="LED GPIO" width="75px" required="true" default="18"/>
            <param field="Mode2" label="Button GPIO" width="75px" required="true" default="23"/>
            <param field="Mode6" label="Debug" width="75px">
                <options>
                    <option label="True" value="Debug" default="true"/>
                    <option label="False" value="Normal"/>
                </options>
            </param>
        </params>
    </plugin>
"""

import Domoticz
import os
import datetime
from gpiozero import LED, Button

# Set the plugin version
PLUGINVERSION = "v0.5.0"
PLUGINSHORTDESCRIPTON = "GPIOZero Push-Button LED OnOff"

# Units for the devices used
UNITALERT = 1      # alert device
```

```

UNITSWITCHLED = 2    # switch onoff device

# Alert Levels
ALERTON = 4          # red
ALERTOFF = 1          # green

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2 (see below).
        self.ledpin = 0      # default = 18
        self.buttonpin = 0   # default = 23
        # Define the alert state as set by the alertlevels
        # This will be used to turn the led on (alert level ALERTON) or off (alert level ALERTOFF)
        self.alertlevel = ALERTOFF
        return

    def onStart(self):
        Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
        Domoticz.Debug("onStart called")
        Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

        #Check debug mode
        if Parameters["Mode6"] == "Debug":
            self.debug = True
            Domoticz.Debugging(1)
            dump_config_to_log()

        #Setting up GPIO, i.e. get the pin numbers, set mode BCM, warnings off, set the pin mode and
events
        self.ledpin = int(Parameters["Mode1"])
        self.buttonpin = int(Parameters["Mode2"])
        # Domoticz.Debug("Setting up GPIO:LED=" + Parameters["Mode1"] + ", Button=" +
Parameters["Mode2"])
        Domoticz.Debug("Setting up GPIO:LED=" + str(self.ledpin) + ", Button=" + str(self.buttonpin))
        try:
            # GPIOZero define led (output) & push-button (input)
            self.led = LED(self.ledpin)
            # define the button as pull down else pressed & released are reversed
            self.button = Button(self.buttonpin, pull_up = False, hold_time = 1)
            # assign the callback functions without ()!
            # ensure onButtonCallback is in the same class and has self, channel as signature
            self.button.when_pressed = self.onButtonCallback
            self.led.off()
            self.alertlevel = ALERTOFF
            Domoticz.Debug("Setting up GPIO: Done")
        except:
            Domoticz.Error("Error setting up GPIO pins.")

        #Create device if needed
        # Domoticz.Debug("Devices:" + str(len(Devices)) )
        if (len(Devices) == 0):
            Domoticz.Debug("Creating devices")
            Domoticz.Device(Name="GPIO Alert", Unit=UNITALERT, TypeName="Alert", Used=1).Create()
            # nValue = Level(0=gray, 1=green, 2=yellow, 3=orange, 4=red), sValue = TEXT (to display)
            Devices[UNITALERT].Update(nValue=ALERTOFF, sValue="LED OFF")
            Domoticz.Debug("Device created: "+Devices[UNITALERT].Name)

            Domoticz.Device(Name="GPIO Switch LED", Unit=UNITSWITCHLED, TypeName="Switch",
Used=1).Create()
            Domoticz.Debug("Device created: "+Devices[UNITSWITCHLED].Name)

    def onStop(self):
        Domoticz.Debug("onStop called: Shut down the device and release all associated resources
(IMPORTANT)")
        self.button.close()
        self.led.close()

    def onConnect(self, Connection, Status, Description):
        Domoticz.Debug("onConnect called")

    def onMessage(self, Connection, Data):
        Domoticz.Debug("onMessage called")

    def onCommand(self, Unit, Command, Level, Hue):

```

```

        Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Command '" + str(Command) + "'",
Level: " + str(Level))
        # onCommand called for Unit 2: Command 'On', Level: 0
        # Handle Switch LED
        if (Unit == UNITSWITCHLED):
            Domoticz.Debug("Switching Device to " + Command + "/" + str(Level))
            if (Command == "On"):
                self.led.on()
                Devices[UNITSWITCHLED].Update(1,"")
                Domoticz.Debug("LED " + Command)
                update_alert(Command, ALERTON)
            if (Command == "Off"):
                self.led.off()
                Devices[UNITSWITCHLED].Update(0,"")
                Domoticz.Debug("LED " + Command)
                update_alert(Command, ALERTOFF)

    def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
        Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," +
str(Priority) + "," + Sound + "," + ImageFile)

    def onDeviceModified(self, Unit):
        Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))

    def onDisconnect(self, Connection):
        Domoticz.Debug("onDisconnect called")

    def onHeartbeat(self):
        Domoticz.Debug("onHeartbeat called")

    """
Handle button changes.
Button pressed (button.is_pressed) = if LED on, then LED off v.v.
Button released is not used.
This function must be in the same class as onStart because it uses the self-object to access the
objects.
"""
    def onButtonCallback(self):
        Domoticz.Debug("onButtonCallback called")
        if self.button.is_pressed:
            # Check the alert level and change the led state to on or off
            # The alert level is off > set to on and turn LED ON
            if self.alertlevel == ALERTOFF:
                self.led.on()
                self.alertlevel = ALERTON
                ledstate = "ON"
            elif self.alertlevel == ALERTON:
                # The alert level is on > set to off and turn LED OFF
                self.led.off()
                self.alertlevel = ALERTOFF
                ledstate = "OFF"
            else:
                ledstate = "UNKNOWN"
            update_alert(ledstate, self.alertlevel)
        Domoticz.Debug("onButtonCallback done")

    global _plugin
    _plugin = BasePlugin()

    def onStart():
        global _plugin
        _plugin.onStart()

    def onStop():
        global _plugin
        _plugin.onStop()

    def onConnect(Connection, Status, Description):
        global _plugin
        _plugin.onConnect(Connection, Status, Description)

    def onMessage(Connection, Data):
        global _plugin
        _plugin.onMessage(Connection, Data)

```

```

def onCommand(Unit, Command, Level, Hue):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Hue)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Domoticz Device Updates
def update_alert(ledstate, alertlevel):
    Domoticz.Debug("Button was pressed. LED changed: {}, Alert Level: {}".format(ledstate, alertlevel))
    isnow = datetime.datetime.now().strftime("%Y%m%d %X")
    msg = "LED state {} (Alert level {}) @ {}".format(ledstate, alertlevel, isnow)
    Devices[UNITALERT].Update(nValue=alertlevel, sValue=msg)
    Domoticz.Log(msg)

# Generic helper functions
"""
Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:
        if Parameters[x] != "":
            Domoticz.Log( "'' + x + ':' + str(Parameters[x]) + ''")
    Domoticz.Debug("Device count: " + str(len(Domoticz)))
    for x in Devices:
        Domoticz.Log("Device:      " + str(x) + " - " + str(Devices[x]))
        Domoticz.Log("Device ID:    '" + str(Devices[x].ID) + "'")
        Domoticz.Log("Device Name:  '" + Devices[x].Name + "'")
        Domoticz.Log("Device nValue: " + str(Devices[x].nValue))
        Domoticz.Log("Device sValue: '" + Devices[x].sValue + "'")
        Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
    return

```

## Domoticz Log

The log shows clicking the Switch device to On, resulting in turning on the LED and updating the Alert device to level 4. Then the Switch device clicked again, and the state changed to Off, LED went Off and the Alert device changed to level 1.

```

2020-06-12 19:28:46.177 (RPiGPIO) Pushing 'onCommandCallback' on to queue
2020-06-12 19:28:46.189 (RPiGPIO) Processing 'onCommandCallback' message
2020-06-12 19:28:46.189 (RPiGPIO) Calling message handler 'onCommand'.
2020-06-12 19:28:46.189 (RPiGPIO) onCommand called for Unit 2: Command 'On', Level: 0
2020-06-12 19:28:46.190 (RPiGPIO) Switching Device to On/0
2020-06-12 19:28:46.190 (RPiGPIO - GPIO Switch LED) Updating device from 0:' to have values 1:'.
2020-06-12 19:28:46.207 (RPiGPIO) LED On
2020-06-12 19:28:46.207 (RPiGPIO) Button was pressed. LED changed: On, Alert Level: 4
2020-06-12 19:28:46.208 (RPiGPIO - GPIO Alert) Updating device from 1:'LED state OFF (Alert level 1) @ 20200612 18:55:15.' to have values 4:'LED state On (Alert level 4) @ 20200612 19:28:46.'.
2020-06-12 19:28:46.221 (RPiGPIO) LED state On (Alert level 4) @ 20200612 19:28:46.
2020-06-12 19:28:46.177 Status: User: Admin initiated a switch command (133/RPiGPIO - GPIO Switch LED/On)
2020-06-12 19:28:46.495 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED state On (Alert level 4) @ 20200612 19:28:46."
2020-06-12 19:28:46.496 Status: dzVents: Info: ----- Start internal script: gpio_btndown: Device: "RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-12 19:28:46.496 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-12 19:28:46.496 Status: dzVents: Info: Alert Level: 4, Text: LED state On (Alert level 4) @ 20200612 19:28:46.
2020-06-12 19:28:46.496 Status: dzVents: Info: Alert is ON, please act!

```

```
2020-06-12 19:28:46.496 Status: dzVents: Info: ----- Finished gpio_btntrigger
2020-06-12 19:28:48.663 (RPiGPIO) Pushing 'onCommandCallback' on to queue
2020-06-12 19:28:48.675 (RPiGPIO) Processing 'onCommandCallback' message
2020-06-12 19:28:48.676 (RPiGPIO) Calling message handler 'onCommand'.
2020-06-12 19:28:48.676 (RPiGPIO) onCommand called for Unit 2: Command 'Off', Level: 0
2020-06-12 19:28:48.676 (RPiGPIO) Switching Device to Off/0
2020-06-12 19:28:48.676 (RPiGPIO - GPIO Switch LED) Updating device from 1:'' to have values 0:''.
2020-06-12 19:28:48.692 (RPiGPIO) LED Off
2020-06-12 19:28:48.692 (RPiGPIO) Button was pressed. LED changed: Off, Alert Level: 1
2020-06-12 19:28:48.692 (RPiGPIO - GPIO Alert) Updating device from 4:'LED state On (Alert level 4) @ 20200612 19:28:46.' to have values 1:'LED state Off (Alert level 1) @ 20200612 19:28:48.'.
2020-06-12 19:28:48.706 (RPiGPIO) LED state Off (Alert level 1) @ 20200612 19:28:48.
2020-06-12 19:28:48.662 Status: User: Admin initiated a switch command (133/RPiGPIO - GPIO Switch LED/Off)
2020-06-12 19:28:48.972 Status: dzVents: Info: Handling events for: "RPiGPIO - GPIO Alert", value: "LED state Off (Alert level 1) @ 20200612 19:28:48."
2020-06-12 19:28:48.972 Status: dzVents: Info: ----- Start internal script: gpio_btntrigger: Device: "RPiGPIO - GPIO Alert (RPiGPIO)", Index: 132
2020-06-12 19:28:48.972 Status: dzVents: Info: Device RPiGPIO - GPIO Alert was changed
2020-06-12 19:28:48.972 Status: dzVents: Info: Alert Level: 1, Text: LED state Off (Alert level 1) @ 20200612 19:28:48.
2020-06-12 19:28:48.972 Status: dzVents: Info: Alert is OFF, No action required.
2020-06-12 19:28:48.972 Status: dzVents: Info: ----- Finished gpio_btntrigger
```

## RGB LED Plugin

### Purpose

This plugin enables to set the color of an RGB LED:

- via a Domoticz Selector Switch to Green, Yellow, Orange, Red
- via a Domoticz On/Off Switch to White (LED is ON) or Black (LED is OFF)

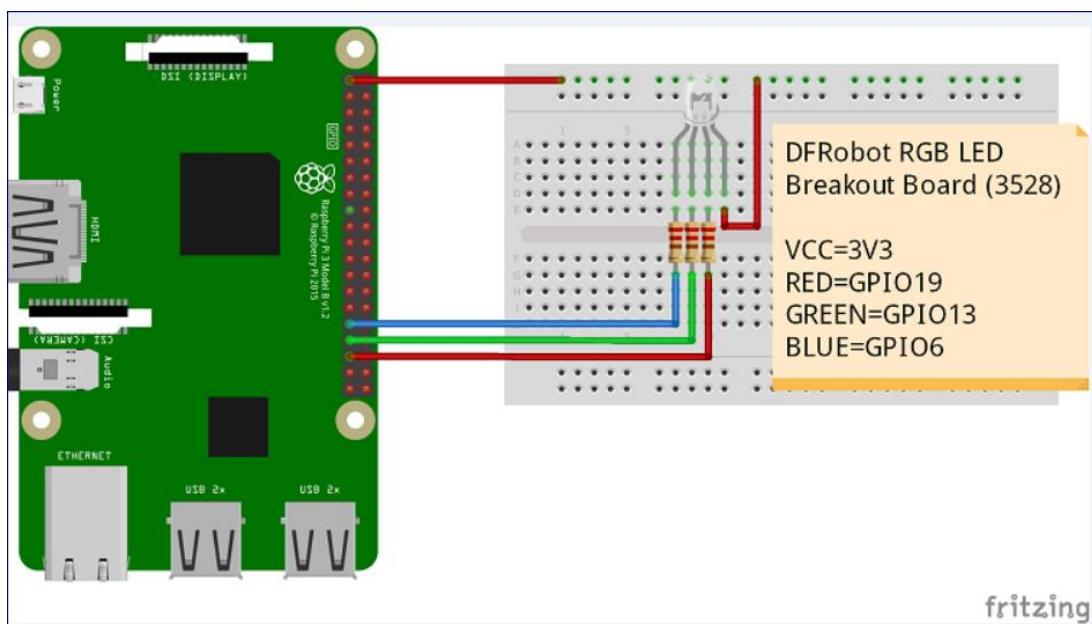
### Parts List

- 1x Raspberry Pi 4 V1.0 2GB
- 1x DFRobot RGB LED Breakout Board (3528)
- 1x Breadboard & Wires

### Wiring

RGB LED	Raspberry Pi
VCC	3V3
GND	GND
RED	Pin #35 GPIO19
GREEN	Pin #33 GPIO13
BLUE	Pin #31 GPIO6

### Circuit



## Plugin

- Installed in folder /home/pi/domoticz/plugins/gpiozerorgbled, file plugin.py.
- Listed as Domoticz hardware "GPIOZero RGB LED".
- Creates an alert device to indicate & log the LED state changes.

### Installing

Create folder:

```
/home/pi/domoticz/plugins/gpiozerorgbled
```

### Copy file

```
gpiozero_rgbled.plugin to /home/pi/domoticz/plugins/gpiozerorgbled/plugin.py
```

### Restart Domoticz:

```
sudo service domoticz.sh restart
```

Domoticz GUI > Setup > Hardware > Type: "GPIOZero RGB LED" > Name: RGBLED (as an example) > Check GPIO pins > Debug: True > Add.

Check the Domoticz log for errors.

The screenshot shows the Domoticz software interface for managing hardware components. At the top, there is a table listing a single entry:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
5	RGBLED	Yes	GPIOZero RGB LED			Disabled

Below the table, a message indicates: "Showing 1 to 1 of 1 entries (filtered from 3 total entries)". There are "Update" and "Delete" buttons at the top left of the configuration area.

The main configuration area for the "RGBLED" device includes the following fields:

- Enabled:**
- Name:** **RGBLED**
- Type:** **GPIOZero RGB LED**
- Data Timeout:** **Disabled** (with a note: "Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Do not enable this option for devices that do not receive data!")
- GPIOZero RGB LED** (description: "Set the color of a full color LED component (composed of red, green, and blue LEDs).")
- Features** (list):
  - Components: RGB LED (output) connected to three GPIO pins and 3V3 pin.
  - Select RGB LED color Green, Yellow, Orange, Red.
  - Set RGB LED light Off (Black) or On (White).
  - Alert device to indicate the color or state.
- Devices (Type,SubType,SwitchType,Name)** (list):
  - Light/Switch, Selector Switch, Signal.
  - Light/Switch, Switch, On/Off, Light.
  - General, Alert, Status.
- Configuration** (note: "Make sure the GPIO pin numbers are not used by other applications. The pin numbers use BCM format, i.e. GPIOnn (checkout for the board used). The VCC of the RGBLED is connected to 3V3 (do not use 5V)."):
  - RGBLED GPIO Pin Numbers (default): RED=GPIO19, GREEN=GPIO13, BLUE=GPIO6
- GPIO RED:** **19**
- GPIO GREEN:** **13**
- GPIO BLUE:** **6**
- Debug:** **True**

After adding, new devices are created by the plugin (below right).

The Domoticz log (left) logs device changes triggered by clicking on a switch.

```

2020-07-01 09:00:38.497 (RGBLED) Pushing 'onCommandCallback' on to queue
2020-07-01 09:00:39.538 (RGBLED) Processing 'onCommandCallback' message
2020-07-01 09:00:39.538 (RGBLED) Calling message handler 'onCommand'.
2020-07-01 09:00:39.538 (RGBLED) OnCommand called for Unit 1: Parameter 'Set Level', Level: 40, Hue:
2020-07-01 09:00:39.538 (RGBLED) SELECTORSWITCH:40
2020-07-01 09:00:39.538 (RGBLED - Signal) Updating device from 2:'10' to have values 2:'40'.
2020-07-01 09:00:39.538 (RGBLED) set_color called: R=0,G=1,B=1
2020-07-01 09:00:38.544 (RGBLED - Alert) Updating device from 1:'1' to have values 4:'4'.
2020-07-01 09:00:38.497 Status: User: Admin initiated a switch command (1/RGBLED - Signal/Set Level)
***
```

**Light/Switch Devices:**

RGBLED - Signal	RED
Last Seen: 2020-07-01 09:00:38	
	GREEN
	YELLOW
	ORANGE
	RED

RGBLED - Light	Off
Last Seen: 2020-07-01 08:55:58	

**Utility Sensors:**

RGBLED - Alert	4
Last Seen: 2020-07-01 09:00:38	

```

2020-07-01 09:02:29.081 (RGBLED) Pushing 'onCommandCallback' on to queue
2020-07-01 09:02:29.095 (RGBLED) Processing 'onCommandCallback' message
2020-07-01 09:02:29.095 (RGBLED) Calling message handler 'onCommand'.
2020-07-01 09:02:29.095 (RGBLED) OnCommand called for Unit 2: Parameter 'On', Level: 0, Hue:
2020-07-01 09:02:29.095 (RGBLED) ONOFFSWITCH:0,nValue=0,sValue=Off
2020-07-01 09:02:29.095 (RGBLED - Light) Updating device from 0:'off' to have values 1:'on'.
2020-07-01 09:02:29.102 (RGBLED) set_color called: R=0,G=0,B=0
2020-07-01 09:02:29.103 (RGBLED - Alert) Updating device from 4:'4' to have values 1:'1'.
2020-07-01 09:02:29.108 (RGBLED) RGBLED - Light - nValue=1,sValue=On
2020-07-01 09:02:29.081 Status: User: Admin initiated a switch command (2/RGBLED - Light/On)
2020-07-01 09:02:38.267 (RGBLED) Pushing 'onHeartbeatcallback' on to queue

```

**Light/Switch Devices:**

RGBLED - Signal	RED
Last Seen: 2020-07-01 09:00:38	
	GREEN
	YELLOW
	ORANGE
	RED

RGBLED - Light	On
Last Seen: 2020-07-01 09:02:29	

**Utility Sensors:**

RGBLED - Alert	1
Last Seen: 2020-07-01 09:02:29	

## Python Script

Source: gpiozero\_rgbled.plugin

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# domoticz-home-automation-workbook - explore GPIOZero
# WARNING: ensure the plugin key is unique across all plugins.
# Restart Domoticz on every change: sudo service domoticz.sh restart
#
# 20200630 rwbl
"""

<plugin key="gpiozero_rgbled" name="GPIOZero RGB LED" author="rwbl" version="1.0.0" wikilink=""
externallink=""
<description>
    <h2>GPIOZero RGB LED</h2>
    Set the color of a full color LED component (composed of red, green, and blue LEDs).<br/>
    <h3>Features</h3>
    <ul style="list-style-type:square">
        <li>Components: RGB LED (output) connected to three GPIO pins and 3V3 pin.</li>
        <li>Select RGB LED color Green,Yellow,Orange,Red.</li>
        <li>Set RGB LED light Off (Black) or On (White).</li>
        <li>Alert device to indicate the color or state.</li>
    </ul>
    <h3>Devices (Type,SubType,SwitchType,Name)</h3>
    <ul style="list-style-type:square">
        <li>Light/Switch, Selector Switch, Signal.</li>
        <li>Light/Switch, Switch, On/Off, Light.</li>
        <li>General, Alert, Status.</li>
    </ul>
    <h3>Configuration</h3>
    Make sure the GPIO pin numbers are not used by other applications. The pin numbers use BCM
format, i.e. GPIOnn (checkout for the board used).<br/>
    The VCC of the RGBLED is connected to 3V3 (do not use 5V).<br/>
    <ul style="list-style-type:square">
        <li>RGBLED GPIO Pin Numbers (default): RED=GPIO19, GREEN=GPIO13, BLUE=GPIO6</li>
    </ul>
</description>
<params>
    <param field="Mode1" label="GPIO RED" width="75px" required="true" default="19"/>
    <param field="Mode2" label="GPIO GREEN" width="75px" required="true" default="13"/>
    <param field="Mode3" label="GPIO BLUE" width="75px" required="true" default="6"/>
    <param field="Mode6" label="Debug" width="75px">
        <options>
            <option label="True" value="Debug" default="true"/>
            <option label="False" value="Normal"/>
        </options>
    </param>
</params>
</plugin>
"""

import Domoticz
import os
import datetime
from gpiozero import RGBLED

# Set the plugin version
PLUGINVERSION = "v1.0.0"
PLUGINSHORTDESCRIPTON = "GPIOZero RGB LED"

# Units for the devices used
UNITSELECTORSWITCH = 1
UNITONOFFSWITCH = 2
UNITALERT = 3

# Alert Levels
ALERTON = 4      # red
ALERTOFF = 1     # green

# Colors with several examples
COLORRED = [0,1,1]
COLORGREEN = [1,0,1]
COLORBLUE = [1,1,0]
COLORYELLOW = [0,0,1]
COLORORANGE = [0,0.5,1]
```

```

COLORCYAN = [1,0,0]
COLORBLACK = [1,1,1]
COLORWHITE = [0,0,0]
COLORON = [0,0,0]
COLOROFF = [1,1,1]

# Define the signal and the on or off colors
# The signal colors are also used for setting the alert device level: 0=gray, 1=green, 2=yellow,
# 3=orange, 4=red
SIGNALCOLORS = [COLOROFF,COLORGREEN,COLORYELLOW,COLORORANGE,COLORRED]
ONOFFCOLORS = [COLORON,COLOROFF]

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2 (see below).
        self.redledpin = 0
        self.greenledpin = 0
        self.blueledpin = 0
        self.rgbled = None
        self.colors = SIGNALCOLORS
        # Define the alert state as set by the alertlevels
        # This will be used to turn the led on (alert level ALERTON) or off (alert level ALERTOFF)
        self.alertlevel = ALERTOFF
        return

    def onStart(self):
        Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
        Domoticz.Debug("onStart called")
        Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

        #Check debug mode
        if Parameters["Mode6"] == "Debug":
            self.debug = True
            Domoticz.Debugging(1)
            dump_config_to_log()

        #Setting up GPIO
        self.redledpin = int(Parameters["Mode1"])
        self.greenledpin = int(Parameters["Mode2"])
        self.blueledpin = int(Parameters["Mode3"])
        Domoticz.Debug("Setting up GPIO Pins=RED=" + str(self.redledpin) + ", GREEN=" +
str(self.greenledpin) + ", BLUE=" + str(self.blueledpin))
        try:
            # RGBLED object
            Domoticz.Debug("RGBLED object creating...")
            self.rgbled = RGBLED(self.redledpin,self.greenledpin,self.blueledpin)
            Domoticz.Debug("RGBLED object created")
            Domoticz.Debug("RGBLED set color OFF...")
            self.set_color(COLOROFF)
            Domoticz.Debug("RGBLED color OFF")
            self.alertlevel = ALERTOFF
            Domoticz.Debug("Setting up GPIO: Done")
        except:
            Domoticz.Error("Error setting up GPIO pins.")

        # Create device if needed
        # Example: Domoticz.Device(Name="Status", Unit=1, Type=17, Switchtype=17).Create()
        # Domoticz.Debug("Devices:" + str(len(Devices)) )
        if (len(Devices) == 0):
            Domoticz.Debug("Creating device(s)")
            # TypeName=Selector Switch
            # Set the options for the selector switch with style Buttons set ("0")
            # Levels are 0,10,20,30,40
            Options = {"LevelActions": "|||",
                       "LevelNames": "OFF|GREEN|YELLOW|ORANGE|RED",
                       "LevelOffHidden": "true",
                       "SelectorStyle": "0"}
            Domoticz.Device(Name="Signal", Unit=UNITSELECTORSWITCH, TypeName="Selector Switch",
Options=Options, Used=1).Create()
            Devices[UNITSELECTORSWITCH].Update( nValue=0, sValue="0")
            Domoticz.Debug("Device created: "+Devices[UNITSELECTORSWITCH].Name)

            # TypeName=Switch, SubType=ONOFF (default)
            #

```

```

Domoticz.Device(Name="Light", Unit=UNITONOFFSWITCH, TypeName="Switch", Used=1).Create()
Devices[UNITONOFFSWITCH].Update( nValue=0, sValue="Off")
Domoticz.Debug("Device created: "+Devices[UNITONOFFSWITCH].Name)

# TypeName=Alert
Domoticz.Device(Name="Alert", Unit=UNITALERT, TypeName="Alert", Used=1).Create()
# nValue = Level(0=gray, 1=green, 2=yellow, 3=orange, 4=red), sValue = TEXT (to display)
Devices[UNITALERT].Update(nValue=ALERTOFF, sValue="RGBLED OFF")
Domoticz.Debug("Device created: "+Devices[UNITALERT].Name)

def onStop(self):
    Domoticz.Debug("onStop called: Shut down the device and release all associated resources (IMPORTANT)")
    self.rgbled.close()

def onConnect(self, Connection, Status, Description):
    Domoticz.Debug("onConnect called")

def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called:" + Data)

def onCommand(self, Unit, Command, Level, Hue):
    Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Parameter '" + str(Command) + "' , Level: " + str(Level) + ", Hue:" + str(Hue))
    alertlevel = 0
    newcolor = None
    # Select the unit
    # Set the color to the level selected
    if Unit == UNITSELECTORSWITCH:
        Domoticz.Debug("SELECTORSWITCH:" + str(Level))
        # Parameter: Turn LED Off: Parameter='Off', Level=0 OR 'On', 1
        # Parameter: Set Color: Parameter='Set Level', Level=10,20,30,40 ... and higher if defined
        # To set the RGB LED color use parameter level, i.e.
        0=COLOROFF,10=COLORGREEN,20=COLORYELLOW,30=COLORORANGE,40=COLORRED
        alertlevel = 0 if Level == 0 else int(Level / 10)
        newcolor = self.colors[alertlevel]
        # Update the selector switch selection
        Devices[Unit].Update( nValue=2, sValue=str(Level) )

    # Toggle the white light on or off
    if Unit == UNITONOFFSWITCH:
        Domoticz.Debug("ONOFFSWITCH:" + str(Level) + ",nValue=" + str(Devices[Unit].nValue) + ",sValue=" + Devices[Unit].sValue)
        # Toggle switch ON(1) or OFF(0) using the Unit nValue
        state = Devices[Unit].nValue
        if state == 0:
            alertlevel=1
            newcolor = COLORON
            Devices[Unit].Update(nValue=1,sValue="On")
        if state == 1:
            alertlevel=0
            newcolor = COLOROFF
            Devices[Unit].Update(nValue=0,sValue="Off")

    # Set the RGBLED color and update the switch device
    self.set_color(newcolor)
    # Update the alert with alertlevel
    Devices[UNITALERT].Update(nValue=alertlevel, sValue=str(alertlevel))
    # Debug
    Domoticz.Debug(Devices[Unit].Name + " - nValue=" + str(Devices[Unit].nValue) + ",sValue=" + Devices[Unit].sValue)

def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
    Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," + str(Priority) + "," + Sound + "," + ImageFile)

def onDeviceModified(self, Unit):
    Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))

def onDisconnect(self, Connection):
    Domoticz.Debug("onDisconnect called")

def onHeartbeat(self):
    Domoticz.Debug("onHeartbeat called")

```

```

"""
Set the color of the RGB LED
Value 0=HIGH (ON), 1=LOW (OFF)
"""

# Set the color using an array [R,G,B]
# Example: COLORRED = [0,1,1], set_color(COLORRED)
def set_color(self,c):
    Domoticz.Debug("set_color called: R=" + str(c[0]) + ",G=" + str(c[1]) + ",B=" + str(c[2]))
    self.rgbled.color = (c[0],c[1],c[2])

# Set the color using values for each color
# Example: set_color_rgb(0,1,1) to set color RED
def set_color_rgb(self,r,g,b):
    Domoticz.Debug("set_color_rgb called:" + str(r)+"-"+str(g)+"-"+str(b))
    self.rgbled.color = (r,g,b)

global _plugin
_plugin = BasePlugin()

def onStart():
    global _plugin
    _plugin.onStart()

def onStop():
    global _plugin
    _plugin.onStop()

def onConnect(Connection, Status, Description):
    global _plugin
    _plugin.onConnect(Connection, Status, Description)

def onMessage(Connection, Data):
    global _plugin
    _plugin.onMessage(Connection, Data)

def onCommand(Unit, Command, Level, Hue):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Hue)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Generic helper functions
"""

Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:
        if Parameters[x] != "":
            Domoticz.Log( "''"+x+"':'" + str(Parameters[x]) + "'''")
    Domoticz.Debug("Device count: " + str(len(Devices)))
    for x in Devices:
        Domoticz.Log("Device:      " + str(x) + " - " + str(Devices[x]))
        Domoticz.Log("Device ID:     '" + str(Devices[x].ID) + "'")
        Domoticz.Log("Device Name:   '" + Devices[x].Name + "'")
        Domoticz.Log("Device nValue:  " + str(Devices[x].nValue))
        Domoticz.Log("Device sValue:  '" + Devices[x].sValue + "'")
        Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
    return

```

## dzVents Script

Source: gpiozerorgbled\_setlevel.dzvents

```
-- gpiozerorgbled_setlevel
-- Test setting the level of a selector switch controlling a gpiozero rgb IDX_RGBLED
-- 20200701 rwbl

-- idx of the rgbled device
local IDX_RGBLED = 1

return {
  on = {
    timer = {
      'every minute'
    }
  },
  execute = function(domoticz, timer)
    domoticz.log('Timer event was triggered by ' .. timer.trigger, domoticz.LOG_INFO)
    math.randomseed( os.time() )
    local newlevel = math.random(4) * 10
    domoticz.devices(IDX_RGBLED).switchSelector(newlevel)
    domoticz.log(string.format('New level: %d, %s', newlevel,
domoticz.devices(IDX_RGBLED).levelNames[(newlevel / 10) + 1]))
    -- Note: the new color name from the lua table is an integer starting at 1. the newlevel is divided
    by 10 and 1 is added to get the range between 1-4.
  end
}
```

## RGB LED Plugin Interface

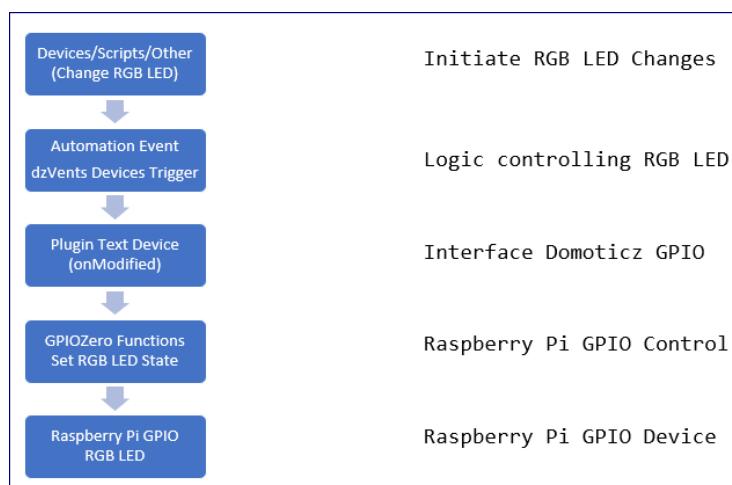
### Purpose

Set the color of the RGB LED via Virtual Device(s) controlling the Plugin Text Device which updates the RGB LED:

- Color Switch, RGB, Dimmer - set the RGB color & brightness
- Switch, On/Off - set the RGB state blinking

The wiring for this experiment, is the same as previous experiment.

The logic to set the state of the RGB LED is handled by Automation Event dzVents Device(s) trigger.



<TODO>

Instead of using a Text Device to set the RGB Color, use a Domoticz User Variable which is modified by the dzVents script and handled by the plugin (i.e. onModified for User Variable change).

Status: 20200720 - not supported by the Domoticz Python Plugin Framework.

### Plugin

- Installed in folder /home/pi/domoticz/plugins/gpiozerorgbledi, file plugin.py.
- Listed as Domoticz hardware "GPIOZero RGB LED Interface".
- Creates an alert device to indicate & log the LED state changes.

### Installing

Create folder:

```
/home/pi/domoticz/plugins/gpiozerorgbledi
```

### Copy file

```
gpiozero_rgbledi.plugin to /home/pi/domoticz/plugins/gpiozerorgbledi/plugin.py
```

### Restart Domoticz:

```
sudo service domoticz.sh restart
```

Domoticz GUI > Setup > Hardware > Type: "GPIOZero RGB LED Interface" > Name: RGBLED (as an example) > Check GPIO pins > Debug: True > Add.

Check the Domoticz log for errors.

Idx	Hardware	ID	Unit	Name	Type	SubType
10	RGB	00050003	3	RGB - Color	General	Text

Text device with data Red (255)-Green (0)-Blue (28)-Level (58)-Blink (0)

## Python Script

Source: `gpiozero_rgbledi.plugin`

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# domoticz-home-automation-workbook - explore GPIOZero
## WARNING: ensure the plugin key is unique across all plugins.
## Installation:
## Create folder <domoticz-path>/plugins/gpiozerorgbledi/
## Copy gpiozerorgbledi.plugin to plugin.py in folder gpiozerorgbledi
## Restart Domoticz (sudo service domoticz.sh restart) and add Hardware
## 20200715 makeRLinn
"""

<plugin key="gpiozerorgbledi" name="GPIOZero RGB LED Interface" author="rwbl" version="1.0.0"
wikilink="" externallink=""
    <description>
        <h2>GPIOZero RGB LED Interface</h2>
        Set the color Red,Green,Blue, brightness (Level) and blinking (On|Off) of a full color RGB LED
component.<br/>
        The values are set by modifying a virtual sensor text device holding the parameter value to set
the RGB Color via the GPIOZero Interfaca.<br/>
        The parameter value is a minus (-) separated string Red-Green-Blue-Level-Blink. Ranges: RGB=0-
255, L=0-100, Blink=0|1<br/>
        The text device value can be modified via dzVents or any other script / method, i.e. it acts as
the interface between Domoticz Devices & GPIOZero.<br/>
    <h3>Features</h3>
    <ul style="list-style-type:square">
        <li>Components: RGB LED (output) connected to three GPIO pins and 3V3 pin.</li>
        <li>Set RGB LED color with brightness and blink.</li>
    </ul>
    <h3>Device (Type, SubType, SwitchType, Name)</h3>
    <ul style="list-style-type:square">
        <li>General, Text, n/a, Color</li>
    </ul>
    <h3>Configuration</h3>
```

```

Make sure the GPIO pin numbers are not used by other applications. The pin numbers use BCM
format, i.e. GPIOnn (checkout for the board used).<br/>
The VCC of the RGB LED is connected to 3V3 (do not use 5V).<br/>
<ul style="list-style-type:square">
    <li>RGB LED GPIO Pin Numbers (default): RED=GPIO19, GREEN=GPIO13, BLUE=GPIO6</li>
</ul>
</description>
<params>
    <param field="Mode1" label="GPIO RED" width="75px" required="true" default="19"/>
    <param field="Mode2" label="GPIO GREEN" width="75px" required="true" default="13"/>
    <param field="Mode3" label="GPIO BLUE" width="75px" required="true" default="6"/>
    <param field="Mode6" label="Debug" width="75px">
        <options>
            <option label="True" value="Debug" default="true"/>
            <option label="False" value="Normal"/>
        </options>
    </param>
</params>
</plugin>
"""
import Domoticz
import os
import sys
import datetime
from gpiozero import RGBLED
import json

# Set the plugin version
PLUGINVERSION = "v1.0.0"
PLUGINSHORTDESCRIPTON = "GPIOZero RGB LED Interface"

# Units for the devices used
UNITRGBLCOLOR = 1

# Colors with several examples
COLORRED = [0,1,1]
COLORGREEN = [1,0,1]
COLORBLUE = [1,1,0]
COLORYELLOW = [0,0,1]
COLORORANGE = [0,0.5,1]
COLORCYAN = [1,0,0]
COLORBLACK = [1,1,1]
COLORWHITE = [0,0,0]
COLORON = [0,0,0]
COLOROFF = [1,1,1]

# Define the signal and the on or off colors
SIGNALCOLORS = [COLOROFF,COLORGREEN,COLORYELLOW,COLORORANGE,COLORRED]
ONOFFCOLORS = [COLORON,COLOROFF]

class BasePlugin:

    def __init__(self):
        # Define the GPIO Pins. Will be assigned from the parameters mode1 and mode2 (see below).
        self.redledpin = 0
        self.greenledpin = 0
        self.blueledpin = 0
        self.rgbled = None
        self.level = 0
        self.red = 0
        self.green = 0
        self.blue = 0
        self.blink = 0
        return

    def onStart(self):
        Domoticz.Debug(PLUGINSHORTDESCRIPTON + " " + PLUGINVERSION)
        Domoticz.Debug("onStart called")
        Domoticz.Debug("Debug Mode:" + Parameters["Mode6"])

        #Check debug mode
        if Parameters["Mode6"] == "Debug":
            self.debug = True
            Domoticz.Debugging(1)
            dump_config_to_log()

```

```

#Setting up GPIO
self.redledpin = int(Parameters["Mode1"])
self.greenledpin = int(Parameters["Mode2"])
self.blueledpin = int(Parameters["Mode3"])
Domoticz.Debug("Setting up GPIO Pins=RED=" + str(self.redledpin) + ", GREEN=" +
str(self.greenledpin) + ", BLUE=" + str(self.blueledpin))
try:
    # RGBLED object
    self.rgbled = RGBLED(self.redledpin,self.greenledpin,self.blueledpin)
    self.set_color(COLOROFF)
    Domoticz.Debug("Setting up GPIO: Done")
except:
    Domoticz.Error("Error setting up GPIO pins.")

# Create device if needed
# Example: Domoticz.Device(Name="Status", Unit=1, Type=17, Switchtype=17).Create()
# Domoticz.Debug("Devices:" + str(len(Devices)) )
if (len(Devices) == 0):
    Domoticz.Debug("Creating device(s)")
    # TypeName=Text = holds the color R-G-B 0-255, Brightness (Level) 0-100 Blink 0|1
    Domoticz.Device(Name="Color", Unit=UNITRGBLCOLOR, TypeName="Text", Used=1).Create()
    Devices[UNITRGBLCOLOR].Update( nValue=0, sValue="255-255-255-100-0")
    Domoticz.Debug("Device created: "+Devices[UNITRGBLCOLOR].Name)

def onStop(self):
    Domoticz.Debug("onStop called: Shut down the device and release all associated resources
(IMPORTANT)")
    self.rgbled.close()

def onConnect(self, Connection, Status, Description):
    Domoticz.Debug("onConnect called:"+str(Status)+","+Description)

def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called")
    if type(Data) == dict:
        Domoticz.Debug("Data dict:"+Data.decode("utf-8", "ignore"))

def onCommand(self, Unit, Command, Level, Color):
    Domoticz.Debug("onCommand called for Unit " + str(Unit) + ": Command '" + str(Command) + "'",
Level: " + str(Level) + ", Color:" + str(Color))

def onNotification(self, Name, Subject, Text, Status, Priority, Sound, ImageFile):
    Domoticz.Debug("Notification: " + Name + "," + Subject + "," + Text + "," + Status + "," +
str(Priority) + "," + Sound + "," + ImageFile)

def onDeviceModified(self, Unit):
    Domoticz.Debug("onDeviceModified called for Unit " + str(Unit))
    # Set the color to the R-G-B-L-B value (red-green-blue-level-blink)
    if Unit == UNITRGBLCOLOR:
        Domoticz.Debug("RGBLCOLOR:" + Devices[Unit].sValue)
        colors = Devices[Unit].sValue.split('-')
        if len(colors) == 5:
            self.red = int(colors[0])
            self.green = int(colors[1])
            self.blue = int(colors[2])
            self.level = int(colors[3])
            self.blink = int(colors[4])
            self.set_color_rgblb(self.red, self.green, self.blue, self.level, self.blink)
            #Domoticz.Debug(Devices[Unit].Name + ":ID:" + str(Devices[Unit].ID) + ",Color:" +
str(colors))
        if len(colors) < 5:
            Domoticz.Log("ERROR: Missing color parameter:"+str(colors))

def onDisconnect(self, Connection):
    Domoticz.Debug("onDisconnect called")

def onHeartbeat(self):
    Domoticz.Debug("onHeartbeat called")

"""
Set the color of the RGB LED
Value 0=HIGH (ON), 1=LOW (OFF)
"""
# Set the color using an array [R,G,B]

```

```

# Example: COLORRED = [0,1,1], set_color(COLORRED)
def set_color(self,c):
    Domoticz.Debug("set_color called:" + str(c))
    Domoticz.Debug("set_color called: R=" + str(c[0]) + ",G=" + str(c[1]) + ",B=" + str(c[2]))
    self.rgbled.color = (c[0],c[1],c[2])

# Set the color, brightness (level), blink (interval 1s) using values for each property
# Example: set_color_rgblb(255,0,0,100,1) to set RED max color blinking
def set_color_rgblb(self,red,green,blue,level,blink):
    Domoticz.Debug("set_color_rgblb:RGB 0-255=" + str(red)+"-"+str(green)+"-
"+str(blue)+":Level=" + str(level)+":Blink=" + str(blink))
    # adjust the level between 0 - 1 to set the color to c = c * 1
    level = level * 0.01
    # 1=maxcolor; 0=min color
    rm = mapFromTo(red * level, 0, 255, 1, 0)
    gm = mapFromTo(green * level,0, 255, 1, 0)
    bm = mapFromTo(blue * level,0, 255, 1, 0)
    self.rgbled.color = (rm,gm,bm)
    # blink: 0=OFF,1=ON
    if blink == 1:
        self.rgbled.blink(on_color=(rm,gm,bm),off_color=(1,1,1))
    Domoticz.Debug("set_color_rgblb:RGB 0-1=" + str(rm) + "," + str(gm) + "," + str(bm) +
",Level=" + str(level) + ",Blink=" + str(blink))

global _plugin
_plugin = BasePlugin()

def onStart():
    global _plugin
    _plugin.onStart()

def onStop():
    global _plugin
    _plugin.onStop()

def onConnect(Connection, Status, Description):
    global _plugin
    _plugin.onConnect(Connection, Status, Description)

def onMessage(Connection, Data):
    global _plugin
    _plugin.onMessage(Connection, Data)

def onCommand(Unit, Command, Level, Color):
    global _plugin
    _plugin.onCommand(Unit, Command, Level, Color)

def onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile):
    global _plugin
    _plugin.onNotification(Name, Subject, Text, Status, Priority, Sound, ImageFile)

def onDeviceModified(Unit):
    global _plugin
    _plugin.onDeviceModified(Unit)

def onDisconnect(Connection):
    global _plugin
    _plugin.onDisconnect(Connection)

def onHeartbeat():
    global _plugin
    _plugin.onHeartbeat()

# Generic helper functions
"""
Dump the config parameters of the plugin to the Domoticz log.
"""
def dump_config_to_log():
    for x in Parameters:
        if Parameters[x] != "":
            Domoticz.Log( "'''" + x + "'':'" + str(Parameters[x]) + "''''")
    Domoticz.Debug("Device count: " + str(len(Devices)))
    for x in Devices:
        Domoticz.Log("Device:           " + str(x) + " - " + str(Devices[x]))
        Domoticz.Log("Device ID:       '" + str(Devices[x].ID) + "'")

```

```

        Domoticz.Log("Device Name:    '" + Devices[x].Name + "'")
        Domoticz.Log("Device nValue:   " + str(Devices[x].nValue))
        Domoticz.Log("Device sValue:   '" + Devices[x].sValue + "'")
        Domoticz.Log("Device LastLevel: " + str(Devices[x].LastLevel))
    return

"""
mapping a range from to
newpos=mapFromTo(pos,0,300,0,1024)
"""

def mapFromTo(x,a,b,c,d):
    y=(x-a)/(b-a)*(d-c)+c
    return y

```

## dzVents Script

Source: gpiozerorgbled\_setcolor.dzvents

```

-- gpiozerorgbled_setcolor.dzvents
-- Listen to device changes to update the RGBL value of the text device RGB Color.
-- Devices (idx,hardware,name,type,subtype,switchtype):
-- 4,VirtualSensors,RGB - Color Picker,Color Switch,RGB,Dimmer
-- 11,VirtualSensors,RGB - Blink,Light/Switch,Switch,On/Off
-- 20200714 makeRLinn

-- idx of the rgbled device
local IDX_RGBCOLORPICKER = 4      -- RGB Color device with a color picker (virtual hardware)
local IDX_RGBBLINK = 11             -- Switch On/Off to let the LED blink (virtual hardware)
local IDX_RGBLCOLOR = 10            -- text device created by the rgb plugin and gets updated by the
RGBCOLORPICKER device

return {
  on = {
    devices = {IDX_RGBCOLORPICKER,IDX_RGBBLINK}
  },
  execute = function(domoticz, device)
    -- dump table data used to understand the properties of the tables from rawdata & getcolor
    local function dumptable(t)
      for i,v in pairs(t) do
        domoticz.log(string.format("%s=%s",i, tostring(v)))
      end
    end

    -- check which device is updated
    domoticz.log(string.format("Device changed: %s; sValue=", device.name, device.sValue),
domoticz.LOG_INFO)

    -- get blink state
    local blink = 0
    if domoticz.devices(IDX_RGBBLINK).sValue == "On" then blink = 1 end

    local value

    if device.idx == IDX_RGBCOLORPICKER then
      -- get the color as a lua table
      -- {[["cw"]]=0, [["green"]]=45, [["r"]]=255, [["blue"]]=26, [["t"]]=0, [["ww"]]=0, [["isWhite"]]=false,
      -- [["temperature"]]=0, [["value"]]=100.0, [["m"]]=3, [["hue"]]=4.9781659388646, [["b"]]=26,
      ["red"]]=255,
      -- [["brightness"]]=100.0, [["cold white"]]=0, [["mode"]]=3, [["g"]]=45,
      ["saturation"]]=89.803921568627, [["warm white"]]=0}
      local devicecolor = domoticz.devices(IDX_RGBCOLORPICKER).getColor()
      -- dumptable(devicecolor)

      -- get level: taken from the rawData key 1, i.e. 1=29 - this is the first entry
      -- the level is not included in the properties of device.getColor()
      -- TODO: checkout if the rawData for a RGB switch changes, i.e. the keys might change
      local level = tonumber(device.rawData[1])
      -- dumptable(device.rawData)

      value = string.format('%d-%d-%d-%d', devicecolor.r, devicecolor.g, devicecolor.b, level,
blink)
    end

    if device.idx == IDX_RGBBLINK then
      value = domoticz.devices(IDX_RGBLCOLOR).sValue
    end
  end
}

```

```
        function replace_char(pos, str, r)
            return str:sub(1, pos-1) .. r .. str:sub(pos+1)
        end
        -- replace last value = blink flag 0|1
        value = replace_char(string.len(value), value, tostring(blink))
        -- domoticz.log(string.format("Blink changed: %s", value), domoticz.LOG_INFO)
        end

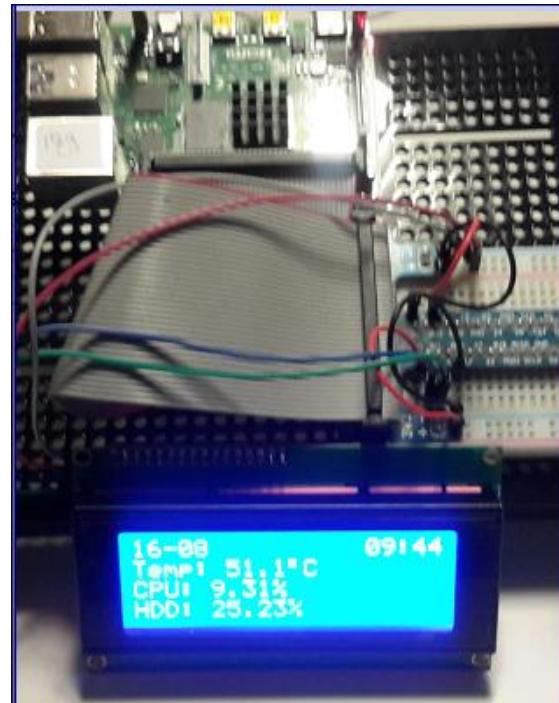
        -- update text device color with the new color string r-g-b-l
        -- the plugin triggers the function onmodified and updates the LED via GPIOZero
        domoticz.devices(IDX_RGBLCOLOR).updateText(value)
        domoticz.log(string.format("New color set:%s",value), domoticz.LOG_INFO)
    end
}
```

# I2C LCD Display

## Purpose

- To explore how to use the Raspberry Pi GPIO with an 20x4 I2C LCD Display (20 columns, 4 lines) connected
- To control the LCD via Domoticz devices & scripts

## Prototype examples Clock & Hardware Monitor



## Abbreviations

RPi = Raspberry Pi, LCD = 20x4 I2C LCD Display, I2C = Inter-Integrated Circuit, CLI = Command-Line Interface, JSON = JavaScript Object Notation.

## Prepare

### Hardware

- Raspberry Pi 4B 2GB
- 20x4 I2C LCD Display  
(AZDelivery HD44780 2004 LCD Display Bundle Blue 4x20 Char with I2C)
- Breadboard & various wires

### Software

- Raspberry Pi OS - Raspbian GNU/Linux 10 (buster)
- Python 3.7.3, GCC 8.3.0
- Domoticz Beta channel 2020.2 (i.e. build 12230)

## Wiring

LCD	RPi (Physical Pin, GPIO)
GND	GND (Pin 9)
VCC	5V (Pin 2)
SDA	SDA (Pin 3, GPIO 2, Serial Data)
SCL	SCL (Pin 5, GPIO 3, Serial Clock)
I2C Address: 0x27	

## I2C Setup

Check if the required Python I2C tools are installed:

```
sudo apt-get install python-smbus i2c-tools
```

Enable the I2C Interface in Raspberry Pi Preferences (using the Desktop) or use sudo raspi-config (using the Command-Line Interface CLI).

Check the I2C LCD Address = 7 bit (excluding the read / write bit):

```
sudo i2cdetect -y 1
```

Result:

```
pi@dodev:~ $ sudo i2cdetect -y 1
  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --- - - - - - - - - - - - -
10: --- - - - - - - - - - - - -
20: --- - - - - - - - - - - 27
30: - - - - - - - - - - - -
40: - - - - - - - - - - - -
50: - - - - - - - - - - - -
60: - - - - - - - - - - - -
70: - - - - - - - - - - - -
```

The LCD I2C device address 0x27 is used in the next Python scripts.

To detect the installed SMBus (System Management Bus):

```
sudo i2cdetect -l
i2c-1 i2c bcm2835 (i2c@7e804000) I2C adapter
```

In this example, one bus is installed, listed as i2c-1.

### Note

- RPi I2C Bus 0 is used by the GPU to control the green LED and some internal power supplies.
- RPi I2C Bus 1, GPIO physical pins 2 (SDA) & 3 (SCL) can handle multiple devices, but the RPi is configured as the master on the I2C bus, and therefore for all other devices are to be configured as slaves.
- Do not use GPIO physical pins 27 (ID\_SC) & 28 (ID\_SD) = reserved for HAT's.

## Concept

### Flow

Domoticz Trigger (Device or Script) > LCD Control Script (Python3) > LCD Library (Python3) > LCD Display Setting.

### Files

All files must be in the Domoticz Python Script folder, i.e. on a Raspberry Pi in folder: /home/pi/domoticz/scripts/python

- lcdi2clib.py - Python library to drive the LCD
- lcdi2ccontrol.py - Python script to control the LCD using the LCD library
- lcdi2ccustomchar.json - Definition of up-to 7 custom characters

Unzip **lcdi2c.zip** to the Domoticz Script folder.

### Examples

- lcdi2c\_clock.dzvents - dzVents Lua script to display date & time with a nice text
- lcdi2c\_hwmonitor.dzvents - dzVents Lua script to display date & time and Raspberry Pi selected hardware data CPU Temperature, CPU Usage, HDD Usage.

## Python LCD Library

The Python library **lcdi2clib.py** is required.

This library is based upon [lcdlib.py](#) = Many thanks for providing.

The commands to control the LCD can be obtained from the HD44780 datasheet (lookup internet).

These are used to write bytes to the SMBus (i2c-1 in this case).

## Python LCD Control Script

The Python script **lcdi2ccontrol.py** has been developed to control the LCD.

This script

- acts as the interface between the required library **lcdi2clib.py** and Domoticz scripts (like dzVents)
- enables to set the text and/or configuration of the LCD
- uses an JSON formatted string argument to control the LCD (LCD Control Argument)

### Run

To run the script, execute Python3 with 2 arguments:

```
python3 <path>lcdi2ccontrol.py 'JSON formatted string'
```

The path is used for executing the script from dzVents - see dzVents examples.

## Description JSON Formatted String Argument

The argument is a JSON structured string (enclosed in ' ') with key:value pairs.

```
'{"address":"0xNN",
"backlight":0|1,"cursor":0|1,"blinking":0|1,"clear":0|1,"debug":0|1,"lcdlines":[{"line":1,"position":1,
"clear":1,"text":"Text 1"}, {"line":2,"position":1,"clear":1,"text":"Text 2"}, {"line":3,"position":1,"clear":1,"text":"Text 3"}, {"line":4,"position":1,"clear":1,"text":"Text 4"}]}'
```

**IMPORTANT:** all keys to be defined in lowercase.

Key:Value	Description
address:String	Address of the LCD. Default: "0x27" (optional)
backlight:0 1	Set backlight off/on (optional)
cursor:0 1	Set cursor off/on (optional)
blinking:0 1	Set blink cursor off/on (optional)
clear:0 1	Clear display (optional)
debug:0 1	Log to the console off/on (optional)
lcdlines:[ {"line":1,"position":1,"clear":1,"text":"Text 1"}, {"line":2,"position":1,"clear":1,"text":"Text 2"}, {"line":3,"position":1,"clear":1,"text":"Text 3"}, {"line":4,"position":1,"clear":1,"text":"Text 4"}]	The key lcdlines is a JSON array with 1 to N entries holding the line properties. It is possible to define a single entry for a line or define multiple entries for a line at different positions. (Mandatory=
line:1-4	Line numbers 1 to 4 (optional, default 1) (internal the lcd uses lines 0-3 but for ease of use the lines are set between 1-4)
position:1-20	Columns 1 to 20 (optional, default 1)
clear:0 1 2	Clear display (optional, default 0): 0=do not clear, 1=clear line, 2=clear display
text:"TEXT"	Text to display at the line (mandatory)

## LCD Custom Characters

The LCD can display up-to 7 custom characters.

The custom characters are defined in an JSON array stored in the file lcdi2ccustomchar.json. This file must be in the same folder as the Python script or plugin.

### Example Content

```
[{"id":0,"name":"battery","char":"14,27,17,17,17,17,17,31"}, {"id":1,"name":"clock","char":"31,17,10,4,14,31,31,0"}, {"id":2,"name":"arrowdown","char":"0,4,4,4,31,14,4,0"}, {"id":3,"name":"arrowup","char":"0,4,14,31,4,4,4,0"}, {"id":4,"name":"nochange","char":"0,0,31,0,31,0,0,0"}, {"id":5,"name":"smiley","char":"0,0,10,0,0,17,14,0"}, {"id":6,"name":"speaker","char":"1,7,15,31,31,15,7,1"}]
```

The custom characters can be defined with the tool [LCD-Custom-Char-Maker](#) written by the author.

The char key defines the character 8 bits as integers.

If the char is defined as bin, convert to integer by for example using the previous mentioned tool.

In Lua & dzVents scripts, the character is defined as unicode \uNNNN with backslash encoded, i.e \uNNNN must be defined as \\uNNNN.

### Example Lua Code

Display the smiley custom character, which is custom character with id 5 [\\0005](#) as defined in the JSON file:

```
local line2 = ('{"line":2,"position":1,"clear":0,"text":"%s"}'):format("Have a Good Day \\u0005")
```

## LCD Character Table

The LCD has its own character table as defined in its datasheet.

The character is also displayed using Unicode (as for the custom characters): \uNNNN.  
Convert the Upper 4 & Lower 4 bits to HEX and Unicode.

### Example Degree Character

(from the datasheet) in dzVents.

Upper 4 bits = 1101, Lower 4 bits = 1111

Converted Bits (UPPERLOWER UUUULLL): 11011111 = DF , 0xDF , \u00DF, 239

```
local line2 = ('{"line":2,"position":1,"clear":0,"text":"%s"}'):format("Have a Good Day \\u00DF")
```

## Examples Running Scripts CLI

### Set Text Lines 1 & 2

```
cd domoticz/scripts/python
python3 lcdi2control.py '{"address":"0x27", "clear":1,
"lcdlines":[{"line":1,"position":1,"text":"Text"}, {"line":2,"position":1,"text":"Text2"}]}'
python3 lcdi2control.py '{"clear":0,
"lcdlines":[{"line":1,"position":1,"clear":1,"text":"Hello"}, {"line":4,"position":1,"text":"World!"}]}'

Result:
{"status": "OK", "title": "Lines written #2"}
```

### Set Text Lines 1,2,3,4

```
python3 lcdi2control.py '{"address":"0x27", "lcdlines":[{"line":1,"position":1,"clear":1,"text":"Text 1"}, {"line":2,"position":1,"clear":1,"text":"Text 2"}, {"line":3,"position":1,"clear":1,"text":"Text 3"}, {"line":4,"position":1,"clear":1,"text":"Text 4"}]}'
```

### Set Backlight Off Default Address 0x27

```
python3 /home/pi/domoticz/scripts/python/lcdi2control.py '{"backlight":0}'

Result:
{"status": "OK", "title": "Set Configuration."}
```

### Display Custom Character

Example displaying a custom char with index 0, which is defined as the battery (see file lcdi2ccustomchar.json).

The custom character is defined in Unicode \u0000 (max is \u0007).

```
python3 lcdi2control.py
'{"lcdlines":[{"line":1,"position":5,"clear":1,"text":"Hello"}, {"line":4,"position":10,"clear":1,"text":"World = \u0000"}]}'
```

### Debug ON

Write two lines with debug on.

```
python3 lcdi2control.py '{"debug":1,
"lcdlines":[{"line":1,"position":5,"text":"Hello"}, {"line":3,"position":10,"clear":1,"text":"World"}]}'

DEBUG::JSON:A=0x27,B=1,C=0,BL=0,CL=1,L=[{"line": 1, "position": 5, "text": "Hello"}, {"line": 3, "position": 10, "clear": 1, "text": "World"}]
DEBUG::Creating LCD Object:A=0x27,L=4,C=20,CL=1
DEBUG::Creating LCD Object = done
DEBUG::LCD clear = done
DEBUG::set_configuration: B=1,C=0,BL=0,CL=1
DEBUG::cursor: 0 = done
DEBUG::blinking: 0 = done
DEBUG::clear: 1 = done
DEBUG::Customchar: #characters defined: 7
DEBUG::Customchar: 0,battery,[14, 27, 17, 17, 17, 17, 17, 31]
DEBUG::Customchar: Index=0,Name=battery,Char=14,27,17,17,17,17,17,31
DEBUG::Customchar: 1,clock,[31, 17, 10, 4, 14, 31, 31, 0]
DEBUG::Customchar: Index=1,Name=clock,Char=31,17,10,4,14,31,31,0
DEBUG::Customchar: 2,arrowdown,[0, 4, 4, 4, 31, 14, 4, 0]
DEBUG::Customchar: Index=2,Name=arrowdown,Char=0,4,4,4,31,14,4,0
DEBUG::Customchar: 3,arrowup,[0, 4, 14, 31, 4, 4, 4, 0]
DEBUG::Customchar: Index=3,Name=arrowup,Char=0,4,14,31,4,4,4,0
DEBUG::Customchar: 4,nochange,[0, 0, 31, 0, 31, 0, 0, 0]
DEBUG::Customchar: Index=4,Name=nochange,Char=0,0,31,0,31,0,0,0
```

```
DEBUG::Customchar: 5,smiley,[0, 0, 10, 0, 0, 17, 14, 0]
DEBUG::Customchar: Index=5,Name=smiley,Char=0,0,10,0,0,17,14,0
DEBUG::Customchar: 6,speaker,[1, 7, 15, 31, 31, 15, 7, 1]
DEBUG::Customchar: Index=6,Name=speaker,Char=1,7,15,31,31,15,7,1
DEBUG::Customchar = done
DEBUG::backlight: 1 = done
DEBUG::set_configuration = done
DEBUG::set_configuration result: {'status': 'OK', 'title': 'Set Configuration.'}
DEBUG::write_lines: text: [{"line": 1, "position": 5, "text": "Hello"}, {"line": 3, "position": 10, "clear": 1, "text": "World"}]
DEBUG::write_lines: parsing json ...
DEBUG::write_lines: #lines: 2
DEBUG::Line:1
DEBUG::line: items (l,p,t,c) 1,5>Hello,0
DEBUG::write_lines: L=0,P=4,T=Hello
DEBUG::Line:3
DEBUG::line: items (l,p,t,c) 3,10,World,1
DEBUG::write_lines: L=2,P=9,T=World
{"status": "OK", "title": "Lines written #2"}
```

## Domoticz Triggers

The LCD control script can be invoked several ways within Domoticz.

### Important

Domoticz database text fields have a max length of 200 characters as defined by SQLite3 VARCHAR(200).

This means, that the LCD Control Argument can be defined with max 200 characters if used for example in a Text Device, User Variable type string, Device On/Off action etc..

Especially if using Unicode, the 200 characters limit can be reached easily.

### Text Device (Virtual Sensor)

The text device holds the LCD Control Argument for all lines.

To hide the text device use \$ (Dollar sign) as name prefix (i.e. \$lcldlines).

If more than 200 characters are required as the LCD Control Argument, then use a text device for each line (i.e. \$lcldline1 to \$lcldline4).

### Domoticz Device

Create Text Device (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
25, LCD Text, General, Text,, {"line":1,"position":1,"text":"17-08"}, {"line":1,"position":8,"text":"10:02"}, {"line":2,"position":3,"text":"Have a Good Day \u0005"}, {"line":3,"position":10,"text":"Enjoy"}, {"line":4,"position":17,"text":"rwbl"}

*Note*

The data contains an example of JSON key lcldlines.

### LCD Control Example

The content of the text device(s) can be set by a dzVents script which triggers another dzVents script listening to text device changes to set the LCD.

The event lcdi2c\_textcontrol\_trigger.dzvents creates every minute text for the text device “LCD Text” and updates the text device “LCD Text”.

(Trigger: Timer Every Minute)

The event lcdi2c\_textcontrol.dzvents listens to text device “LCD Text” changes and updates the LCD by executing the Python script lcdi2c\_control.py.

(Trigger: Device Change)

Source: lcdi2c\_textcontrol\_trigger.dzvents

```
-- lcdi2c_textcontrol_trigger.dzvents
-- Change the text of the text device to update the lcd.
-- The event lcdi2c_textcontrol handles device text changes, which can be triggered by events, plugins,
mqtt or other.

-- device idx
local IDX_LCDETEXT = 25 -- text device holding the lcd control parameter (max 200 char)

return {
    on = {timer = {'every minute',}},
    execute = function(domoticz, timer)
        local now = os.date("*t")
        local datenow = ("%02d-%02d"):format(now.day, now.month)
        local timenow = ("%02d:%02d"):format(now.hour, now.min)
```

```

local line1 = ('{"line":1,"position":1,"text":"%s"}'):format(datenow) .. ',' ..
({'{"line":1,"position":8,"text":"%s"}'):format(timenow)
local line2 = ('{"line":2,"position":3,"text":"%s"}'):format("Have a Good Day \u263a")
local line3 = ('{"line":3,"position":10,"text":"%s"}'):format("Enjoy")
local line4 = ('{"line":4,"position":17,"text":"%s"}'):format("rwbl")
local lcdlines = string.format('%s,%s,%s,%s', line1, line2, line3, line4)
domoticz.devices(IDX_LCDETEXT).updateText(lcdlines)
end
}

```

### Source: lcdi2c\_textcontrol.dzvents

```

-- lcdi2c_textcontrol.dzvents
-- Handle text device change trigger to update the lcd display
-- Library Lua JSON
local JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux
-- Python command to be executed - will become additional JSON argument for the lines
local CMD = "python3 /home/pi/domoticz/scripts/python/lcdi2ccontrol.py"
-- Default LCD I2C address for LCD I2C 20x4 displays
local ADDRESS = "0x27"

-- device idx
local IDX_LCDETEXT = 25 -- text device holding the lcd control parameter (max 200 char)

-- Helper Functions
local function isempty(s)
    return s == nil or s == ''
end

-- Run external command: python in this case
-- Returns
-- Output of the command
-- Return code (from returnTable[3])
-- Return code selection (interpretation error numbers system dependent):
-- 0 = No error
-- 2 = No such file or directory
-- 13 = Permission denied
local function osExecute(domoticz, cmd)
    local fileHandle = assert(io.popen(cmd, 'r')) -- Read-only read
    local commandOutput = assert(fileHandle:read('*a')) -- Read all content *all
    local returnTable = {fileHandle:close()}
    -- returnTable: OK = {true,exit,0}; Error = {exit,2}
    local returnCode = returnTable[3]
    if isempty(commandOutput) and returnCode > 0 then
        if returnCode == 2 then commandOutput = "No such file or directory" end
        if returnCode == 13 then commandOutput = "Permission denied" end
    end;
    return commandOutput, returnCode
end

-- Run the external python scripts and handle return code
local function runscript(domoticz, args)
    if not isempty(args) then
        domoticz.log(args, domoticz.LOG_INFO)
        -- build the python command string - the args must be in ''
        local pycmd = string.format("%s '%s'", CMD, args)
        -- run the python script
        local output,rt = osExecute(domoticz, pycmd)
        -- domoticz.log(string.format('%d:%s',rt,output), domoticz.LOG_INFO)
        -- handle the result of the python command: 0 = OK, >0 = ERROR
        if rt == 0 then
            -- decode the result string, which is json format, i.e.
            -- {"status": "OK", "title": "Lines written #N"}
            -- {"status": "ERROR", "title": "Error message"}
            local result = JSON:decode(output);
            if result.status == "OK" then
                domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_INFO)
            else
                domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_ERROR)
            end
        -- handle the result of the python command
        else

```

```
domoticz.log(string.format('Python command: Code:%d, Output:%s',rt,output),
domoticz.LOG_ERROR)
    end
end
end

return {
on = {devices = {IDX_LCDTEXT}},
execute = function(domoticz, device)
    if device.idx == IDX_LCDTEXT then
        local lcdlines = ('%s'):format(device.sValue)
        local args = ('>{"address":"%s", "clear":1, "lcdlines":[%s]}'):format(ADDRESS, lcdlines)
        runscript(domoticz, args)
    end
end
}
```

## Switch LCD Backlight (Virtual Sensor)

The switch, from type On/Off, uses device actions to explicitly set the LCD backlight by invoking the script.

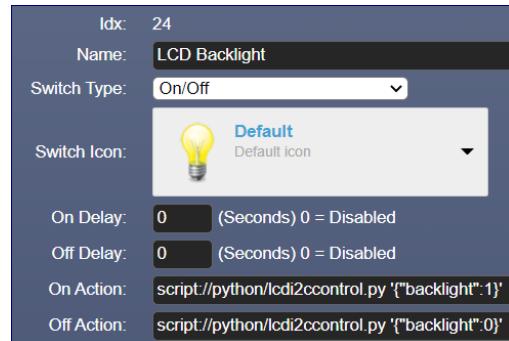
### Domoticz Device

Create Switch Device (Hardware Dummy Virtual Sensor):

<b>Idx, Hardware, Name, Type, SubType, SwitchType, Data</b>
---

24, VirtualDevices, LCD Backlight, Light/Switch, Switch, On/Off, Off
--

Define Switch Device On | Off Actions:



### Python Script

The Python script must be in the RPi folder:

/home/pi/domoticz/scripts/python
----------------------------------

Example Python Script to turn Backlight On/Off:

script://python/lcdi2ccontrol.py '{"backlight":1}' script://python/lcdi2ccontrol.py '{"backlight":0}'
--

### Notes

Running Python scripts on a Raspberry Pi, ensure to set the line endings to LF (UNIX).

Especially if developed on a Windows device with CRLF.

Check the Domoticz log after running the script.

### Domoticz Log

Script OK: The LCD Backlight is switched Off:

2020-08-17 09:03:04.592 (VirtualSensors) Light/Switch (LCD Backlight) 2020-08-17 09:03:04.587 Status: User: Admin initiated a switch command (24/LCD Backlight/Off) 2020-08-17 09:03:04.796 Status: Executing script: /home/pi/domoticz/scripts/python/lcdi2ccontrol.py
---

Script Error: The Line Endings are not correct

2020-08-16 18:47:48.129 Error: Error executing script command (/home/pi/domoticz/scripts/python/lcdi2ccontrol.py). returned: 32512
---

Action: Check the line endings in the script lcdi2ccontrol.py

## Hardware Monitor

Display selected Raspberry Pi 4 Hardware Data on the LCD (see picture beginning of this chapter).

The automation event is triggered by a timer every minute and executes the Python script: <domoticz folder>/scripts/python/lcdi2ccontrol.py to update the LCD.

### Python Script

The Python script must be in the RPi folder:

```
/home/pi/domoticz/scripts/python
```

```
-- File: lcdi2c_hwmonitor.dzvents
-- Configuration:
-- Set the path to the domoticz scripts folders for locals JSON, CMD
-- 20200803 by rwBL

-- Library Lua JSON
local JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux

-- Python command to be executed - will become additional JSON argument for the lines
local CMD = "python3 /home/pi/domoticz/scripts/python/lcdi2ccontrol.py"

-- Default LCD I2C address for LCD I2C 20x4 displays
local ADDRESS = "0x27"

-- Idx of the devices. The devices must be active! Not all used for the LCD
local IDX_RPi_CPU_Usage = 13      -- 13,RPi Hardware,RPi CPU Usage,General,Percentage,0.68%
local IDX_RPi_Temperature = 14    -- 14,RPi Hardware,RPi Temperature,Temp,LaCrosse TX3,45.3 C
local IDX_RPi_Memory_Usage = 15    -- 15,RPi Hardware,RPi Memory Usage,General,Percentage,11.11%
local IDX_RPi_Process_Usage = 16   -- 16,RPi Hardware,RPi Process Usage,General,Custom Sensor,45.03 MB
local IDX_RPi_HDD_Boot = 17        -- 17,RPi Hardware,RPi HDD /boot,General,Percentage,21.1%
local IDX_RPi_HDD = 18            -- 18,RPi Hardware,RPi HDD /,General,Percentage,24.62%

-- Helper Functions
function tablelength(T)
    local count = 0
    for _ in pairs(T) do count = count + 1 end
    return count
end

local function isempty(s)
    return s == nil or s == ''
end

-- Run external command: python in this case
-- Returns
-- Output of the command
-- Return code (from returnTable[3])
-- Return code selection (interpretation error numbers system dependent):
-- 0 = No error
-- 2 = No such file or directory
-- 13 = Permission denied
local function osExecute(domoticz, cmd)
    local fileHandle = assert(io.popen(cmd, 'r')) -- Read-only read
    local commandOutput = assert(fileHandle:read('*a')) -- Read all content *all
    local returnTable = {fileHandle:close()}
    -- returnTable: OK = {true,exit,0}; Error = {exit,2}
    local returnCode = returnTable[3]
    if isempty(commandOutput) and returnCode > 0 then
        if returnCode == 2 then commandOutput = "No such file or directory" end
        if returnCode == 13 then commandOutput = "Permission denied" end
    end;
    return commandOutput, returnCode
end

local function runscript(domoticz, args)
    -- run the external python scripts and handle return code
    if not isempty(args) then
```

```

domoticz.log(args, domoticz.LOG_INFO)
-- build the python command string - the args must be in ''
local pycmd = string.format("%s '%s'", CMD, args)
-- run the python script
local output,rt = osExecute(domoticz, pycmd)
-- domoticz.log(string.format('%d:%s',rt,output), domoticz.LOG_INFO)
-- handle the result of the python command: 0 = OK, >0 = ERROR
if rt == 0 then
    -- decode the result string, which is json format, i.e.
    -- {"status": "OK", "title": "Lines written #N"}
    -- {"status": "ERROR", "title": "Error message"}
    local result = JSON:decode(output);
    if result.status == "OK" then
        domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title), domoticz.LOG_INFO)
    else
        domoticz.log(string.format('Status:%s,Title:%s',result.status,result.title), domoticz.LOG_ERROR)
    end
    -- handle the result of the python command
else
    domoticz.log(string.format('Python command: Code:%d, Output:%s',rt,output), domoticz.LOG_ERROR)
end
end

return {
on = {
    timer = {'every minute',},
},
execute = function(domoticz, item)
    -- Update lcd
    -- Two options: either via timer every minute or on every device change
    -- Here the timer is used because hardware data might change rather frequent
    local args = ""
    if item.isTimer then
        local lines = {}      -- array holding the line entries. can be more then 4
        local linecnt = 1     -- counter for the line entries of the lines array
        local now = os.date("*t")
        -- define date & time now to display at line 1 [date    time]
        local datenow = ("%02d-%02d"):format(now.day, now.month)
        local timenow = ("%02d:%02d"):format(now.hour, now.min)
        lines[linecnt] = ('{"line":1,"position":1,"text":"%s"}'):format(datenow)
        linecnt = linecnt + 1
        lines[linecnt] = string.format('{"line":1,"position":%d,"text":"%s"}', (1 + (20 -
string.len(timenow)) ), timenow)
        linecnt = linecnt + 1
        lines[linecnt] = ('{"line":2,"position":1,"text":"Temp: %s\u00DFC"}'):format(
domoticz.devices(IDX_RPi_Temperature).sValue)
        linecnt = linecnt + 1
        lines[linecnt] = ('{"line":3,"position":1,"text":"CPU: %s%"}'):format(
domoticz.devices(IDX_RPi_CPU_Usage).sValue)
        linecnt = linecnt + 1
        lines[linecnt] = ('{"line":4,"position":1,"text":"HDD: %s%"}'):format(
domoticz.devices(IDX_RPi_HDD).sValue)
        -- Built the lines string for the script argument
        local lcdlines = table.concat(lines, ", ")
        -- Update LCD. Clear the display
        args = ('{"address":"%s", "clear":1, "lcdlines":[%s]}'):format(ADDRESS, lcdlines)
        runscript(domoticz, args)
    end
end
}

```

## Node-RED

A few Node-RED flows to illustrate using Node-RED as GPIO handler.  
The Node-RED nodes from the palette “node-red-node-pi-gpio” (v1.1.1) are used.  
The communication between Node-RED and Domoticz is via MQTT, topic “domoticz/in”.

### LED On|Off Push-Button

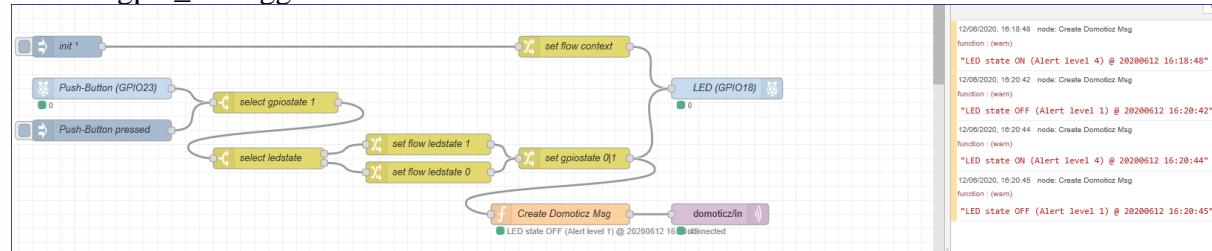
A sample Node-RED flow to handle push-button pressed to turn the LED on and set the Alert level.

The logic is handled by switch and changes nodes. The function node is creating the MQTT message for Domoticz.

The “node rpi-gpio in” listens to changes of GPIO23 (push-button connected). If the state is 1, means the push-button is pressed, the LED is set to On or Off depending state. The led state is captured by a flow.context “ledstate”.

The LED state is set by the node “rpi-gpio out”.

Source: [gpio\\_btntrigger.flow](#)



# Kodi Music Player

## Purpose

- To explore the music player functionality of the Kodi Media Player.
- To explore Kodi configuration, communicating with Kodi using JSON-RPC API requests, control & monitor the media player from various systems (Remote Browser, Domoticz and other).
- To setup a Kodi music player running on a Raspberry Pi with LibreELEC and Kodi Media Player, connected to the home stereo system and integrated in the local network.
- To use an old RPi 2 B v.1.1 (2014) as a starter - re-activated after its retirement. If the solution is running fine, the old RPi 2 will be replaced by a newer RPi 4.
- To prepare for a Domoticz Music Player Solution based on virtual sensors controller by a single dzVents Lua Automation Script (see Domoticz-Home-Automation-Workbook function Music Player)

[Kodi](#) - free and open-source media player software application developed by the XBMC Foundation.

[LibreELEC](#) - lightweight operating system designed to run Kodi.

## Parts List

- RPi 2B V1.1 (2014)
- SDHC card 32GB
- USB stick with media files
- USB WiFi Adapter 802.11m

## Installation

- Download image Raspberry Pi NOOBS: NOOBS\_v3\_3\_1.zip.
- Flashed on a 32GB SDHC Card using Etcher 1.5.53 under Microsoft Windows [Version 10.0.18363.815] device.
- Inserted the SDHC card into the RPi.
- RPi connect HDMI Monitor, USB Keyboard, USB Mouse, Ethernet wired cable.
- RPi plug in power.
- RPi installation process started - select LibreELEC\_RPi2 - Kodi Entertainment Center.
- After completion, reboot the RPi.
- (Brief information, for more details lookup various websites, i.e. Raspberry Pi, LibreELEC, Kodi)

### Notes

- Installed LibreELEC (official): 9.2.1 (RPi2.arm) with Kodi 18.6-Leia, Linux ARM 32-bit version 4.19.106 (267114), JSON-RPC API v8 (Krypton).
- Software versions are subject to change - checkout for newer versions.

## Kodi Settings

- RPi IP-Address Static: player-ip, Port:8080
- HTTP Remote access
- Audio: output device: PI:Analogue; Channels: 2.0;
- Input: Keyboard layout: German
- Internet access: Use proxy server: disabled;
- My add-ons>All>LibreELEC Configuration>Run:
  - System>System name: KEC; Keyboard: de
  - Network>Wireless>Active
  - Connections>Selected Wifi and set static IP address (set interface: wlan0,type: manual, address: player-ip)
  - Bluetooth disabled
  - Services: Samba and SSH enabled
- Services>Control:
  - Webserver: allow remote control via HTTP; Port:8080; Leave UN & PW empty (if not then issues with Node-RED or Domoticz HTTP RPC requests)
  - Application control: allow from this & other systems
- Addon "Script - Raspberry Pi Tools" Author:Team LibreELEC
  - Installed via GUI: Addons > Search > Script - Raspberry Pi Tools

## Kodi SSH Access

Use "ssh root@player-ip" to access Kodi.

Example from a Windows 10 Command Prompt Window:

```
Microsoft Windows [Version 10.0.18363.815]
(c) 2019 Microsoft Corporation. All rights reserved.

ssh root@player-ip
root@player-ip's password:
#####
#          LibreELEC      #
#          https://libreelec.tv    #
#####
LibreELEC (official): 9.2.1 (RPi2.arm)
```

## Kodi GUI Access

Open web browser and enter URL: http://player-ip:8080

## Kodi Hints

### *Get Kodi Version*

Run command from the CLI:

```
grep Kodi .kodi/temp/kodi.log | head -3
```

## Output

```
2020-05-11 08:42:00.196 T:1937043472 NOTICE: Starting Kodi (18.6 Git:newclock5_18.6-Leia). Platform:
Linux ARM 32-bit
2020-05-11 08:42:00.196 T:1937043472 NOTICE: Using Release Kodi x32 build (version for Raspberry Pi)
2020-05-11 08:42:00.196 T:1937043472 NOTICE: Kodi compiled 2020-03-05 by GCC 8.3.0 for Linux ARM 32-
bit version 4.19.106 (267114)
```

### *Set Raspberry Pi Volume*

Set the volume to max on the RPi and control the volume remote on the connected device or remote application (like browser).

#### Option via Browser

In the Desktop (attach monitor, keyboard, mouse to the RPi)

Open the Kodi player in the browser and set the volume via the keyboard + key.

This enables to remote control the volume when the Raspberry Pi is connected to an amp or a headset.

#### Option via Command Line

Open a terminal and use the command *amixer* with parameter GET or SET the volume.

```
GET volume: amixer -M sget PCM  
SET volume: amixer -q -M sset PCM 50%
```

Example set and get the volume to 70%

SET Volume

```
amixer -q -M sset PCM 70%
```

GET Volume

```
amixer -M sget PCM
```

Output

```
Simple mixer control 'PCM',0  
Capabilities: pvolume pvolume-joined pswitch pswitch-joined  
Playback channels: Mono  
Limits: Playback -10239 - 400  
Mono: Playback -529 [70%] [-5.29dB] [off]
```

## Kodi JSON-RPC API Communication

[JSON-RPC](#) is a HTTP- and/or raw TCP socket-based interface for communicating with Kodi. The JSON RPC API methods are used to get or set data in Kodi via Domoticz or other systems, like Node-RED.

To get started, went through the JSON-RPC API documentation v8 (Krypton) at the time of writing.

Examples using the JSON RPC API are:

- **Get** the current song playing and display in a Domoticz text device - Method Player.GetItem
- **Set** the playback to play or pause via a Domoticz Switch - Method Player.PlayPause

# Domoticz Automation Events

This simple music player is controlled by Domoticz devices triggering Domoticz Automation Events [dzVents](#).

The communication between Domoticz and the player is handled via Asynchronous HTTP GET and POST requests defined in the dzVents scripts.

The dzVents command `domoticz.openURL()` with parameter is used.

The dzVents scripts are triggered by Domoticz device changes or by timer.

## Read Data from Kodi

To read data from Kodi, use the **GET method**.

Example Kodi JSON-RPC API method: `Player.GetItem`

Below info has been taken from the the JSON-RPC API v8 documentation.

Have added additional clarification.

### Method

`Player.GetItem`

Retrieves the currently played item.

### Permissions

`ReadData`.

For `_ReadData_`, the HTTP GET request can be used. If permission is from type `_ControlPlayback_` then a HTTP POST request is required (the post data defined in the HTTP request body).

### Parameters

`Player.Id` `playerid`; `List.Fields.All` properties

The parameters are defined as JSON key:value pairs in the request key "params":

- Key `playerid` defined as "playerid":0
- Key `properties` defined as array with selected keys taken from "List.Fields.All" which has items defined as enumeration.
- For the example, selected "title","album","artist","duration".

### Returns

**Type**

`Object`

Properties: `List.Item.All` item

### Key

`item` contains the key from "List.Item.All" which has a range of properties.

Returned are the selected properties from the request:

"album":"ALBUM","artist":["ARTIST"],"duration":NNN,"id":NNN,"label":"LABEL","title":"TITLE","type":"song"

**Note**

The property "type" is always returned.

**Note**

On dzVents function `openURL`.

- The **postdata** MUST be encoded using the backslash character - if not done, Kodi cannot parse the request.

- The **header** MUST be from content type application/json (which is set explicitly to show its important).
- The HTTP response string must be unique across all Domoticz Automation Events.

### dzVents script snippet

```
local KODI_IP = 'http://player-ip:8080/jsonrpc'
local KODI_JSONRPC_GETITEM =
'{"jsonrpc":"2.0","method":"Player.GetItem","params": {"properties":["title","album","artist","duration"], "playerid":0}, "id": "AudioGetItem"}'
local KODI_RES = "KODISONGPLAYING"

-- HTTP GET Request to readdata
local urlRequest = string.format("%s?request=%s", KODI_IP, KODI_JSONRPC_GETITEM)
domoticz.openURL({url = urlRequest, method = 'GET', callback = KODI_RES})

-- HTTP Response
--
{"id": "AudioGetItem", "jsonrpc": "2.0", "result": {"item": {"album": "ALBUM", "artist": ["ARTIST"], "duration": 277, "id": 233, "label": "LABEL", "title": "TITLE", "type": "song"}}}
-- Pick from the property result.item the properties artist & title
local song = item.json.result.item
-- note the artist is the first entry (1) of the artist array
domoticz.log(string.format("%s - %s", song.artist[1], song.title))
```

Trigger: Timer every minute.

### Write Data to Kodi

To write data to Kodi, use the **POST method**.

Example Kodi JSON-RPC method: *Player.PlayPause*.

### dzVents script snippet

```
local KODI_IP = 'http://player-ip:8080/jsonrpc'
local KODI_RES = "KODISONGPLAYBACK"
local KODI_JSONRPC_POSTDATA =
'{"jsonrpc": "2.0", "method": "Player.PlayPause", "params": {"playerid": 0}, "id": 1}

-- HTTP POST Request to ControlPlayback
local urlRequest = string.format("%s", KODI_IP)
domoticz.openURL({url = urlRequest, method = 'POST', headers = { ['content-type'] = 'application/json' }, callback = KODI_RES, postData = KODI_JSONRPC_POSTDATA })

-- HTTP Response
-- {"id": 1, "jsonrpc": "2.0", "result": {"speed": 0}}
local speed = item.json.result.speed
```

The trigger to set the player state to play or pause is via Domoticz Device change using a selector switch (virtual sensor).

## JSON-RPC API HTTP Request Examples

The JSON-RPC API HTTP requests can be tested prior using in dzVents via CLI **curl** command with the JSON-RPC API command as parameter.

The next examples have been tested on the Kodi system via the CLI with url:port localhost:8080.

For more commands, see Chapter Kodi JSONRPC Commands (Selection).

### GET - Player.GetItem

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetItem\", \"params\": { \"properties\": [\"title\"], \"playerid\": 0 }, \"id\": \"AudioGetItem\"}" http://localhost:8080/jsonrpc
```

#### HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"id":606,"label":"SONGLABEL","title":"SONGTITLE","type":"song"}}}
```

### GET - Player.GetProperties

This method can be used to check if the player is playing or paused, i.e. speed=1 or speed=0

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\": { \"playerid\": 0, \"properties\":[\"speed\", \"time\"]}, \"id\": \"AudioGetItem\"}" http://player-ip:8080/jsonrpc
```

#### HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"speed":1,"time":{"hours":0,"milliseconds":287,"minutes":0,"seconds":0}}}
```

### Check the total time and time played for the current song playing

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\": { \"playerid\": 0, \"properties\":[\"speed\", \"percentage\", \"time\", \"totaltime\"]}, \"id\": \"AudioGetItem\"}" http://player-ip:8080/jsonrpc
```

#### HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":25.5638370513916,"speed":1,"time":{"hours":0,"milliseconds":268,"minutes":1,"seconds":4},"totaltime":{"hours":0,"milliseconds":402,"minutes":4,"seconds":11}}}
```

### POST - Player.PlayPause

```
curl -X POST -H "Content-Type: application/json" -d '{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":0}, "id":"AudioGetItem"}' http://localhost:8080/jsonrpc
```

#### HTTP Response

```
{"id":1,"jsonrpc":"2.0","result":{"speed":0}}
```

The player set to paused. Running again sets the speed to 1 and the item is playing.

## Domoticz Explore

Several devices with single Automation Scripts dzVents Lua are created to explore communicating and controlling a Kodi Music Player.

The devices created are from hardware type "Dummy" Virtual Sensors. The hardware entry is named "VirtualDevices".

For the JSONRPC commands, it is strongly recommended to escape the JSON formatted string (i.e. character " to \").

The focus of the examples is to show how to use the HTTP JSON-RPC API command requests GET to read data or POST to write data.

There are many commands available, but the shared examples are used as a base for any other solutions.

### *Hint*

The devices used are updated rather frequent, which means the Domoticz database growth in size. Recommend clearing the device logs regularly either manual or Domoticz settings or use JSON/API request.

### Get Song Playing Timer

Display the current song playing; Lists the history of songs played including error messages.

Device(s)	117,VirtualDevices,Kodi Song,General,Text,DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playing_timer.dzvents
Trigger	Timer every minute
Async HTTP request method	GET
JSON-RPC API Command	local JSONRPC_GETITEM = "{"jsonrpc":"2.0", "method":"Player.GetItem", "params":{ "properties":["title", "album", "artist", "duration"], "playerid":0}, "id":"AudioGetItem"}"

### Automation Event

```
--[[
-- kodi_explore_playing_timer.dzvents
-- Get the current song playing from the Kodi Media Center and display in a text device
-- Uses a timer and updates every minute - highest update interval possible via timer
-- url get (=readdata) request from the kodi server, to obtain artist & title playing
-- http example:
-- http://player-
ip:8080/jsonrpc?request={"jsonrpc":"2.0","method":"Player.GetItem","params":{"properties":["title","alb
um","artist","duration"],"playerid":0}, "id":1}
-- http response JSON string:
--
{"id":1,"jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":274,"id":227,"
label":"LABEL","title":"TITLE","type":"song"}}}
-- Notes:
-- * Method Player.GetItem still responses a title if the song played is paused.
-- * Introspect: detailed information about an API method - example Player.GetItem
-- http://player-ip:8080/jsonrpc?request={"jsonrpc": "2.0", "method": "JSONRPC.Introspect", "params": {
"filter": { "id": "Player.GetItem", "type": "method" } }, "id": 1 }
--
-- 20200523 rwbl
]]--
```

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)  
local REQUEST\_URL = "http://player-ip:8080/jsonrpc"

```
-- Event unique callback
local REQUEST_CALLBACK = "KODIPLAYINGTIMER"
-- Request for method Player.GetItem for artist:title
local JSONRPC_GETITEM =
"{"jsonrpc":"2.0","method":"Player.GetItem","params":{properties:[{"title"}, {"album"}, {"artist"}, {"duration"}], "playerid":0}, "id": "AudioGetItem"
-- Domoticz device - text song playing artist:title
local IDX_PLAYING = 117

return {
  on = { timer = { 'every minute' }, httpResponses = { REQUEST_CALLBACK } },
  data = { songprevious = { initial = "" } },
  execute = function(domoticz, item)
    -- timer every minute
    if (item.isTimer) then
      local urlRequest = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM)
      domoticz.log(urlRequest)
      domoticz.openURL({url = urlRequest, method = 'GET', callback = REQUEST_CALLBACK})
    end

    if (item.isHTTPResponse) then
      domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
      item.statusCode))
      domoticz.log(string.format('Item data: %s', item.data))
      if (item.statusCode == 200 and item.callback == REQUEST_CALLBACK) then
        if (item.isJSON) then
          -- Pick the song artist & title
          local song = item.json.result.item
          -- Check if there is a title
          local songcurrent = "Nothing playing"
          if (song.title ~= "") then
            -- note the artist is the first entry (1) of the artist array
            songcurrent = string.format("%s:%s", song.artist[1], song.title)
          end
          domoticz.log(string.format("Song playing: %s", songcurrent))
          -- check if the song has changed
          if (songcurrent ~= domoticz.data.songprevious) then
            domoticz.log(string.format("Song playing changed: %s to %s",
            domoticz.data.songprevious, songcurrent))
            -- Update the text device
            domoticz.devices(IDX_PLAYING).updateText(songcurrent)
            -- Update the data
            domoticz.data.songprevious = songcurrent
          end
        end
        return
      else
        domoticz.devices(IDX_PLAYING).updateText(item.statusText)
        domoticz.log('[ERROR] Handling the request: ' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## Domoticz Log

```
2020-05-23 11:22:00.642 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_timer: HTTPResponse: "KODIEXPLOREPLAYINGTIMER"
2020-05-23 11:22:00.643 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGTIMER, Status Code = 200
2020-05-23 11:22:00.643 Status: dzVents: Info: Item data:
{"id": "AudioGetItem", "jsonrpc": "2.0", "result": {"item": {"album": "ALBUM", "artist": ["ARTIST"], "duration": 254, "id": 1531, "label": "LABEL", "title": "TITLE", "type": "song"}}}
2020-05-23 11:22:00.643 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 11:22:00.643 Status: dzVents: Info: Song playing changed: Nothing playing to ARTIST:TITLE
2020-05-23 11:22:00.662 Status: dzVents: Info: ----- Finished kodi_explore_playing_timer
```

## Get Song Playing & Progress Timer

Display the current song playing with progress Percentage, Time, Total Time.

There are two JSON-RPC API commands required: GetItem for Artist:Title and GetProperties for Percentage, Time, Total Time.

The commands are initiated every minute by a timer starting with GetItem.

If GetItem completes, GetProperties is initiated.

*Note*

The log of the devices (i.e. progress) can contain many entries if the player runs for a while. Clear the devices from time to time.

### Challenge

The devices are updated ONLY every minute because this is the highest frequency possible by the Domoticz event system.

It would be good to have a higher device update frequency, let us say every 10 seconds or even ad hoc.

See next example exploring solutions.

Device(s)	117,VirtualDevices,Kodi Song,General,Text,DATA 123,VirtualDevices,Player Progress,General,Text,DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playing_progress_timer.dzvents
Trigger	Timer every minute
Async HTTP request method	GET
JSON-RPC API Command	local JSONRPC_GETITEM = "{"jsonrpc": "2.0", "method": "Player.GetItem", "params": {"properties": ["title", "album", "artist", "duration"], "playerid": 0}, "id": "AudioGetItem"}"
JSON-RPC API Command	local JSONRPC_GETPROPERTIES = "{"jsonrpc": "2.0", "method": "Player.GetProperties", "params": {"playerid": 0, "properties": ["speed", "percentage", "time", "totaltime"]}, "id": "AudioGetItem"}"

### Automation Event

```
-- [[
-- kodi_explore_playing_progress_timer.dzvents
-- Get the current song playing and the progress percentage, time, totaltime from the Kodi Media Center
and display in two text devices.
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
-- Event unique callbacks
local REQUEST_GETITEM_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETITEM"
local REQUEST_GETPROPERTIES_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
-- Request for method Player.GetItem for artist:title
local JSONRPC_GETITEM =
"{"jsonrpc": "2.0", "method": "Player.GetItem", "params": {"properties": ["title", "album", "a
rtist", "duration"], "playerid": 0}, "id": "AudioGetItem"}"
-- Request for method Player.GetProperties for the total time and time played for the current song
playing
local JSONRPC_GETPROPERTIES = '{"jsonrpc": "2.0", "method": "Player.GetProperties", "params":  
{"playerid": 0, "properties": ["speed", "percentage", "time", "totaltime"]}, "id":  
"AudioGetItem"}'
```

```

-- Domoticz devices
local IDX_PLAYING = 117      -- text song playing artist:title
local IDX_PROGRESS = 123      -- percentage, time, totaltime

return {
  on = {
    timer = { 'every minute' },
    httpResponses = { REQUEST_GETITEM_CALLBACK, REQUEST_GETPROPERTIES_CALLBACK }
  },
  data = {
    songprevious = { initial = "" }
  },
  execute = function(domoticz, item)
    -- timer every minute
    if (item.isTimer) then
      domoticz.openURL({
        url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM),
        method = 'GET',
        callback = REQUEST_GETITEM_CALLBACK})
    end

    if (item.isHTTPResponse) then
      domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
      item.statusCode))
      domoticz.log(string.format('Item data: %s', item.data))
      if (item.statusCode == 200) then

        if (item.callback == REQUEST_GETITEM_CALLBACK and item.isJSON) then
          --
          {"id":"AudioGetItem","jsonrpc":"2.0","result": {"item": {"album": "ALBUM", "artist": ["ARTIST"], "duration": 54, "id": 1531, "label": "LABEL", "title": "TITLE", "type": "song"} } }
          -- Pick the song artist & title
          local song = item.json.result.item
          -- Check if there is a title
          local songcurrent = "Nothing playing"
          if (song.title ~= "") then
            -- note the artist is the first entry (1) of the artist array
            songcurrent = string.format("%s:%s", song.artist[1], song.title)
          end
          domoticz.log(string.format("Song playing: %s", songcurrent))
          -- check if the song has changed
          if (songcurrent ~= domoticz.data.songprevious) then
            domoticz.log(string.format("Song playing changed: %s to %s",
            domoticz.data.songprevious, songcurrent))
            -- Update the text device
            domoticz.devices(IDX_PLAYING).updateText(songcurrent)
            -- Update the data
            domoticz.data.songprevious = songcurrent
          end
          domoticz.openURL({
            url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETPROPERTIES),
            method = 'GET',
            callback = REQUEST_GETPROPERTIES_CALLBACK})
        end

        if (item.callback == REQUEST_GETPROPERTIES_CALLBACK and item.isJSON) then
          --
          {"id": "AudioGetItem", "jsonrpc": "2.0", "result": {"percentage": 25.5638370513916, "speed": 1, "time": {"hours": 0, "milliseconds": 268, "minutes": 1, "seconds": 4}, "totaltime": {"hours": 0, "milliseconds": 402, "minutes": 4, "seconds": 11}} }
          -- init percentage with 0%
          local progress = string.format("%s", "0%")
          -- check seconds played
          if (item.json.result.time.seconds > 0) then
            -- set progress with percentage, time played and totaltime
            progress = string.format('%.1f%% / %02d:%02d:%02d / %02d:%02d:%02d',
            item.json.result.percentage,
            item.json.result.time.hours, item.json.result.time.minutes,
            item.json.result.time.seconds,
            item.json.result.totaltime.hours, item.json.result.totaltime.minutes, item.
            json.result.totaltime.seconds
          )
          end
          domoticz.devices(IDX_PROGRESS).updateText(progress)
        end
      end
    end
  end
}

```

```

        else
            domoticz.devices(IDX_PLAYING).updateText(item.statusText)
            domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
            return
        end
    end

end
}

```

## Domoticz Log

```

2020-05-23 12:27:00.228 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 12:27:00.363 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 12:27:00.363 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_progress: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 12:27:00.364 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETITEM, Status Code
= 200
2020-05-23 12:27:00.364 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ALTIST"],"duration":2
44,"id":1542,"label":"LABEL","title":"TITLE","type":"song"}}}
2020-05-23 12:27:00.364 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 12:27:00.364 Status: dzVents: Info: Song playing changed: ARTIST:TITLE to ARTIST:TITLE
2020-05-23 12:27:00.383 Status: dzVents: Info: ----- Finished kodi_explore_playing_progress
2020-05-23 12:27:00.383 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 12:27:00.510 Status: dzVents: Info: Handling events for: "Player Playing", value:
"ARTIST:TITLE"
2020-05-23 12:27:00.510 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_progress: Device: "Player Playing (VirtualDevices)", Index: 117
2020-05-23 12:27:00.511 Status: dzVents: Info: ----- Finished kodi_explore_playing_progress
2020-05-23 12:27:00.584 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 12:27:00.585 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_progress: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 12:27:00.586 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES,
Status Code = 200
2020-05-23 12:27:00.586 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":12.370973587036133,"speed":1,"time":{"hours
":0,"milliseconds":238,"minutes":0,"seconds":30},"totaltime":{"hours":0,"milliseconds":427,"minutes":4,
"seconds":4}}}
2020-05-23 12:27:00.604 Status: dzVents: Info: ----- Finished kodi_explore_playing_progress
2020-05-23 12:27:00.604 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua

```

## Get Song Playing & Progress afterSec

This example is based on the previous example updating the devices via timer every minute.

### Challenge

The devices are updated ONLY every minute because this is the highest frequency possible by the Domoticz event system.

It would be good to have a higher device update frequency, let us say every 10 seconds or even ad hoc.

### How to do so - explore solutions

- Method afterSec to control a delay for the method openURL(). This solution is tested and working fine (used by the function Music\_Player)
- Timer with a for loop (for var=start\_value,end\_value,step\_value do GETITEM followed by GETPROPERTIES end) within the trigger. This solution is not worked out further.
- Listen to MQTT messages from other applications, i.e. Node-RED

Read [Timer Update Less One Minute](#) for the solutions worked out.

### Solution dzVents Method afterSec

Make use of the dzVents device method afterSec(NN) to control a delay for the method openURL().

#### Pseudocode

```
DELAYSECONDS = 15
Trigger openURL(GETITEM, CALLBACK_GETITEM)
Handle CALLBACK_GETITEM
  Run openURL(GETPROPERTIES, CALLBACK_GETPROPERTIES)
Handle CALLBACK_GETPROPERTIES
  Run openURL(GETITEM, CALLBACK_GETITEM).afterSec(DELAYSECONDS)
```

#### Notes

- The trigger could be a device change, i.e. setting a switch to ON (see below) or a selector to play.
- The change of the switch device triggers the first openURL for the JSON-RPC API GETITEM.
- The GETITEM callback runs openURL for the JSON-RPC API GETPROPERTIES.
- The GETPROPERTIES callback runs GETITEM again, but after a delay, set by the constant DELAYSECONDS to 15 seconds.
- After the delay, the loop starts again from the GETITEM callback.

This solution is used for the function Music Player.

### Automation Event

```
-- [[
-- kodi_explore_playing_progress_aftersec.dzvents
-- Get the current song playing and the progress percentage, time, totaltime from the Kodi Media Center
and display in two text devices.
-- The data is refreshed every 10 seconds handled by afterSec method initiated by a switch device.
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
-- Event unique callbacks
```

```

local REQUEST_GETITEM_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETITEM"
local REQUEST_GETPROPERTIES_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
-- Request for method Player.GetItem for artist:title
local JSONRPC_GETITEM =
"{"jsonrpc":"2.0","method":"Player.GetItem","params":{ "properties": [\"title\", \"album\", \"artist\", \"duration\"], \"playerid\":0}, \"id\":\"AudioGetItem\"}"
-- Request for method Player.GetProperties for the total time and time played for the current song
playing
local JSONRPC_GETPROPERTIES = '{ "jsonrpc": "2.0", "method": "Player.GetProperties", "params": { "playerid": 0, "properties": [\"speed\", \"percentage\", \"time\", \"totaltime\"]}, "id": "AudioGetItem" }'

-- Refresh Artist:Title & Progress Rate every NN seconds - afterSec(NN)
local REFRESHRATE = 10

-- Domoticz devices
local IDX_PLAYING = 117          -- Text song playing artist:title
local IDX_PROGRESS = 123          -- Text percentage, time, totaltime
local IDX_REMOTECONTROL = 126     -- Switch to enable or disable the player

local MSG_PLAYERTURNEDOFF = "Player turned Off"
local MSG_PLAYERTURNEDON = "Player turned On"
local MSG_INITPROGRESS = "0%"

-- clear the various device logs to limit database size with not needed entries
local function clearDeviceLogs(domoticz)
    local ip = "http://domoticz-ip:8080/json.htm"
    local urlBase = string.format('%s?type=command&param=clearlightlog&idx=', ip)
    devices = {IDX_PLAYING, IDX_PROGRESS}
    for key,idx in ipairs(devices) do
        domoticz.openURL({url = string.format('%s%d', urlBase, idx), method = 'GET'})
    end
    domoticz.log("Device logs cleared.")
end

return {
on = {
    devices = {
        IDX_REMOTECONTROL           -- turn the player on or off
    },
    httpResponses = {
        REQUEST_GETITEM_CALLBACK,      -- get artist:title
        REQUEST_GETPROPERTIES_CALLBACK -- get percetage,time,totaltime
    },
    data = {
        songprevious = { initial = "" }
    },
execute = function(domoticz, item)

    -- Get the player remote control state
    playerRemoteControl = domoticz.devices(IDX_REMOTECONTROL).nValue

    -- Check device
    -- If ON run GETITEM request first time
    if (item.isDevice and item.idx == IDX_REMOTECONTROL) then
        domoticz.log(string.format("%s=%s|%d|%s", item.name, item.state, item.nValue, item.sValue))
        -- reset songprevious
        domoticz.data.songprevious = ""
        -- Player remote control turned OFF
        if (playerRemoteControl == 0) then
            domoticz.devices(IDX_PLAYING).updateText(MSG_PLAYERTURNEDOFF)
            domoticz.devices(IDX_PROGRESS).updateText(MSG_INITPROGRESS)
            -- OPTION to clear the log of the various devices
            clearDeviceLogs(domoticz)
        else
            -- Player remote control turned ON
            domoticz.devices(IDX_PLAYING).updateText(MSG_PLAYERTURNEDON)
            domoticz.devices(IDX_PROGRESS).updateText(MSG_INITPROGRESS)
        end
    end
end
}

```

```

if (item.isHTTPResponse) then
    -- stop if playerRemoteControl is OFF
    if (playerRemoteControl == 0) then
        return
    end
    --
    domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
    domoticz.log(string.format('Item data: %s', item.data))
    if (item.statusCode == 200) then

        if (item.callback == REQUEST_GETITEM_CALLBACK and item.isJSON) then
            --
            {"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
54,"id":1531,"label":"LABEL","title":"TITLE","type":"song"}}}
            -- Pick the song artist & title
            local song = item.json.result.item
                -- Check if there is a title
                local songcurrent = "Nothing playing"
                if (song.title ~= "") then
                    -- note the artist is the first entry (1) of the artist array
                    songcurrent = string.format("%s:%s", song.artist[1], song.title)
                end
                domoticz.log(string.format("Song playing: %s", songcurrent))
                -- check if the song has changed
                if (songcurrent ~= domoticz.data.songprevious) then
                    domoticz.log(string.format("Song playing changed: %s to %s",
domoticz.data.songprevious, songcurrent))
                    -- Update the text device
                    domoticz.devices(IDX_PLAYING).updateText(songcurrent)
                    -- Update the data
                    domoticz.data.songprevious = songcurrent
                end
            domoticz.openURL({
                url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETPROPERTIES),
                method = 'GET',
                callback = REQUEST_GETPROPERTIES_CALLBACK})
        end

        if (item.callback == REQUEST_GETPROPERTIES_CALLBACK and item.isJSON) then
            --
            {"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":25.5638370513916,"speed":1,"time":{"hours":0,
"milliseconds":268,"minutes":1,"seconds":4},"totaltime":{"hours":0,"milliseconds":402,"minutes":4,"se
conds":11}}}
            -- init percentage with 0%
            local progress = string.format("%s", "0%")
            -- check seconds played
            if (item.json.result.time.seconds > 0) then
                -- set progress with percentage, time played and totaltime
                progress = string.format('%.1f%% / %02d:%02d:%02d / %02d:%02d:%02d',
                    item.json.result.percentage,
                    item.json.result.time.hours, item.json.result.time.minutes,
item.json.result.time.seconds,
                    item.json.result.totaltime.hours, item.json.result.totaltime.minutes,item.
json.result.totaltime.seconds
                )
            end
            domoticz.devices(IDX_PROGRESS).updateText(progress)
            -- Start next GETITEM request after NN seconds
            domoticz.openURL({
                url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM),
                method = 'GET',
                callback = REQUEST_GETITEM_CALLBACK}).afterSec(REFRESHRATE)
            end
        else
            domoticz.devices(IDX_PLAYING).updateText(item.statusText)
            domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
            return
        end
    end
end
}

```

## Domoticz Log

```

2020-05-23 19:02:58.611 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 19:03:07.942 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 19:03:07.942 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_progress_aftersec: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 19:03:07.972 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETITEM, Status Code
= 200
2020-05-23 19:03:07.972 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":290,"id":2322,"label":"LABEL","title":"TITLE","type":"song"}}}
2020-05-23 19:03:07.972 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 19:03:07.973 Status: dzVents: Info: ----- Finished kodi_explore_playing_progress_aftersec
2020-05-23 19:03:07.973 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 19:03:08.093 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 19:03:08.093 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playing_progress_aftersec: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 19:03:08.112 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES,
Status Code = 200
2020-05-23 19:03:08.112 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":94.3416748046875,"speed":1,"time":{"hours":0,"milliseconds":872,"minutes":4,"seconds":33},"totaltime":{"hours":0,"milliseconds":298,"minutes":4,"seconds":50}}}
2020-05-23 19:03:08.114 Status: dzVents: Info: ----- Finished kodi_explore_playing_progress_aftersec

```

## Solution dzVents for Loop

Timer with a for loop within the time trigger.

### Pseudocode

```

for seconds_value = start_value, end_value, step_value do
    GETITEM.afterSec(seconds_value) followed by GETPROPERTIES
end

```

If the time trigger is set by every minute and the data to be updated every 10 seconds, then the for loop variables could be:

start\_value=0, end\_value=50, step\_value=10. The seconds\_value gets the 5 values 10,20,30,40,50.

This solution has not been worked out further.

## Solution MQTT with Node-RED

This solution uses Node-RED as the *man-in-the-middle*, i.e. between the player and Domoticz.

Domoticz to listen to MQTT messages published by Node-RED with topic "domoticz/in" (for incoming messages) and let Domoticz update the device data.

This requires the Domoticz hardware, like "MQTT Client with LAN Interface".

The MQTT messages published by Node-RED could either via PULL or PUSH:

**PULL** data from the player in regular intervals using the "inject" node with a trigger repeating every NN seconds submitting a HTTP request.

The HTTP request contains the JSON-RPC API command "Player.GetItem" as used previous (with some extra properties just to check out):

```
{"jsonrpc":"2.0","method":"Player.GetItem","params":{"properties":["title","album","artist","duration","thumbnail","file","fanart","streamdetails"],"playerid":0},"id":"AudioGetItem"}
```

OR

**PUSH** data from the player by creating a websocket using the "websocket in" node (port:9090 as configured in the player) and use JSON-RPC API callback method.  
A configuration node "websocket-listener" acting as a WebSocket Server endpoint using the path: ws://player-ip:9090/jsonrpc and send/receive: payload.

Publishing of the message via a "mqtt out" node to the Domoticz topic "domoticz/in".

Payload example or the Domoticz device with idx 117 ("Player Playing") which receives as msg.payload the string "ARTIST:TITLE".

This msg.payload is extracted from the result of the JSON-RPC API Player.GetItem.

```
const IDX = 117;
const ARTISTTITLE = msg.payload; // ARTIST:TITLE
msg.payload = {"command": "udevice", "idx" : IDX, "nvalue" : 0, "svalue" : ARTISTTITLE}
```

### Domoticz Log

```
2020-05-24 09:57:44.860 MQTT: Topic: domoticz/in, Message:
{"command": "udevice", "idx": 117, "nvalue": 0, "svalue": "ARTIST:TITLE"}
```

Node-RED Flow Source: kodi\_explore\_node-red-domoticz-mqtt.flow

## Set Player to Play or Pause

This is a POST request, to change the player play mode to PLAY or PAUSE.

Device(s)	119, VirtualDevices, Kodi PlayPause, Light/Switch, Switch, Selector, DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playback_device.dzvents
Trigger	Device change state with IDX=119
Async HTTP request method	POST
JSON-RPC API Command	See automation event source local constants, like JSONRPC_PLAY, JSONRPC_PAUSE

## Automation Event

```
--[[[  
-- kodi_explore_playpause_device.dzvents  
-- Change the player play mode to PLAY or PAUSE using a selector switch  
-- 20200523 rwbl  
]]--  
  
-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)  
local REQUEST_URL = "http://player-ip:8080/jsonrpc"  
-- Event unique callback (=eventname in uppercase without underscores  
local REQUEST_CALLBACK = "KODIEXPLOREPLAYPAUSE"  
-- Command for method to set the player to play, pause  
local JSONRPC_PLAY =  
"{"jsonrpc":"2.0","method":"Player.PlayPause","params":{\"playerid\":0,\"play\":true},\"id\":  
\"AudioGetItem\"}"  
local JSONRPC_PAUSE =  
"{"jsonrpc":"2.0","method":"Player.PlayPause","params":{\"playerid\":0,\"play\":false},\"id\":  
\"AudioGetItem\"}"  
-- Domoticz device  
local IDX_PLAYPAUSE = 119  
local LEVEL_PLAY = 10  
local LEVEL_PAUSE = 20  
-- HINT: instead, constants for the jsonrpc command and the device levels, a lua table could be used  
local modePlayPause = {  
    [10] =  
    {"jsonrpc":"2.0","method":"Player.PlayPause","params":{\"playerid\":0,\"play\":true},\"id\":  
    \"AudioGetItem\"},  
    [20] =  
    {"jsonrpc":"2.0","method":"Player.PlayPause","params":{\"playerid\":0,\"play\":false},\"id\":  
    \"AudioGetItem\""},  
}  
  
return {  
    on = {  
        devices = { IDX_PLAYPAUSE },  
        httpResponses = { REQUEST_CALLBACK }  
    },  
    execute = function(domoticz, item)  
        -- check device  
        if (item.isDevice and item.idx == IDX_PLAYPAUSE) then  
            local jsonrpcCommand = ""  
            -- set jsonrpc command using constants  
            if (item.level == LEVEL_PLAY) then jsonrpcCommand = JSONRPC_PLAY end  
            if (item.level == LEVEL_PAUSE) then jsonrpcCommand = JSONRPC_PAUSE end  
            domoticz.log(string.format("CONST=Level selected: %d, %s", item.level, jsonrpcCommand))  
            -- set jsonrpc command using lua table  
            jsonrpcCommand = modePlayPause[item.level]  
            domoticz.log(string.format("TABLE=Level selected: %d, %s", item.level, jsonrpcCommand))  
            domoticz.openURL({  
                url = REQUEST_URL,  
                method = 'POST',  
                headers = { ['content-type'] = 'application/json' },  
                callback = REQUEST_CALLBACK,  
            })  
        end  
    end  
}
```

```
        postData = jsonrpcCommand})
    domoticz.log(string.format("POST Request: %s, %s", REQUEST_URL, jsonrpcCommand))
end

if (item.isHTTPResponse) then
    domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
    domoticz.log(string.format('Item data: %s', item.data))
    if (item.statusCode == 200 and item.callback == REQUEST_CALLBACK) then
        domoticz.log(string.format("[INFO] Handling request: %s", item.statusText))
    else
        domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
    end
end
end
end
}
```

## Domoticz Log

The log shows the playpause mode set to pause = method Player.PlayPause with parameter play:false.

```
2020-05-23 11:52:45.448 Status: User: Admin initiated a switch command (119/Player PlayPause/Set Level)
2020-05-23 11:52:45.594 Status: dzVents: Info: Handling events for: "Player PlayPause", value: "Pause"
2020-05-23 11:52:45.594 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playpause_device: Device: "Player PlayPause (VirtualDevices)", Index: 119
2020-05-23 11:52:45.594 Status: dzVents: Info: CONST=Level selected: 20,
{"jsonrpc": "2.0", "method": "Player.PlayPause", "params": {"playerid": 0, "play": false}, "id": "AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: TABLE=Level selected: 20,
{"jsonrpc": "2.0", "method": "Player.PlayPause", "params": {"playerid": 0, "play": false}, "id": "AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: POST Request: http://player-ip:8080/jsonrpc,
{"jsonrpc": "2.0", "method": "Player.PlayPause", "params": {"playerid": 0, "play": false}, "id": "AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: ----- Finished kodi_explore_playpause_device
2020-05-23 11:52:45.595 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 11:52:45.793 Status: dzVents: Info: Handling httpResponse-events for: "KODIEXPLOREPLAYPAUSE"
2020-05-23 11:52:45.793 Status: dzVents: Info: ----- Start internal script:
kodi_explore_playpause_device: HTTPResponse: "KODIEXPLOREPLAYPAUSE"
2020-05-23 11:52:45.794 Status: dzVents: Info: Callback: KODIEXPLOREPLAYPAUSE, Status Code = 200
2020-05-23 11:52:45.794 Status: dzVents: Info: Item data:
{"id": "AudioGetItem", "jsonrpc": "2.0", "result": {"speed": 0}}
2020-05-23 11:52:45.794 Status: dzVents: Info: [INFO] Handling request: OK
2020-05-23 11:52:45.794 Status: dzVents: Info: ----- Finished kodi_explore playpause device
```

## Appendix: Kodi JSONRPC Command (Selection)

### *Note*

Ensure to escape special characters with a backslash ()).

Test with curl prior using in scripts.

### **Example**

This method can be used to check if the player is playing or paused, i.e. speed=1 or speed=0  
Curl Command:

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\": {\"playerid\": 0, \"properties\": [\"speed\", \"time\"]}, \"id\": \"AudioGetItem\"}" http://player-ip:8080/jsonrpc
```

### HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"speed":1,"time":{"hours":0,"milliseconds":287,"minutes":0,"seconds":0}}}
```

## Transport

Play pause:

```
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{ "playerid": 1 }, "id":1}
```

Play:

```
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{ "playerid":1,"play":true}, "id":1}
```

Pause:

```
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{ "playerid":1,"play":false}, "id":1}
```

Stop:

```
{"jsonrpc":"2.0", "method":"Player.Stop", "params":{ "playerid":1}, "id":1}
```

Next:

```
{"jsonrpc":"2.0","method":"Player.GoTo","params":{ "playerid":1,"to":"next"}, "id":1}
```

Previous:

```
{"jsonrpc":"2.0","method":"Player.GoTo","params":{ "playerid":1,"to":"previous"}, "id":1}
```

Fwd with speed 2:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":2}, "id":1}
```

Fwd with speed 4:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":4}, "id":1}
```

Fwd with speed 8:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":8}, "id":1}
```

Fwd with speed 16:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":16}, "id":1}
```

Fwd with speed 32:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":32}, "id":1}
```

Rwd with speed 2:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":-2}, "id":1}
```

Rwd with speed 4:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":-4}, "id":1}
```

Rwd with speed 8:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":-8}, "id":1}
```

Rwd with speed 16:

```
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{ "playerid":1,"speed":-16}, "id":1}
```

Rwd with speed 32:

```
{"jsonrpc": "2.0", "method": "Player.SetSpeed", "params": {"playerid": 1, "speed": -32}, "id": 1}
```

Jump forward:

```
{"jsonrpc": "2.0", "method": "Player.Seek", "params": {"playerid": 1, "value": "smallforward"}, "id": 1}
```

Jump backward:

```
{"jsonrpc": "2.0", "method": "Player.Seek", "params": {"playerid": 1, "value": "smallbackward"}, "id": 1}
```

## Repeat and Shuffle

Repeat one:

```
{"jsonrpc": "2.0", "method": "Player.SetRepeat", "params": {"playerid": 1, "repeat": "one"}, "id": 1}
```

Repeat all:

```
{"jsonrpc": "2.0", "method": "Player.SetRepeat", "params": {"playerid": 1, "repeat": "all"}, "id": 1}
```

Shuffle on:

```
{"jsonrpc": "2.0", "method": "Player.SetShuffle", "params": {"playerid": 1, "shuffle": true}, "id": 1}
```

Shuffle false:

```
{"jsonrpc": "2.0", "method": "Player.SetShuffle", "params": {"playerid": 1, "shuffle": false}, "id": 1}
```

Repeat off:

```
{"jsonrpc": "2.0", "method": "Player.SetRepeat", "params": {"playerid": 1, "repeat": "off"}, "id": 1}
```

## Navigation

Back:

```
{"jsonrpc": "2.0", "method": "Input.Back", "id": 1}
```

Select:

```
{"jsonrpc": "2.0", "method": "Input.Select", "id": 1}
```

Left:

```
{"jsonrpc": "2.0", "method": "Input.Left", "id": 1}
```

Right:

```
{"jsonrpc": "2.0", "method": "Input.Right", "id": 1}
```

Up:

```
{"jsonrpc": "2.0", "method": "Input.Up", "id": 1}
```

Down:

```
{"jsonrpc": "2.0", "method": "Input.Down", "id": 1}
```

Home:

```
{"jsonrpc": "2.0", "method": "Input.Home", "id": 1}
```

Info:

```
{"jsonrpc": "2.0", "method": "Input.info", "id": 1}
```

Show OSD:

```
{"jsonrpc": "2.0", "method": "Input.showOSD", "id": 1}
```

Context Menu: {"jsonrpc": "2.0", "method": "Input.ContextMenu", "id": 1}

Show codec information of playing item:

```
{"jsonrpc": "2.0", "method": "Input.ShowCodec", "id": 1}
```

Page Down:

```
{"jsonrpc":"2.0","method":"input.executeaction","params":{"action":"pagedown"},"id":1}
```

Page Up:

```
{"jsonrpc":"2.0","method":"input.executeaction","params":{"action":"pageup"},"id":1}
```

## Additional

Volume up:

```
{"jsonrpc":"2.0","method":"Application.SetVolume","params":{"volume":"increment"},"id":1}
```

Volume down:

```
{"jsonrpc":"2.0","method":"Application.SetVolume","params":{"volume":"decrement"},"id":1}
```

Get Playlists:

```
{"jsonrpc":"2.0","method":"Playlist.GetItems","params":{"properties":["title","album","artist","duration"],"playlistid":1},"id":1}
```

Get Albums:

```
{"jsonrpc":"2.0","method":"AudioLibrary.GetAlbums","params":{"properties":["playcount","artist","genre"],"limits":{"end":10,"start":0},"sort":{"order":"ascending","method":"album","ignorearticle":true}),"id":"libAlbums"}
```

Activate Genres:

```
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://1/"]},"id":1}
```

Activate artists:

```
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://2/"]},"id":1}
```

Activate albums:

```
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://3/"]},"id":1}
```

Activate songs:

```
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://4/"]},"id":1}
```

## System

Reboot:

```
{"jsonrpc":"2.0","method":"System.Reboot","id":1}
```

Shutdown:

```
{"jsonrpc":"2.0","method":"SystemShutdown","id":1}
```

## Appendix: Various Hints

### Config.txt Modifications

LibreELEC is read-only by default ([Info](#))

Modify config.txt via SSH:

```
ssh root@player-ip
```

#### Set write-permissions

```
mount -o remount,rw /flash
```

#### Edit config.txt

```
nano /flash/config.txt
```

#### Set readonly-permissions:

```
mount -o remount,ro /flash
```

Reboot for the changes to take effect.

### Python

Kodi uses python 2.7

Test from CLI

```
python
Python 2.7.16 (default, Mar  5 2020, 07:14:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

## RPi Button Shutdown & Restart

Add a push-button to enable shutdown and restart.

The push-button is connected to GPIO03 (=SCL pin #05). The dtoverlay gpio-shutdown is used.

Pressing the push-button once: shortens the GPIO3 (pin 5) to ground/0V, which is the same as the shutdown command. Both shutdown the OS and go into a low-power halt state.

Pressing the push-button again: the Raspberry restarts from the halt state.

**NOTE:** On the old RPi2 did not find way to use the push-button connected to a pin other than GPIO3.

The GPIO3 pin is also the SCL pin, means not possible to connect a I2C device, i.e. LCD display.

### Reference

Name: gpio-shutdown

Info: Initiates a shutdown when GPIO pin changes. The given GPIO pin is configured as an input key that generates KEY\_POWER events. This event is handled by systemd-logind by initiating a shutdown. This overlay only handles shutdown. After shutdown, the system can be powered up again by driving GPIO3 low.

### Steps

#### Wiring Push-Button

The push-button is connected to GPIO3 (=SCL, physical pin #05) and GND (=GND, physical pin #06).

#### Add dtoverlay to config.txt

```
ssh root@player-ip
# Set readwrite
mount -o remount,rw /flash
# Edit config.txt
nano /flash/config.txt
#Add line:
dtoverlay= gpio-shutdown,active_low=1, gpio_pull=up
Set readonly:
#mount -o remount,ro /flash
```

#### Set key map to avoid Kodi Shutdown Menu

```
nano /storage/.kodi/userdata/keymaps/keyboard.xml:
# Add
<keymap>
  <global>
    <keyboard>
      <power>shutdown</power>
    </keyboard>
  </global>
</keymap>
```

#### Reboot

```
shutdown -r 0
```

#### Check bootlog

This is to ensure the dtoverlay is defined properly

```
vcdbg log msg
```

## RPi Addition Status LED

In addition to the push-button, a RED LED indicates the status of the Raspberry Pi running Kodi. LED on = Kodi is running.

The LED is turned on during boot and turned off during shutdown.

### Steps

#### Wiring LED

The LED RED is connected to GPIO13 (=physical pin #33) and GND (=physical pin #34).

#### Status LED Python Script

Requires the Kodi Addon "Script - Raspberry Pi Tools"

(installed via GUI Addons > Search > Raspberry > select ""Script - Raspberry Pi Tools" > install).

The addon is installed in folder:

```
/storage/.kodi/addons/virtual.rpi-tools/
```

Python Script Location:

```
/storage/scripts/status_led.py
```

Python Script:

```
import sys
sys.path.append('/storage/.kodi/addons/virtual.rpi-tools/lib')
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
led_gpio_number = 13

# Use BCM pin numbering (i.e. the GPIO number, not pin number).
GPIO.setmode(GPIO.BCM)

# Switch LED on.
# LED off at shutdown by using GPIO state.
GPIO.setup(led_gpio_number, GPIO.OUT)

state = True
# state = False
GPIO.output(led_gpio_number, state)
```

#### Autostart Bash Script

Bash Script Location:

```
/storage/.config/autostart.sh
```

Bash Script:

```
#!/bin/sh
## Status led RED connected to gpio13
python /storage/scripts/status_led.py
```

#### Make Scripts Executable

```
chmod +x /storage/scripts/status_led.py
chmod +x /storage/.config/autostart.sh
```

# MQTT

## Purpose

To enable subscribing and publishing MQTT messages ([see Wiki](#)).

Ensure also to lookup the reference [Domoticz API/JSON Url's](#).

## Install

### Raspberry Pi

Install [mosquitto](#) on the Raspberry Pi.

Open Terminal and run

```
sudo apt-get install mosquitto mosquitto-clients
```

Check if mosquitto is running:

```
mosquitto -h
```

### Output

```
mosquitto version 1.4.10 (build date Fri, 22 Dec 2017 08:19:25 +0000)
mosquitto is an MQTT v3.1 broker.
```

### Hint

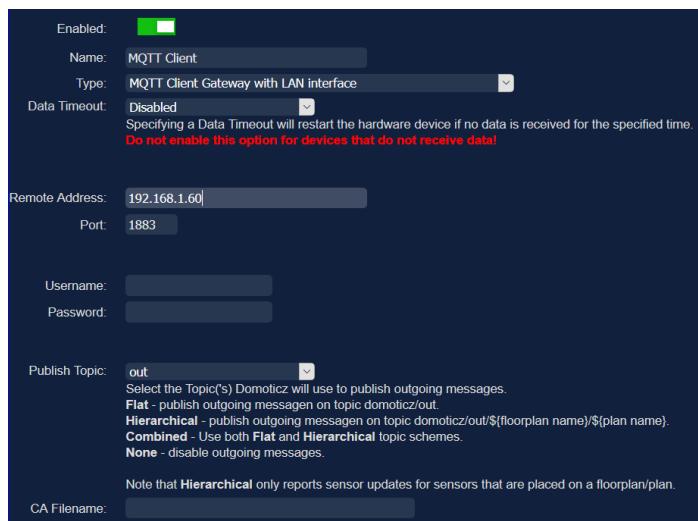
The MQTT parameter to add the host to the mosquitto command is

```
-h localhost
```

## Domoticz

Domoticz Add Hardware: Domoticz GUI > Setup > Hardware and add the “MQTT Client Gateway with LAN Interface”.

Property	Value
Type	MQTT Client Gateway with LAN Interface
Name	MQTT
Remote Address	domoticz-ip
Port	1883
Publish Topic	Out
Username	Empty
Password	Empty



Check the Domoticz Log: Domoticz GUI > Setup > Log

```
2018-08-27 12:21:12.126 Status: MQTT: Connecting to domoticz-ip:1883
2018-08-27 12:21:12.228 Status: MQTT: connected to: domoticz-ip:1883
2018-08-27 12:21:12.328 Status: MQTT: Subscribed
```

## Domoticz Topics

Subscribe to Topics from Domoticz	domoticz/out
Publish a Topic to Domoticz	domoticz/in

# Mosquitto

Test MQTT messaging using mosquitto commands from a terminal.

## Subscribe

### Command

Subscribe to all MQTT messages published by Domoticz (topic: domoticz/out):

```
mosquitto_sub -t 'domoticz/out'
```

### Sample Messages

Some messages published by Domoticz in JSON format.

The source is from the devices provided by the Hardware Device Motherboard (IDX=1 and 2):

□	2	Motherboard	0001	1	Internal Temperature	Temp	LaCrosse TX3	42.4 C
□	1	Motherboard	0000044D	1	CPU_Usage	General	Percentage	0.19%

For each of the messages the topic is *domoticz/out*, and the payload is a JSON string.  
To get a value from a key, the JSON string needs to be parsed.

### CPU Usage

```
{  
    "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "General", "id" : "0000044D",  
    "idx" : 1, "name" : "CPU_Usage", "nvalue" : 0, "stype" : "Percentage", "svalue1" : "0.19",  
    "unit" : 1  
}
```

### CPU Temperature

```
{  
    "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "Temp", "id" : "1",  
    "idx" : 2, "name" : "Internal Temperature", "nvalue" : 0, "stype" : "LaCrosse TX3",  
    "svalue1" : "42.4", "unit" : 1  
}
```

## Publish

Let us change the text for a Virtual Device Type General, Text with Idx 9, Name “Biotonne Pi” from “Hello World” to “Moin Moin” (a typical greeting word in northern part of Germany).

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
9	Virtual Devices	00082009	1	Biotonne Pi	General	Text	Hello World

## Command

```
mosquitto_pub -t 'domoticz/in' -m '{"idx":9,"nvalue":0,"svalue":"Moin Moin"}'
```

There is no terminal output from this command.

Possible checks if the command was successful:

Check Domoticz Widget has changed (Domoticz GUI Tab Utility)	
Check the Domoticz Log (Domoticz GUI Tab Setup > Log)	2018-08-27 15:57:26.792 MQTT: Topic: domoticz/in, Message: {"idx":9,"nvalue":0,"svalue":"Moin Moin"}

# Python

To be able to use MQTT from python, the paho client library is required.

## Paho Client Library

To install, download the client and run the Python setup script to install the client.

There are two ways:

- cloning from github, which will install the client with source code and examples
- using pip and pip3

## GitHub

### Command & Output

```
mkdir /home/pi/python
git clone https://github.com/eclipse/paho.mqtt.python.git
cd paho.mqtt.python
sudo python setup.py install
**Output**
mkdir python
git clone https://github.com/eclipse/paho.mqtt.python.git
Cloning into 'paho.mqtt.python'...
remote: Counting objects: 2932, done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 2932 (delta 27), reused 85 (delta 24), pack-reused 2834
Receiving objects: 100% (2932/2932), 835.76 KiB | 447.00 KiB/s, done.
Resolving deltas: 100% (1487/1487), done.
cd paho.mqtt.python
sudo python setup.py install
running install
...
Installed /usr/local/lib/python2.7/dist-packages/paho_mqtt-1.3.1-py2.7.egg
Processing dependencies for paho-mqtt==1.3.1
Finished processing dependencies for paho-mqtt==1.3.1
```

## Pip (Python2) & Pip3 (Python3)

### Pip Command & Output

```
pip install paho-mqtt
**Output**
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://files.pythonhosted.org/packages/59/11/1dd5c70f0f27a88a3a05772cd95f6087ac479fac66d9c7752ee5e16dd
bbc/paho-mqtt-1.5.0.tar.gz (99kB)
Building wheels for collected packages: paho-mqtt
  Running setup.py bdist_wheel for paho-mqtt ... done
  Stored in directory:
/home/pi/.cache/pip/wheels/02/94/6c/8474137cb7a5a3e001d70a22c8ff919caee69435376bccce79
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.0
```

### Pip3 Command & Output

```
pip3 install paho-mqtt
**Output**
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-1.5.0-py3-none-any.whl (61kB)
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.0
```

## Python Subscribe Script

This Python sample script connects to the MQTT broker running on the Raspberry Pi Domoticz System, subscribes to the Domoticz Out topic (“domoticz/out”) and filters the value for the device with Idx 39 and display the properties name & svalue ... plus additional debug information.

### Domoticz Device

Idx=39, Hardware=Raspberry Pi, Type=General, SubType=Percentage
---

### Python3 Script

Source code: mqtt\_sub\_idx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""

File: mqtt_sub_idx.py
Project: Domoticz-Home-Automation-Workbook
Purpose: Subscribe to MQTT messages and filter device with Idx=39
Run: python3 /home/pi/domoticz/scripts/python/mqtt_sub_idx.py
Version: 20200227 by rwbl
"""

import time
import paho.mqtt.client as mqtt
import json
BROKER = "localhost"
TOPIC = "domoticz/out"
IDX = 39

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(TOPIC)

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
    json_string = str(message.payload.decode("utf-8"))
    getidxvalue(json_string, IDX)

def on_log(client, userdata, level, buf):
    print("log: ",buf)

# JSON Parsing
def getidxvalue(json_string, idx):
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        parsed_name = str(parsed_json['name'])
        parsed_value = str(parsed_json['svalue1'])
        print(parsed_idx, parsed_name, parsed_value)

# Main
if __name__ == "__main__":
    print("Creating new instance")
    client = mqtt.Client("P1")
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_log = on_log
    print("Connecting to the broker", BROKER)
    client.connect(BROKER, 1883, 60)
    time.sleep(4)
    client.loop_forever()
```

## Output Sample with lots of prints to show the steps & actions

```
python3 mqtt_sub_ix.py
creating new instance
connecting to broker
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
log: Received CONNACK (0, 0)
Connected with result code 0
log: Sending SUBSCRIBE (d0) [(b'domoticz/out', 0)]
log: Received SUBACK
log: Received PUBLISH (d0, q0, r0, m0), 'domoticz/out', ... (229 bytes)
message received {
    "Battery" : 255,
    "RSSI" : 12,
    "description" : "",
    "dtype" : "General",
    "id" : "0000044D",
    "idx" : 39,
    "name" : "CPU_Usage",
    "nvalue" : 0,
    "stype" : "Percentage",
    "svalue1" : "0.77",
    "unit" : 1
}

message topic= domoticz/out
message qos= 0
message retain flag= 0
39 CPU_Usage 0.77

message topic= domoticz/out
message qos= 0
message retain flag= 0
1 CPU_Usage 0.33
^CTraceback (most recent call last):
  File "/home/pi/.local/lib/python3.5/site-packages/paho/mqtt/client.py", line 1481, in loop_forever
    rc = self.loop(timeout, max_packets)
  File "/home/pi/.local/lib/python3.5/site-packages/paho/mqtt/client.py", line 988, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt
```

## Python Publish Script

This sample script connects to the MQTT broker running on the Raspberry Pi Domoticz System and publishes text to a Virtual Device.

The Python script sets the text of the device to i.e. “Hello World at 2020-02-27 19:35:39”.

### Domoticz Device

```
Idx=108, Hardware=VirtualDevices, Type General, SubType Text
```

### Python3 Script

Source code: mqtt\_pub\_idx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""

File: mqtt_sub_idx.py
Project: Domoticz-Home-Automation-Workbook
Purpose: Publish a value for a Domoticz text device with idx=108
Run: python3 /home/pi/domoticz/scripts/python/mqtt_pub_idx.py
Result: Updated Domoticz text device with text i.e. Hello World at 2020-02-27 19:35:39
Version: 20200227 by rwbl
"""

import time
from datetime import datetime
import paho.mqtt.client as mqtt

BROKER = "localhost"
TOPIC = "domoticz/in"
IDX = 108
payload = '{"command": "udevice", "idx": %s, "svalue": "Hello World at %s"}' % (IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))

# MQTT Callback Functions
def on_connect(client, userdata, flags, rc):
    print("on_connect: Client={},userdata={},flags={},rc={}".format(client, userdata, flags, rc))

def on_message(client, userdata, message):
    print("on_message: topic={}".format(message.topic))
    print("on_message: payload={}".format(message.payload.decode("utf-8")))
    print("on_message: qos={},retain=".format(message.qos,message.retain))

def on_log(client, userdata, level, buf):
    print("on_log:client={},userdata={},level=={},buf={}".format(client, userdata, level, buf))

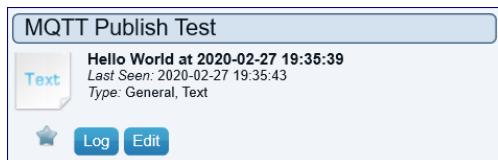
# Main
if __name__ == "__main__":
    print("Creating new instance")
    client = mqtt.Client("P1")
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_log = on_log
    print("Connecting to the broker={}".format(BROKER))
    client.connect(BROKER, 1883, 60)
    time.sleep(4)
    print("Publishing message to topic={},payload={}".format(TOPIC,payload))
    client.publish(TOPIC,payload)
    time.sleep(4)
    client.loop_stop()
    print("Disconnected from the broker={}".format(BROKER))
```

### Output

```
python3 mqtt_pub_idx.py
Creating new instance
Connecting to the broker=localhost
```

```
on_log:client=<paho.mqtt.client.Client object at 0x766d4b70>,userdata=None,level==16,buf=Sending
CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
Publishing message to topic=domoticz/in,payload=
on_log:client=<paho.mqtt.client.Client object at 0x766d4b70>,userdata=None,level==16,buf=Sending
PUBLISH (d0, q0, r0, m1), 'b'domoticz/in'', ... (82 bytes)
Disconnected from the broker=localhost
```

## Widget with updated text



# Node-RED

## Purpose

To explore Node-RED functionality for the Domoticz Home Automation system:

- Node-RED as alternative script engine, i.e. perform tasks, i.e. MQTT subscribe or publish
- Node-RED UI Dashboard Add-On as a simple monitor using Gauges & Charts or to control using buttons / slider / input-fields
- Trigger notifications or as watchdog for Domoticz processes

[Node-RED](#) = Low-code programming for event-driven applications.

If not familiar with the Node-RED concept, strongly recommend reading the Node-RED [get started](#) documentation.

Several Functions make use of Node-RED, like [Volumio](#) or [Email Control](#) and many other.

Node-RED runs on the same Raspberry Pi as the Domoticz Production & Development systems running the Raspberry Pi OS.

In addition, made several tests with Node-RED under Ubuntu running on a notebook.

BTW: Programs in Node-RED are called flows.

## Install & Update

### Installation & Update Command Linux ([Info](#))

From the CLI run :

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

#### Important

Prior updating Node-RED, backup the flows file, located in

```
/home/pi/.node-red/
```

The flows are named depending on the Raspberry Pi device name, i.e.

```
flows_DoPro.json  
flows_DoDev.json
```

After updating Node-RED, reboot the Raspberry Pi.

### Update Log Example (Raspberry Pi)

```
This script will remove versions of Node.js prior to version 7.x, and Node-RED and if necessary replace them with Node.js 10.x LTS (dubnium) and the latest Node-RED from Npm.
```

It also moves any Node-RED nodes that are globally installed into your user `~/.node-red/node_modules` directory, and adds them to your `package.json`, so that you can manage them with the palette manager.

It also tries to run '`npm rebuild`' to refresh any extra nodes you have installed that may have a native binary component. While this normally works ok, you need to check that it succeeds for your combination of installed nodes.

To do all this it runs commands as root - please satisfy yourself that this will not damage your Pi, or otherwise compromise your configuration. If in doubt please backup your SD card first.

Are you sure you want to do this ? [y/N] ? y

Would you like to install the Pi-specific nodes ? [y/N] ? y

Running Node-RED install for user pi at /home/pi on raspbian

This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED	✓
Remove old version of Node-RED	✓
Using N to manage Node.js	+
Update Node.js LTS	✓ Node v12.14.0 Npm 6.13.4
Clean npm cache	✓
Install Node-RED core	✓ 1.0.3
Move global nodes to local	-
Install extra Pi nodes	✓
Npm rebuild existing nodes	-
Add shortcut commands	✓
Update systemd script	✓

Any errors will be logged to `/var/log/nodered-install.log`

All done.

You can now start Node-RED with the command `node-red-start` or using the icon under Menu / Programming / Node-RED

Then point your browser to `localhost:1880` or `http://{your_pi_ip}:1880`

Started Sat 21 Dec 10:10:17 CET 2019 - Finished Sat 21 Dec 10:12:28 CET 2019

After installation, a new Desktop Menu Entry is available:

Programming > Node-RED

To enable autostart of Node-RED at boot, run the terminal command:

`sudo systemctl enable nodered.service`

Output

```
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service →
/lib/systemd/system/nodered.service.
```

To go for sure if Node-RED is started during boot, reboot the Raspberry Pi:

`sudo shutdown -r now`

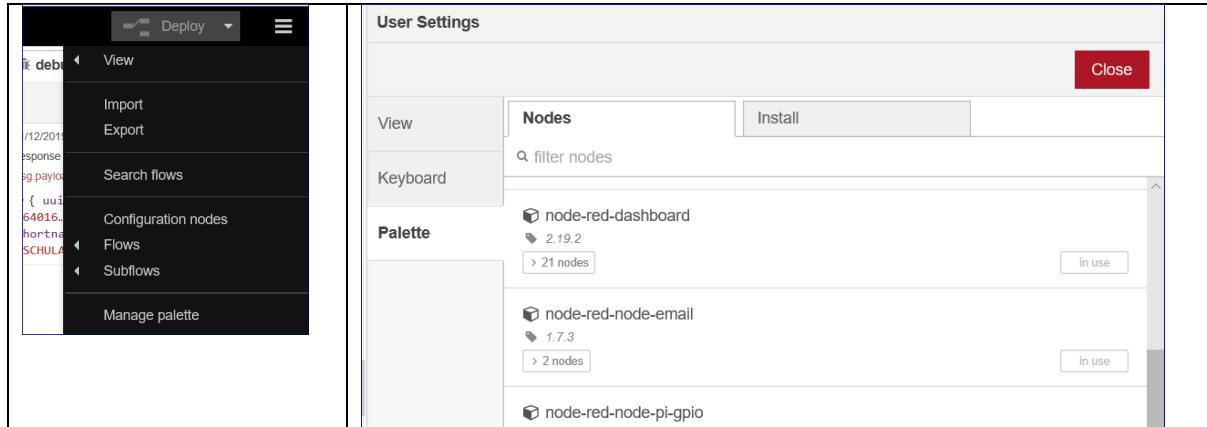
and access Node-RED (`http://domoticz-ip:1880`).

Use `disable`, to disable starting Node-RED at autostart:

`sudo systemctl disable nodered.service`

## Install Add-Ons

It is strongly recommended to install Node-RED add-ons, via the Node-RED menu “Manage Palette”.



## Access Flows & Dashboard UI

To access Node-RED, point a browser at URL:

```
http://localhost:1880 (when using a Raspberry Pi with a connected monitor)
```

or

```
http://domoticz-ip:1880 (remote)
```

This enables to manage Node-RED, define & run flows etc.

If the Dashboard add-on is installed, access the UI with its tabs via URL:

```
http://domoticz-ip:1880/ui
```

## Start, Stop, Log

Task	Terminal Command
Node-RED Start	node-red-start
Node-RED Stop	node-red-stop
Node-RED view recent log output	node-red-log

## Manage Node Packages

npm

To manage node packages in Node-RED, the npm package is required.  
Check if installed:

```
npm -v
```

If package not found, then install via Desktop > Preferences > Add / Remove Software and restart Node-RED (terminal commands node-red-stop, node-red-start)

Check npm version:

```
npm -v
Output
6.4.1
```

## Install/Updates

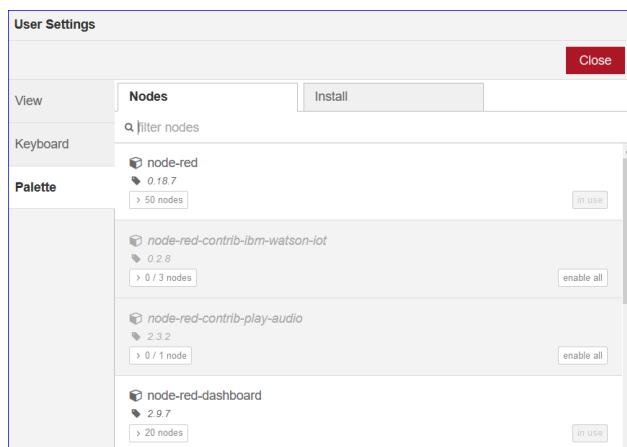
### Important

To install or update any node package, Node-RED recommends via *Node-RED Menu Settings > Manage Palette > Install*.

### Note

If direct installed via npm, Node-RED might be missing dependencies.

After install/update, restart Node-RED (terminal commands node-red-stop, node-red-start). It is not recommended to install using terminal commands.



## Folder Locations

Data	Folder
Modules	/home/pi/.node-red
Flows	/home/pi/.node-red/ flows_dodev.json
Additional Node Packages	/home/pi/.node-red/node_modules/<package name>
Additional Node Packages: Sample Dashboard Nodes	/home/pi/.node-red/node_modules/node-red-dashboard

## Example Flows

### Log Domoticz MQTT Messages

A simple flow, to test Node-RED receiving data from Domoticz and get a key:value.

#### Node-RED Flow



#### Node mqtt in

Connects to the MQTT broker and subscribes to the MQTT topic “domoticz/out” to listen to all incoming Domoticz MQTT messages.

The MQTT Broker is localhost with default port 1883 because Node-RED and Domoticz are running on the same hardware (Raspberry Pi with Raspian Buster).

The output of the node is a parsed JSON object.

#### Node-RED Flow Source

Source: mqtt\_listener\_example.flow

```

[{"id": "fe01be83.5c17d", "type": "mqtt in", "z": "41e8885e.6f1a68", "name": "", "topic": "domoticz/out", "qos": "2", "datatype": "json", "broker": "de20bce4.03135", "x": 110, "y": 60, "wires": [[{"id": "739fc80e.4b63f8"}]}, {"id": "739fc80e.4b63f8", "type": "debug", "z": "41e8885e.6f1a68", "name": "DEBUG MQTTDOPRO", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "targetType": "msg", "x": 350, "y": 60, "wires": []}, {"id": "de20bce4.03135", "type": "mqtt-broker", "z": "", "name": "MQTT Broker DoPro", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": false, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}], ]
  
```

#### Output debug

info	debug	dashboard
		<b>Payload example (Hue Light with IDX=118)</b>
		{ "Battery":255, "Level":90, "RSSI":12, "description:"", "dtype":"Light/Switch", "id":"00000005", "idx":118, "name":"Hue MakeLab", "nvalue":2, "stype":"Switch", "svalue1":"90", "switchType": "Dimmer", "unit":1}
		<b>Access a key (case sensitive), convert the Domoticz message to a JSON object:</b>
		msg.payload.idx returns 118
		msg.payload.Level returns 90

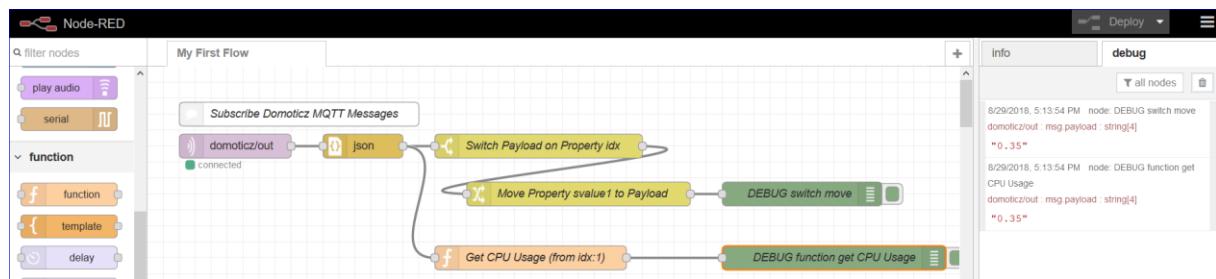
## Log Raspberry Pi CPU Usage

Subscribe to MQTT topic “Domoticz/in” to listen to all Domoticz MQTT messages. From the message payload (after conversion to JSON), select the property idx with value 1. Then select the property svalue1 to get the CPU Usage.

### Domoticz Topic “Domoticz/in” Message Payload for idx = 1

```
{"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"0000044D","idx":1,"name":"CPU_Usage","nvalue":0,"stype":"Percentage","svalue1":"0.28","unit":1}
```

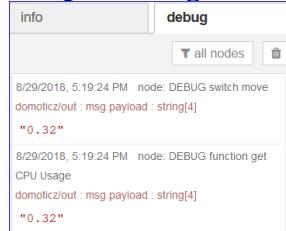
### Node-RED Flow



### Node-RED Flow Source

```
[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 100, "wires": [[[ "db5fff92.295628"]]]}, {"id": "35214a2f.f89656", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG switch move", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 820, "y": 160, "wires": []}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 100, "wires": [[[ "5f7a4a09.abb57c", "13e0d06f.2b15d"]]]}, {"id": "5f7a4a09.abb57c", "type": "switch", "z": "76cb1798.6a823", "name": "Switch Payload on Property idx", "property": "payload.idx", "propertyType": "msg", "rules": [{"t": "eq", "v": "1", "vt": "num"}], "checkall": "true", "repair": false, "outputs": 1, "x": 490, "y": 100, "wires": [[[ "912289d2.e8c1"]]]}, {"id": "13e0d06f.2b15d", "type": "function", "z": "76cb1798.6a823", "name": "Get CPU Usage (from idx:5)", "func": "// Get the property idx from the payload\nidx = msg.payload.idx;\ncpu_usage=-1;\n// Check for idx 1\nif (idx == 5) {\n    // Get the cpu usage from property svalue1\n    cpu_usage = msg.payload.svalue1;\n    // Assign the cpu usage to the message payload\n    msg.payload = cpu_usage;\n}\n// Return the message\nreturn msg;\n"}, {"id": "c030082a.286338", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG function get CPU Usage", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 240, "wires": []}, {"id": "acdab92.b0151c8", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG idx=1", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 800, "y": 100, "wires": []}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}]
```

## Output debug



The screenshot shows the Node-RED interface with the 'debug' tab selected. It displays two log entries:

```

8/29/2018, 5:19:24 PM node: DEBUG switch move
domoticz/out : msg payload : string[4]
"0..32"
8/29/2018, 5:19:24 PM node: DEBUG function get
CPU Usage
domoticz/out : msg payload : string[4]
"0..32"

```

## Node-RED as MQTT Publisher

Instead of defining MQTT topics, a Node-RED flow is an alternative to publish MQTT topics with payload.

For an example, see Dashboard UI > Flow Virtual Sensor.

## Domoticz as Node-RED Data Logger

Log data from Node-RED and update Domoticz devices to access the charts for the devices.

This example is using the data provided by [WASC](#), a tailor made Home Automation & Information System with a WebUI for use with smartphones.

WASC – Web-App Site Control created by the author - is an experimental solution based on Node-RED.

### Concept

(Provider) The Node-RED flow

- builds a JSON data object from several WASC functions, i.e. weather, energy
- updates the JSON data object by subscribing to MQTT messages from the functions
- listens to HTTP GET requests and returns the JSON data object

## HTTP GET Request to Node-RED WASC

```
http://node-red-ip:1880/wasc
```

## HTTP Response

```
{
  "weather": {
    "temperature":12.3, "humidity":79, "windspeed":5, "windgust":12, "winddirection":"WSW",
    "windbearing":247, "airpressure":999,"timestamp":"09:32"
  },
  "energy": {
    "house":140,"makelab":0,"timestamp":"09:33"
  }
}
```

(Requestor) The HTTP response JSON data object is parsed to get the key values.

In Domoticz, an automation event using a dzVents Lua script places a GET request (function domoticz.openURL) in regular intervals and parses from the HTTP response, the key values to update the Domoticz devices.

The HTTP response item data is converted to a Lua table (example weather data wind):

```
-- HTTP response main keys
local KEY_WEATHER = 'weather';
-- Convert selected key into Lua table
local weatherData = item.json[KEY_WEATHER];
```

The variable “weatherData” contains a Lua table:

```
{
  ["timestamp"] = "16:24", ["temperature"] = 15.7, ["windgust"] = 6, ["humidity"] = 70, ["winddirection"] = "N",
  ["windspeed"] = 7, ["airpressure"] = 1000, ["windbearing"] = 0
}
```

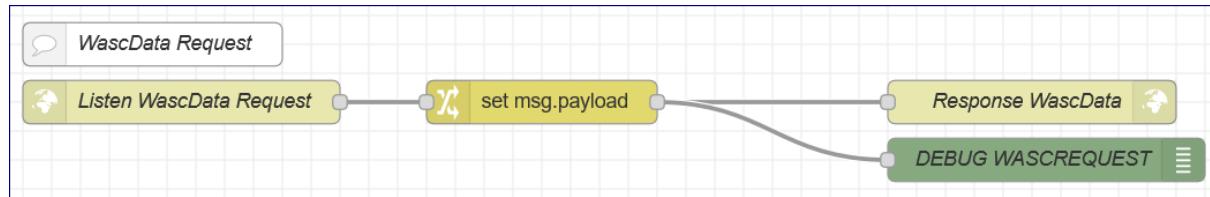
To access keys and update the Domoticz device, use

```
Variable.key, i.e. weatherData.windbearing
```

```
-- 117,VirtualSensors,Windstärke,Wind,WTGR800,203;SSW;10;21.0;14.0;14.0
local IDX_WIND = 117;

domoticz.devices(IDX_WIND).updateWind(weatherData.windbearing, weatherData.winddirection,
weatherData.windspeed / 3.6, weatherData.windgust / 3.6, weatherData.temperature,
weatherData.temperature);
```

## Node-Red Flow



The flow is an extract of the WASC-Interface flow.

### Node http in

This is a HTTP end-point for the web service listening to a GET request (method: GET, URL: /wasc).

The GET request uses the IP address of the system running Node-RED (which is the same as the Domoticz Production system in the authors setup on the Raspberry Pi running Raspian Buster).

```
http://node-red-ip:1880/wasc
```

### Node change

The msg.payload is set with the JSON object from the flow context “wascdatadata”.

The JSON object has nested key:value pairs based on the function. In next example, the functions “weather” and “energy” are used:

Example of the flow context “wascdata” JSON object:

```
{
  "weather": {
    {"temperature": -1, "humidity": -1, "windgust": -1, "winddirection": "", "airpressure": -1, "timestamp": ""},
    "energy": {
      {"house": 210, "makelab": 0, "timestamp": "09:28"}
    }
}
```

### Node http response

Sends the response back to request received from the HTTP Input node.  
The msg.payload is the wascdata as JSON formatted string – as shared previous.

### Domoticz

See the concept earlier.

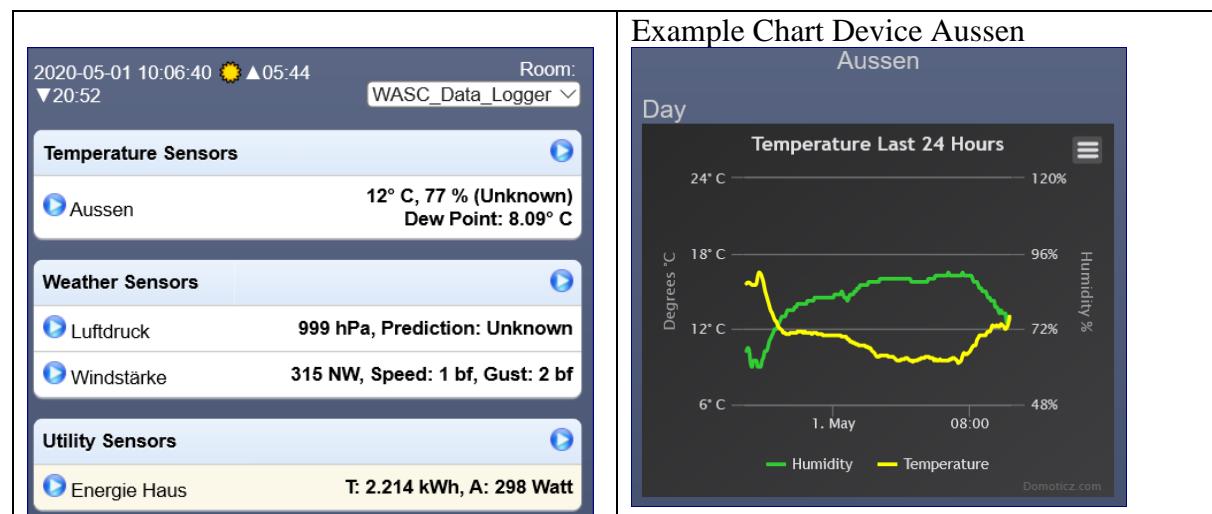
Create several Devices (Hardware Dummy Virtual Sensor):

Idx, Hardware, Name, Type, SubType, SwitchType, Data
117, VirtualSensors, Windstärke, Wind, WTGR800,203;SSW;10;21.0;14.0;14.0
116, VirtualSensors, Luftdruck, General, Barometer, 999 hPa
300, VirtualSensors, Außentemperatur, Temp + Humidity, THGN122/123/132, THGR122/228/238/268, 0.0 C, 50 %
171, VirtualSensors, Energie Haus, General, kWh, 2250.074 kWh

### Domoticz Dashboard

Roomplan: WASC\_Data\_Logger with the devices Idx 117,116,30,171.

Settings UI: Language: English, Theme: simple-gray, Dashboard: Mobile, Mobile: Mobile



The devices are updated via the automation event dzVents Lua script:

Event name: wasc\_data\_logger

```

local URL_DOMOTICZ = 'http://domoticz-ip:8080'
local URL_WASC_DATA = 'http://node-red-p:1880/wasc';
local RES_WASC_DATA = 'RES_WASC_DATA';
local KEY_WEATHER = 'weather';
local KEY_ENERGY = 'energy';
local IDX_WIND = 117;
local IDX_AIRPRESSURE = 116;
local IDX_OUTDOORTEMPUM = 300;
local IDX_ELECTRICUSAGEHOUSE = 171 -- Type=General, SubType=kWh
return {
    on = {timer = { 'every minute' }, httpResponses = { RES_WASC_DATA, }},
    execute = function(domoticz, item)
        if (item.isTimer) then
            domoticz.openURL({url = URL_WASC_DATA, method = 'GET', callback = RES_WASC_DATA,})
        end
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                local weatherData = item.json[KEY_WEATHER];
                domoticz.devices(IDX_WIND).updateWind(weatherData.windbearing, weatherData.winddirection,
                weatherData.windspeed / 3.6, weatherData.windgust / 3.6, weatherData.temperature,
                weatherData.temperature);
                local status = -1; -- not used
                domoticz.devices(IDX_OUTDOORTEMPUM).updateTempHum(weatherData.temperature,
                weatherData.humidity, status);
                local forecast = -1; -- not used
                domoticz.devices(IDX_AIRPRESSURE).updateBarometer(weatherData.airpressure, forecast)
                local energyData = item.json[KEY_ENERGY];
                local power = math.floor(energyData.house);
                domoticz.openURL({url = string.format(
                    '%s/json.htm?type=command&param=udevice&idx=%d&nvalue=0&svalue=%d;0', URL_DOMOTICZ,
                    IDX_ELECTRICUSAGEHOUSE, power), method = 'GET'})
                return
            else
                domoticz.log('[ERROR]: ' .. item.statusText, domoticz.LOG_ERROR)
                return
            end
        end
    end
}
}

```

## Domoticz Log

```

2020-05-01 10:00:03.726 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-01 10:01:00.529 Status: dzVents: Info: ----- Start internal script: wasc_data_logger:, trigger: "every minute"
2020-05-01 10:01:00.529 Status: dzVents: Info: ----- Finished wasc_data_logger
2020-05-01 10:01:00.530 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-01 10:01:00.639 Status: dzVents: Info: Handling httpResponse-events for: "RES_WASC_DATA"
2020-05-01 10:01:00.640 Status: dzVents: Info: ----- Start internal script: wasc_data_logger: HTTPResponse: "RES_WASC_DATA"
2020-05-01 10:01:00.640 Status: dzVents: Info: Status Code = 200, Name=RES_WASC_DATA, Data={"weather":{"temperature":12.2,"humidity":77,"windspeed":3,"windgust":5,"winddirection":"WSW","windbearing":247,"airpressure":999,"timestamp":"09:57"},"energy":{"house":242,"makelab":0,"timestamp":"09:57"}}
2020-05-01 10:01:00.655 Status: dzVents: Info: Windstärke: 0.8, 12.2
2020-05-01 10:01:00.656 Status: dzVents: Info: Aussen: 12.2, 77.0
2020-05-01 10:01:00.657 Status: dzVents: Info: Luftdruck: 999
2020-05-01 10:01:00.658 Status: dzVents: Info: Energie Haus: Wh = Actual: 242, Today: 2188, Total: 2255916
2020-05-01 10:01:00.658 Status: dzVents: Info: ----- Finished wasc_data_logger

```

## Example Dashboard UI Flows

The Node-RED [Dashboard](#) module offers UI nodes to build a Dashboard.

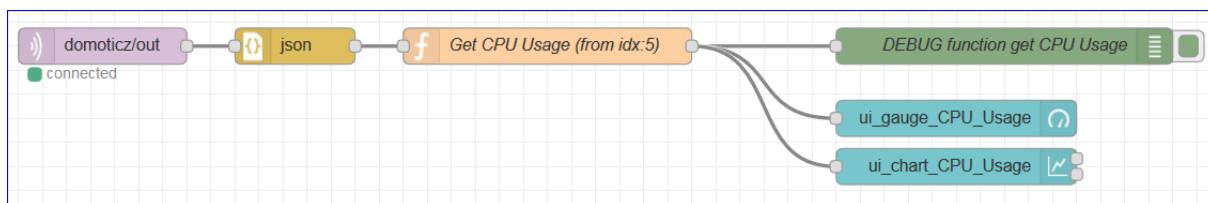
### Monitor Raspberry Pi CPU Usage

Build a simple example of a dashboard using the Raspberry Pi CPU Usage information, which is displayed in a Gauge and Line Chart.

#### Dashboard



#### Node-RED Flow



The Function Node selects idx 5 and builds message payload holding the CPU Usage in %.

```
// Get the property idx from the payload
idx = msg.payload.idx;
cpu_usage=-1;
// Check for idx 1
if (idx == 5) {
    // Get the cpu usage from property svalue1
    cpu_usage = msg.payload.svalue1;
    // node.warn("Idx=1, CPU Usage=" + cpu_usage);
    // Assign the cpu usage to the message payload
    msg.payload = cpu_usage;
    // Return the message
    return msg;
}
```

## Node-RED Flow Source

```
[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 240, "wires": [[{"id": "db5fff92.295628"}]]}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 240, "wires": [[{"id": "13e0d06f.2b15d"}]]}, {"id": "13e0d06f.2b15d", "type": "function", "z": "76cb1798.6a823", "name": "Get CPU Usage (from idx:5)", "func": "// Get the property idx from the payload\nidx = msg.payload.idx;\nncpu_usage=-1;\n//\nCheck for idx 1\nif (idx == 5) {\n    // Get the cpu usage from property svalue1\n    cpu_usage = msg.payload.svalue1;\n    // node.warn(\"Idx=1, CPU Usage=\") + cpu_usage);\n    // Assign the cpu usage to the message payload\n    msg.payload = cpu_usage;\n    // Return the message\n    return msg;\n}\n", "outputs": 1, "noerr": 0, "x": 480, "y": 240, "wires": [{"id": "c030082a.286338", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG function get CPU Usage", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 240, "wires": []}], {"id": "c030082a.286338", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "gtype": "gage", "title": "", "label": "%", "form": "at": "{{value}}", "min": 0, "max": 100, "colors": ["#00b500", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "x": 820, "y": 300, "wires": []}, {"id": "d5bc4ce8.ae1f28", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "label": "", "chartType": "line", "legend": false, "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": true, "ymin": 0, "ymax": 100, "removeOlder": 1, "removeOlderPoints": "", "removeOlderUnit": "3600", "cutout": 0, "useOneColor": false, "colors": ["#1f77b4", "#aec7e8", "#ff7f0e", "#2ca02c", "#98df8a", "#d62728", "#ff9896", "#9467bd", "#c5b0d5"], "useOldStyle": false, "x": 820, "y": 340, "wires": []}, {"id": "d58d713d.0e0eb8", "type": "mqtt_broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}], {"id": "c3b6865e.2b93e", "type": "ui_group", "z": "", "name": "CPU", "tab": "8a1743ab.9d931", "disp": true, "width": 6, "collapse": false}, {"id": "8a1743ab.9d931", "type": "ui_tab", "z": "", "name": "Raspberry Pi", "icon": "dashboard"}]
```

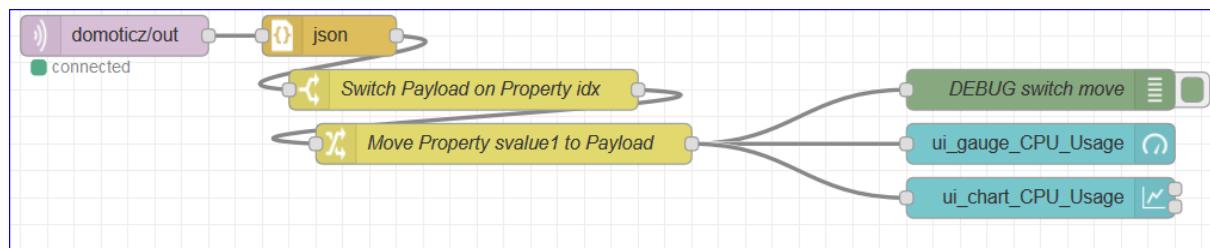
### Note

The charts display each data change which could result in a rather frequent update. To avoid many updates, use a Delay Node which limits the number of messages to for example 12 messages per hour (Properties: "pauseType": "rate", "rate": "12", "nbRateUnits": "1", "rateUnits": "hour").

## Switch Node and Change Node

Instead of using a Function Node, Switch Node and Change Node can be used.

### Node-RED Flow



## Node-RED Flow Source

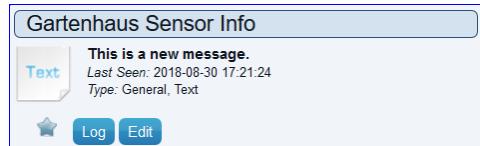
```
[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 100, "wires": [[{"id": "db5fff92.295628"}]]}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 100, "wires": [{"id": "5f7a4a09.abb57c"}]]}, {"id": "5f7a4a09.abb57c", "type": "function", "z": "76cb1798.6a823", "name": "Switch Payload on Property idx", "func": "move", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 800, "y": 140, "wires": []}, {"id": "912289d2.e8c1", "type": "function", "z": "76cb1798.6a823", "name": "Move Property svalue1 to Payload", "func": "payload", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 800, "y": 180, "wires": [{"id": "912289d2.e8c1"}]]}, {"id": "912289d2.e8c1", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG switch move", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 140, "wires": []}, {"id": "912289d2.e8c1", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "76cb1798.6a823", "order": 0, "width": 0, "height": 0, "gtype": "gage", "title": "", "label": "%", "form": "at": "payload", "min": 0, "max": 100, "colors": ["#00b500", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "x": 820, "y": 180, "wires": []}, {"id": "912289d2.e8c1", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "group": "76cb1798.6a823", "order": 0, "width": 0, "height": 0, "label": "", "chartType": "line", "legend": false, "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": true, "ymin": 0, "ymax": 100, "removeOlder": 1, "removeOlderPoints": "", "removeOlderUnit": "3600", "cutout": 0, "useOneColor": false, "colors": ["#1f77b4", "#aec7e8", "#ff7f0e", "#2ca02c", "#98df8a", "#d62728", "#ff9896", "#9467bd", "#c5b0d5"], "useOldStyle": false, "x": 820, "y": 220, "wires": []}, {"id": "76cb1798.6a823", "type": "ui_group", "z": "", "name": "CPU", "tab": "76cb1798.6a823", "disp": true, "width": 6, "collapse": false}, {"id": "76cb1798.6a823", "type": "ui_tab", "z": "", "name": "Domoticz", "icon": "dashboard"}]
```

```
Payload","rules":[{"t":"move","p":"payload.svalue1","pt":"msg","to":"payload","tot":"msg"}],"action":"" , "property": "", "from": "", "to": "", "reg": false, "x": 400, "y": 180, "wires": [{"id": "35214a2f.f89656", "z": "76cb1798.6a823"}, {"id": "61fef7cc.4fc02", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "gtype": "gauge", "title": "", "label": "%", "format": "{{value}}", "min": 0, "max": "100", "colors": ["#00b500", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "x": 800, "y": 180, "wires": []}, {"id": "d5bc4ce8.ae1f28", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "label": "", "chartType": "line", "legend": "false", "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": true, "ymin": "0", "ymax": "100", "removeOlder": 1, "removeOlderPoints": "", "removeOlderUnit": "3600", "cutout": 0, "useOneColor": false, "colors": ["#1f77b4", "#aec7e8", "#ff7f0e", "#2ca02c", "#98df8a", "#d62728", "#ff9896", "#9467bd", "#c5b0d5"], "useOldStyle": false, "x": 800, "y": 220, "wires": [[], []]}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}, {"id": "c3b6865e.2b93e", "type": "ui_group", "z": "", "name": "CPU", "tab": "8a1743ab.9d931", "disp": true, "width": "6", "collapse": false}, {"id": "8a1743ab.9d931", "type": "ui_tab", "z": "", "name": "Raspberry Pi", "icon": "dashboard"}]
```

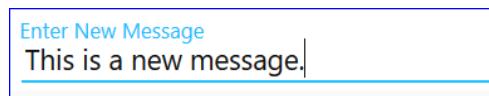
## Virtual Sensor Text Update

Update the Virtual Sensor “Gartenhaus Sensor Info” with new text.  
The device has idx 52, Type General, Text.

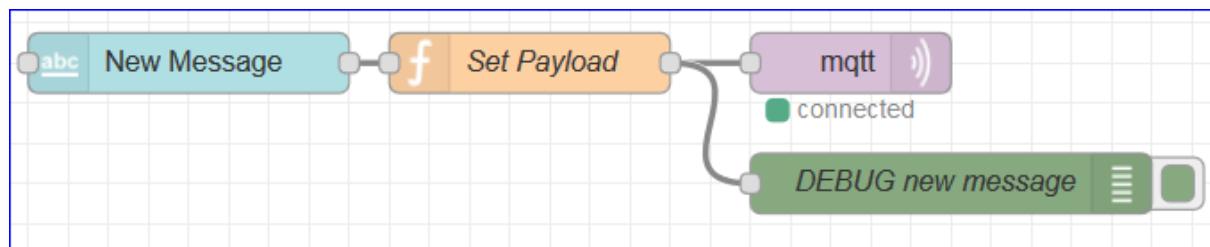
### Widget showing the updated text



### Node-RED Dashboard UI with ui\_text\_input Node



### Node-RED Flow with Nodes ui\_text\_input > function > mqtt out



The text input node has a delay of 0, means the content is sent after pressing enter.  
The function node created the new message which is published via MQTT using the localhost broker.

```

// Build a new message
var msginfo = {};
msginfo.topic = "domoticz/in";

// Get the text from the incoming payload from the text input node.
text = msg.payload;

// Build the new payload
msginfo.payload = { "idx" : 52, "nvalue": 0, "svalue": text};
node.warn(msginfo.payload);

// Return the message
return msginfo;
  
```

The information for building the MQTT payload is taken from the Domoticz API/JSON URL's documentation ([here](#)).

## Node-RED Flow Source

```
[{"id": "38d6f42f.539a94", "type": "ui_text_input", "z": "76cb1798.6a823", "name": "New Message", "label": "Enter New Message", "group": "27ca36a9.4c3cba", "order": 0, "width": 0, "height": 0, "passthru": false, "mode": "text", "delay": "0", "topic": "", "x": 120, "y": 820, "wires": [{"id": "896e27c.ed14dd8"}]}, {"id": "d4be5444.33e23", "type": "mqtt_out", "z": "76cb1798.6a823", "name": "", "topic": "", "qos": "", "retain": "", "broker": "d58d713d.0e0eb8", "x": 450, "y": 820, "wires": []}, {"id": "896e27c.ed14dd8", "type": "function", "z": "76cb1798.6a823", "name": "Set Payload", "func": "// Build a new message\nvar msginfo = {};\nmsginfo.topic = \"domoticz/in\";\n// Get the text from the incoming payload\nvar msg = msg.payload;\n// Build the new payload\nmsginfo.payload = {\n    \"idx\": 52,\n    \"nvalue\": 0,\n    \"svalue\": text\n};\nnode.warn(msginfo.payload);\n// Return the message\nreturn\nmsginfo;"}, {"outputs": 1, "noerr": 0, "x": 290, "y": 820, "wires": [{"id": "d4be5444.33e23", "z": "f167bf33.2066e8"}]}, {"id": "f167bf33.2066e8", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG new message", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "true", "x": 500, "y": 880, "wires": []}, {"id": "27ca36a9.4c3cba", "type": "ui_group", "z": "", "name": "Hue", "tab": "8a1743ab.9d931", "dis": true, "width": 6, "collapse": false}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": 1883, "clientId": "", "useTls": false, "compatmode": true, "keepalive": 60, "cleanSession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "closeTopic": "", "closeQos": 0, "closePayload": "", "willTopic": "", "willQos": 0, "willPayload": ""}], {"id": "8a1743ab.9d931", "type": "ui_tab", "z": "", "name": "Raspberry Pi", "icon": "dashboard"}]
```

# Plugin Development

## Purpose

The Python Plugin Framework allows to create an interface between Hardware (or Virtual Hardware) and Domoticz.

Developed various Python Plugins mainly interacting with [TinkerForge](#) building blocks:

- Soil Moisture Monitor ([Info](#))
- Traffic Light ([Info](#))

Must state, it makes real fun to see how new hardware devices with according devices are created and used.

Mandatory to consult this [manual](#) as reference.

The next chapter describes in depth the build of a sample plugin interacting between Domoticz and Tinkerforge.

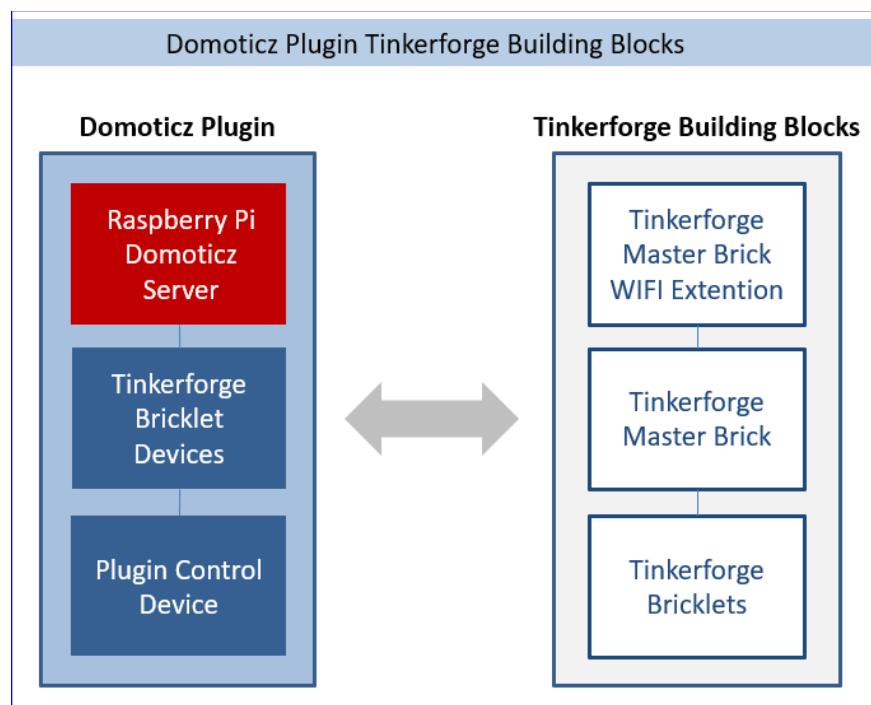
There are other Domoticz Python plugin examples in this document, to mention

- Chapter [RPi.GPIO : Push-button trigger by Python Plugin, LED On|Off via push-button controlled by Python Plugin](#)
- Chapter [Tinkerforge : Domoticz Plugin](#)

The picture shows the setup and interaction between the Domoticz Plugin and the Tinkerforge building blocks.

Each plugin is composed out of

- Domoticz Devices associated with one or more Tinkerforge Bricklets
- Domoticz Text Device (Virtual Sensor) to display plugin state information



## Hints

Some examples & hints on Plugin errors (from file plugin.py) logged the Domoticz Log.

### KeyError

If a plugin starts for the first time and creates new devices in the function onStart, ensure to set the option Setup > Settings > Accept new Hardware (like Allow for 5 minutes). If this option is not set, Domoticz cannot create new devices and logs an error like:

```
Device creation failed, Domoticz settings prevent accepting new devices.
2019-06-04 09:17:25.085 Error: (Soil Moisture Monitor - MakeLab) 'onStart' failed 'KeyError'.

OR
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) Device creation failed, Domoticz settings prevent accepting new devices.
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) 'onStart' failed 'KeyError'.
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) ----> Line 307 in '/home/pi/domoticz/plugins/hmip-etriv/plugin.py', function onStart
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) ----> Line 126 in '/home/pi/domoticz/plugins/hmip-etriv/plugin.py', function onStart
```

### TabError

Ensure to set the tab indents right, else error message in the Domoticz log during start: "TabError".

```
2019-05-24 11:07:17.254 Status: (Air Quality Livingroom) Started.
2019-05-24 11:07:17.424 Error: (AirQualityMonitor) failed to load 'plugin.py', Python Path used was ...
2019-05-24 11:07:17.424 Error: (Air Quality Livingroom) Module Import failed, exception: 'TabError'
2019-05-24 11:07:17.425 Error: (Air Quality Livingroom) Import detail: File:
/home/pi/domoticz/plugins/airqualitymonitor/plugin.py, Line: 165, offset: 92
2019-05-24 11:07:17.425 Error: (Air Quality Livingroom) Error Line ' iaq_index, iaq_index_accuracy,
temperature, humidity, air_pressure = ab.get_all_values()
```

### SyntaxError

```
2019-05-26 11:24:47.046 Error: (TrafficLight) failed to load 'plugin.py', Python Path used was
'/home/pi/domoticz/plugins/TrafficLight/:/usr/lib/python35.zip:/usr/lib/python3.5:/usr/lib/python3.5/plat-arm-linux-gnueabihf:/usr/lib/python3.5/lib-dynload:/usr/local/lib/python3.5/dist-
packages:/usr/lib/python3/dist-packages:/usr/lib/python3.5/dist-packages'.
2019-05-26 11:24:47.046 Error: (Traffic Light Control) Module Import failed, exception: 'SyntaxError'
2019-05-26 11:24:47.046 Error: (Traffic Light Control) Import detail: File:
/home/pi/domoticz/plugins/TrafficLight/plugin.py, Line: 159, offset: 120
2019-05-26 11:24:47.046 Error: (Traffic Light Control) Error Line ' Domoticz.Debug(Devices[2].Name + "-nValue;sValue:" + str(Devices[2].nValue) + ";" + Devices[2].sValue) )
```

### Cause

An additional bracket at the end of the line:

```
Domoticz.Debug(Devices[2].Name + "-nValue;sValue:" + str(Devices[2].nValue) + ";" + Devices[2].sValue)
)
```

### Fix

```
Domoticz.Debug(Devices[2].Name + "-nValue;sValue:" + str(Devices[2].nValue) + ";" + Devices[2].sValue)
```

## Error sValue missing

Example, device “Air Quality” error stating *sValue is missing* used by CDevice\_update

```
2019-06-04 14:58:29.215 Error: (CDevice_update) AQM - IAQ Index: Failed to parse parameters: 'nValue', 'sValue', 'Image', 'SignalLevel', 'BatteryLevel', 'Options', 'TimedOut', 'Name', 'TypeName', 'Type', 'Subtype', 'Switchtype', 'Used', 'Description', 'Color' or 'SuppressTriggers' expected.
2019-06-04 14:58:29.215 Error: (AQM) 'CDevice_update' failed 'TypeError': Required argument 'sValue' (pos 2) not found'.
```

### Cause

Caused by missing sValue parameter for Air Quality update (Devices[1].Update).

```
Devices[1].Update(nValue=iaq_index)
```

### Fix

Add empty sValue parameter.

```
# Update the value - only nValue is used, but mandatory to add an sValue
```

```
Devices[1].Update(nValue=iaq_index, sValue="")
```

## Initial Value New Device

To ensure a new device as an initial value, add update after device create.

Example code snippet from HomematicIP HmIP-eTRV plugin.

```
class BasePlugin:

    def __init__(self):
        ...
        self.SetPoint = 0      # setpoint in C
        self.Temperature = 0   # actual temperature
        self.LowBat = "true"   # low battery "true" or "false"
        self.Level = 0         # valve position 0 - 100%
        ...

    def onStart(self):
        # if there no devices, create these and set initial value
        if (len(Devices) == 0):
            Domoticz.Debug("Creating new devices ...")
            ## SET_POINT_TEMPERATURE - TypeName: Thermostat
            Domoticz.Device(Name="Setpoint", Unit=UNITSETPOINTTEMPERATURE, Type=242, Subtype=1,
Used=1).Create()
            Devices[UNITSETPOINTTEMPERATURE].Update( nValue=1, sValue= str(self.SetPoint) )
            Domoticz.Debug("Device created: "+Devices[UNITSETPOINTTEMPERATURE].Name)

            ## ACTUAL_TEMPERATURE - TypeName: Temperature
            Domoticz.Device(Name="Temperature", Unit=UNITACTUALTEMPERATURE, TypeName="Temperature",
Used=1).Create()
            Devices[UNITACTUALTEMPERATURE].Update( nValue=0, sValue=str(self.Temperature) )
            Domoticz.Debug("Device created: "+Devices[UNITACTUALTEMPERATURE].Name)

            ## LOW_BAT - TypeName: Alert
            Domoticz.Device(Name="Battery", Unit=UNITLOWBAT, TypeName="Alert", Used=1).Create()
            Devices[UNITLOWBAT].Update( nValue=1, sValue=LOWBATMSGOK )
            Domoticz.Debug("Device created: "+Devices[UNITLOWBAT].Name)

            ## LEVEL - TypeName: Percentage
            Domoticz.Device(Name="Valve", Unit=UNITLEVEL, TypeName="Percentage", Used=1).Create()
            Devices[UNITLEVEL].Update( nValue=0, sValue=str(self.Level) )
            Domoticz.Debug("Device created: "+Devices[UNITLEVEL].Name)
        ...

```

## Extract Value XPath

Get a value from an XML tree.

Requires the Python library lxml XML toolkit.

The Python library lxml XML toolkit is used to parse the XML-API response.

### Install

```
sudo apt-get install python3-lxml
```

#### *Note*

If the plugin is added to the Domoticz Hardware without lxml installed, then delete the hardware prior installing lxml. After installing lxml, restart Domoticz: sudo service domoticz.sh restart

### Python Script Import

```
from lxml import etree
```

### Example

Get the value from XML tree entry - taken from function [Radiator Thermostat HmIP-eTRV \(Plugin\)](#).

```
<datapoint type="ACTUAL_TEMPERATURE" ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1577027367" valueunit="" valuetype="4" value="19.600000"/>
```

### XPath Statement

```
actualtemperaturevalue = tree.xpath('//datapoint[@ise_id=1566]/@value')
```

used below with variable.

Returned is an array with a single entry as the datapoint ise\_id is unique, i.e. to get the value use actualtemperaturevalue[0] after ensuring the array length is 1.

```
def onMessage(self, Connection, Data):
    Domoticz.Debug("onMessage called")

    # If not connected, then leave
    if self.httpConnected == 0:
        return

    # Parse the JSON Data Object with keys Status (Number) and Data (ByteArray)
    ## 200 is OK
    responseStatus = int(Data["Status"])
    ## Domoticz.Debug("STATUS="+ str(responseStatus))

    ## decode the data using the encoding as given in the xml response string
    responseData = Data["Data"].decode('ISO-8859-1')
    ## Domoticz.Debug("DATA=" + responseData)

    # Parse the xml string
    # Get the xml tree - requires several conversions
    tree = etree.fromstring(bytes(responseData, encoding='utf-8'))
    ...
    # Get the value for datapoint actual_temperature & update the device and log
    ## note that a list is returned from tree.xpath, but holds only 1 value
    actualtemperaturevalue = tree.xpath('//datapoint[@ise_id=' +
                                         self.DataPointsList[DATAPOINTINDEXACTUALTEMPERATURE] + ']/@value')
```

```
if len(actualtemperaturevalue) == 1:  
    Domoticz.Debug("T Act=" + actualtemperaturevalue[0])  
    ## convert the raspberrymatic value to float  
    atv = float(actualtemperaturevalue[0])
```

## GPIO Pin Usage

Ensure the GPIO pins used are not used by other applications or plugins. Even if a plugin uses the same (or some) of the GPIO pins and the plugin is disabled, the active plugin cannot use the GPIO pins – error setting up GPIO when starting Domoticz, check the log.

## Unique Key & Name

Ensure each of the plugins have a unique key across all plugins:

```
<plugin key="UNIQUE KEY" name="UNIQUENAME"
```

## Development Iteration

During development, restart Domoticz after every change and check the Domoticz log.

```
sudo service domoticz.sh restart
```

If the plugin behaves not as expected, then reinstall the plugin, i.e. delete, restart Domoticz, install.

# Plugin Traffic Light (Tinkerforge Building Blocks)

Sample building a Domoticz Plugin using Tinkerforge Building Blocks.

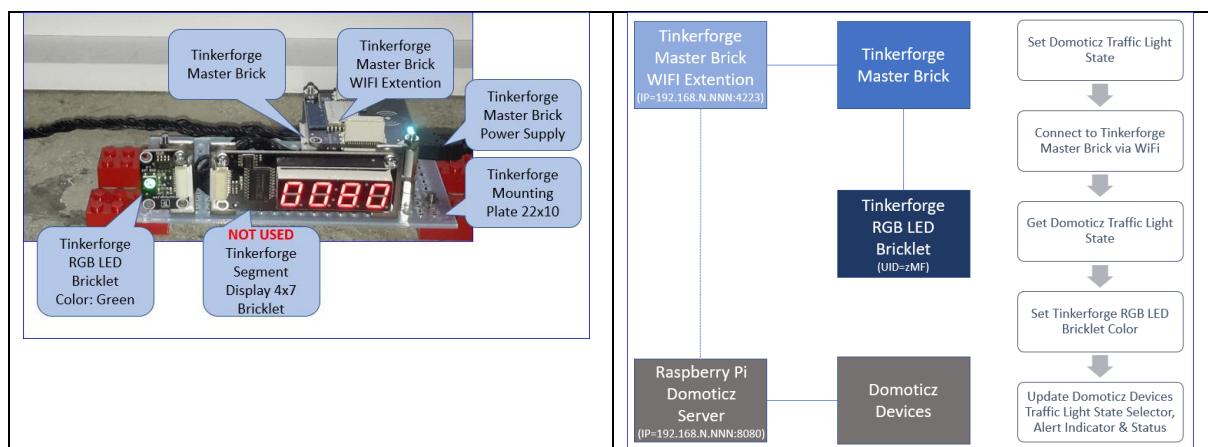
## Objectives

- To set the color of a Tinkerforge RGB Bricklet to red, yellow, or green using a Domoticz Selector Switch Device.
- To learn how to write a Python [Plugin](#) for the Domoticz system.
- To learn how to interact with [Tinkerforge](#) building blocks.
- To use this sample plugin as a template for other Domoticz plugins interacting with Tinkerforge Building Blocks.

## Solution

A Domoticz Python plugin "Traffic Light" with a Tinkerforge RGB LED Bricklet.

The Tinkerforge RGB LED Bricklet is connected to a Tinkerforge Master Brick with WiFi extension.



## Hardware

- Raspberry Pi 3B+ ([Info](#))
- Tinkerforge Master Brick 1.1 ([Info](#))
- Tinkerforge WIFI Master Extension 2.0 ([Info](#))
- Tinkerforge RGB LED Bricklet ([Info](#))

## Software

Versions for developing & using this plugin.

- Raspberry Pi Raspbian Linux 4.19.42-v7+ #1219
- Domoticz Home Automation System V4.1nnnn (BETA)
- Tinkerforge Python Binding v2.1.22
- Python 3.5.3

## Setup Tinkerforge and Domoticz

### Tinkerforge Python API Bindings Installation

```
sudo pip3 install tinkerforge
```

Check if Tinkerforge Python API bindings are installed in folder: /usr/local/lib/python3.5/dist-packages

### Tinkerforge Master Brick and Bricklets Setup

Ensure the Master Brick and Bricklets are running with the latest firmware.

To update the Tinkerforge [Brick Viewer](#) is required.

For Tinkerforge development purposes installed the Brick Viewer and the required Brick Daemon on a Linux PC (gave it the name piDevBook as running [Raspberry Pi Desktop](#)). Steps to update the Master Brick and Bricklets:

1. Connect the Master Brick to the piDevBook using USB mini cable
2. Start the Brick Viewer (ensure latest version, used v2.4.4)
3. Connect localhost:4223
4. Check if Master brickand Bricklets found
5. Select Update and check version differences
6. Update Master Brick
  - a. Button Erase - press and hold (DO NOT RELEASE)!
  - b. Button Reset - press and release
  - c. Button Erase - release!

The Master Brick Blue LED is turned off indicating boot mode.

7. The Brick Viewer shows only the Brick Tab
8. Refresh serial port= Serial Port: /dev/ttyACM0, Firmware: Master (2.4.10)
9. Flash
10. Master Brick reboots > Blue LED turns on and the Brick Viewer shows tabs Brick and Bricklets

## Traffic Light Prototype

Build the prototype by connecting the Tinkerforge building blocks (see hardware). Connect the Master Brick to a device running the Brick Deamon and Viewer.

Summary steps to setup the Tinkerforge building blocks using the Tinkerforge Brick Viewer.

- Update the devices firmware
- Set the WiFi master extension fixed IP address in client mode
- Obtain the UID's of the Tinkerforge bricklets as required by the Python plugin

After setting up the Tinkerforge building blocks, reset the master brick and check if the master brick can be reached via WLAN:

```
ping tf-wifi-ext-ip
```

## Domoticz Plugin Folder & File

The plugin **plugin.py** is installed in a subfolder of the Domoticz Plugin folder on the Domoticz system.

### **Folder:**

```
mkdir /home/pi/domoticz/plugins/trafficlight
```

### **File:**

As a starter, take the template from [here](#).

Save as **plugin.py** in the folder /home/pi/domoticz/plugins/trafficlight

## Tinkerforge API Bindings Path

In the Python Plugin code **plugin.py** amend the import path to enable using the Tinkerforge API Bindings:

```
from os import path
import sys
sys.path
sys.path.append('/usr/local/lib/python3.5/dist-packages')
```

### *Note*

Use **pip3** to install the bindings in a common dist-packages folder, which is on the Raspberry Pi Domoticz system, folder:

```
/usr/lib/python3/dist-packages
```

### Running pip3:

```
sudo pip3 install tinkerforge
```

### Log Output Sample

```
Collecting tinkerforge
  Installing collected packages: tinkerforge
    Successfully installed tinkerforge-2.1.22
```

The Tinkerforge Python API bindings are installed in folder:

```
/usr/local/lib/python3.5/dist-packages
```

Check the content results in two folders, from which the folder tinkerforge is required

```
ls /usr/local/lib/python3.5/dist-packages  
tinkerforge tinkerforge-2.1.22.dist-info
```

### Important

Depending setup of the Python distributed packages, following steps are required.

Check the folder /usr/local/lib/python3.5/dist-packages if there are more distributed packages.  
If that's the case, then leave the next steps out.

Copy the content of the folder /usr/local/lib/python3.5/dist-packages/tinkerforge to folder /usr/lib/python3/dist-packages/tinkerforge,

because all the Python distributed packages are in folder /usr/lib/python3/dist-packages/:

```
sudo cp -r /usr/local/lib/python3.5/dist-packages/tinkerforge /usr/lib/python3/dist-packages
```

Check the content of the dist-packages folder tinkerforge:

```
ls /usr/lib/python3/dist-packages/tinkerforge  
brick_dc.py bricklet_led_strip.py brick_hat.py bricklet_led_strip_v2.py ...
```

Remove the dist-packages folders:

```
sudo rm -r /usr/local/lib/python3.5/dist-packages/tinkerforge  
sudo rm -r /usr/local/lib/python3.5/dist-packages/tinkerforge-2.1.22.dist-info  
ls /usr/local/lib/python3.5/dist-packages
```

## Development Setup

Development PC:

- A shared drive Z: is defined pointing to /home/pi/domoticz
- Domoticz GUI > Setup > Log
- Domoticz GUI > Setup > Hardware
- Domoticz GUI > Setup > Devices
- WinSCP session connected to the Domoticz system
- Putty session connected to the Domoticz system

The browser tabs are required to add the new hardware with its device and monitor if the plugin code is running without errors.

## Development Iteration

The development process step:

1. Editor to develop z:\plugins\trafficlight\plugin.py – Note z: is shared [Samba](#) drive.
2. Make changes and save plugin.py
3. Restart Domoticz from Terminal: sudo service domoticz.sh restart
4. Wait a moment and refresh the browser tab Domoticz GUI > Setup > Log
5. Check the log and fix as required

### Important

In the Domoticz GUI > Setup > Settings, enable accepting new hardware.

This is required to add the new hardware with its device and monitor if the plugin code is running without errors.

## Domoticz GUI's

In a browser, opened three tabs Domoticz GUI

1. Setup > Hardware
2. Setup > Log
3. Setup > Devices

This is required to add the new hardware with its device and monitor if the plugin code is running without errors.

## Create the plugin

The plugin has a mandatory filename **plugin.py** located in the created plugin folder /home/pi/domoticz/plugins/trafficlight.

For Python development used Idle, running on a Windows 10 device.

See more, in the well documented Python Plugin source code **plugin.py**.

Domoticz Devices are created and set as used by the plugin.

```
Domoticz.Device(Name="State Selector", Unit=1, TypeName="Selector Switch", Options=Options, Used=1).Create()
Domoticz.Device(Name="Alert Indicator", Unit=2, TypeName="Alert", Used=1).Create()
Domoticz.Device(Name="Status", Unit=3, TypeName="Text", Used=1).Create()
```

The devices are manually added to the Domoticz Dashboard.

Handling the state change of the Traffic Light Selector Switch is done by the **onCommand** function.

This function updates the state of the Domoticz Devices: State Selector (IDX=13), Alert Indicator (IDX=14), Status (IDX=15).

## Plugin Pseudo Code

FIRST TIME: Function **onStart** to create the Domoticz Devices

NEXT TIME(S): Function **onCommand** to handle state changes

1. Domoticz make IP connection to the Tinkerforge Master Brick  
If error, update Domoticz Device "Status" and return
2. Get the Level of the Domoticz Device "State Selector" (Selector Switch)  
If error, update Domoticz Device "Status" and return
3. Update the Tinkerforge Bricklet RGB LED with the new Color depending Level
4. Update the Domoticz Devices "Alert Indicator" and "Status"
5. Domoticz to disconnect from the Tinkerforge Master Brick

*Note*

Function **onHeartbeat** not used as not polling for the state of a Tinkerforge Bricklet and update Domoticz Device(s) accordingly.

## Restart Domoticz

Restart Domoticz to find the plugin:

```
sudo systemctl restart domoticz.service
```

*Note*

When making changes to the Python plugin code, ensure to restart Domoticz and refresh any of the Domoticz GUI's.

## Domoticz Add Hardware Traffic Light

Prior adding set in the Domoticz Settings the option to allow new hardware.

If this option is not enabled, no new soilmoisture device is created.

Check in the Domoticz log as error message Python script at the line where the new device is used i.e. Domoticz.Debug("Device created: "+Devices[1].Name).

### Add Hardware - Check the Domoticz Log

After adding, ensure to check the Domoticz Log (Domoticz GUI, select tab Setup > Log)

Example:

```
2019-05-27 09:43:31.992 Status: (Traffic Light) Started.
2019-05-27 09:43:32.550 (Traffic Light) Debug logging mask set to: PYTHON PLUGIN QUEUE IMAGE DEVICE CONNECTION MESSAGE ALL
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzHash':'fccd39bb'
2019-05-27 09:43:32.550 (Traffic Light) 'HardwareID':'7'
2019-05-27 09:43:32.550 (Traffic Light) 'Database':'/home/pi/domoticz/domoticz.db'
2019-05-27 09:43:32.550 (Traffic Light) 'Version':'1.0.0'
2019-05-27 09:43:32.550 (Traffic Light) 'HomeFolder':'/home/pi/domoticz/plugins/TrafficLight/'
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzVersion':'4.10826'
2019-05-27 09:43:32.550 (Traffic Light) 'Name':'Traffic Light'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode6':'Debug'
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzBuildTime':'2019-05-24 10:04:40'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode4':'100'
2019-05-27 09:43:32.550 (Traffic Light) 'StartupFolder':'/home/pi/domoticz/'
2019-05-27 09:43:32.550 (Traffic Light) 'Port':'4223'
2019-05-27 09:43:32.550 (Traffic Light) 'Key':'TrafficLight'
2019-05-27 09:43:32.550 (Traffic Light) 'Language':'en'
2019-05-27 09:43:32.550 (Traffic Light) 'Address':'domoticz-ip'
2019-05-27 09:43:32.550 (Traffic Light) 'Author':'rwbl'
2019-05-27 09:43:32.550 (Traffic Light) 'UserDataFolder':'/home/pi/domoticz/'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode1':'zMF'
2019-05-27 09:43:32.550 (Traffic Light) Device count: 0
2019-05-27 09:43:32.550 (Traffic Light) Creating new Devices
```

```

2019-05-27 09:43:32.551 (Traffic Light) Creating device 'State Selector'.
2019-05-27 09:43:32.552 (Traffic Light) Device created: Traffic Light - State Selector
2019-05-27 09:43:32.552 (Traffic Light) Creating device 'Alert Indicator'.
2019-05-27 09:43:32.553 (Traffic Light) Device created: Traffic Light - Alert Indicator
2019-05-27 09:43:32.553 (Traffic Light) Creating device 'Status'.
2019-05-27 09:43:32.554 (Traffic Light) Device created: Traffic Light - Change Info
2019-05-27 09:43:32.554 (Traffic Light) Pushing 'PollIntervalDirective' on to queue
2019-05-27 09:43:32.554 (Traffic Light) Processing 'PollIntervalDirective' message
2019-05-27 09:43:32.554 (Traffic Light) Heartbeat interval set to: 60.
2019-05-27 09:43:32.547 Status: (Traffic Light) Entering work loop.
2019-05-27 09:43:32.548 Status: (Traffic Light) Initialized version 1.0.0, author 'rwbl'

```

## Domoticz Log

### State Change Off to RED

Handling the state changes are managed by the function `onCommand`.

```

2019-05-27 09:51:50.997 Status: User: Admin initiated a switch command (13/Traffic Light - State Selector/Set Level)
2019-05-27 09:51:51.002 (Traffic Light) Processing 'onCommandCallback' message
2019-05-27 09:51:51.002 (Traffic Light) Calling message handler 'onCommand'.
2019-05-27 09:51:51.003 (Traffic Light) onCommand called for Unit 1: Parameter 'Set Level', Level: 10
2019-05-27 09:51:51.012 (Traffic Light) IP Connection - OK
2019-05-27 09:51:51.012 (Traffic Light) Traffic Light - State Selector-Traffic Light State New=10
2019-05-27 09:51:51.012 (Traffic Light) Traffic Light - State Selector-nValue=0,sValue=
2019-05-27 09:51:51.017 (Traffic Light) Traffic Light - State Selector-RGB LED Colors R-G-B:0-100-0
2019-05-27 09:51:51.018 (Traffic Light) RGB LED Brightness updated:100
2019-05-27 09:51:51.018 (Traffic Light - State Selector) Updating device from 0:'' to have values 2:'10'.
2019-05-27 09:51:51.030 (Traffic Light) Traffic Light - State Selector-nValue=2,sValue=10
2019-05-27 09:51:51.030 (Traffic Light - Alert Indicator) Updating device from 0:'No Alert!' to have values 4:'RED'.
2019-05-27 09:51:51.038 (Traffic Light) Traffic Light - Alert Indicator-nValue=4,sValue=RED
2019-05-27 09:51:51.038 (Traffic Light - Status) Updating device from 0:'' to have values 0:'Traffic Light changed from 0 to 10'.
2019-05-27 09:51:51.045 (Traffic Light) Traffic Light - Change Info-nValue=0,sValue=Traffic Light changed from 0 to 10
2019-05-27 09:51:51.148 (Traffic Light) TrafficLight Update: OK

```

## Domoticz - MQTT Message

Example of an MQTT message issued by the Traffic Light Selector Switch.

Analyzing MQTT messages helps to understand the properties and values of a device.

```
{
  "Battery" : 255,
  "LevelActions" : "|||",
  "LevelNames" : "Off|RED|YELLOW|GREEN",
  "LevelOffHidden" : "true",
  "RSSI" : 12,
  "SelectorStyle" : "0",
  "description" : "",
  "dtype" : "Light/Switch",
  "id" : "00050001",
  "idx" : 13,
  "name" : "Traffic Light - State Selector",
  "nvalue" : 2,
  "stype" : "Selector Switch",
  "svalue1" : "30",
  "switchType" : "Selector",
  "unit" : 1
}
```

# RaspberryMatic

## Purpose

- To explore [RaspberryMatic](#) and how to control [Homematic](#) compatible devices via Domoticz.
  - To setup a RaspberryMatic operating system running a Homematic Central-Control-Unit (CCU) interfacing with Domoticz.
  - To create several functions using [HomematicIP](#) devices, like Pluggable Switch and Meters (HmIP-PSM), Radiator Thermostats (HmIP-eTRV-B & HmIP-eTRV-2), Window/Door Contacts (HmIP-SWDM & HmIP-SWDO).
- Note: only HomematicIP devices are used.

### Notes

As taken from [here](#): The RaspberryMatic project is a collaborate effort to provide a lightweight, Linux/buildroot-based [Homematic](#) compatible operating system for embedded single board computers (SBC) like the RaspberryPi or Tinkerboard.

## Solution

The solution runs the RaspberryMatic CCU on a Raspberry Pi 3B+ with an RPI-RF-MOD GPIO Radio Module HAT (~40EUR in 2020).

## Installation

### RaspberryMatic

The Raspberry Pi 3B+ has been setup according [these](#) guidelines.

RaspberryMatic version 3.51.6.20200621 is used (check out for newer versions [here](#)).

Several Addons installed:

Additional software for RaspMatic			
System-Update	Installed version: 1.13.12 Available version: 1.13.13	<a href="#">Download</a>  <a href="#">Uninstall</a> <a href="#">Set</a>	<b>rmupdate addon</b> <a href="https://github.com/j-a-n/raspberrymatic-addon-rmupdate">https://github.com/j-a-n/raspberrymatic-addon-rmupdate</a>
XML-API	Installed version: 1.20 Available version: 1.20	<a href="#">Download</a>  <a href="#">Uninstall</a> <a href="#">Set</a>	<b>XML-API CCU Addon</b> <a href="https://github.com/hobbyquaker/XML-API">https://github.com/hobbyquaker/XML-API</a>
CUX-Daemon	Installed version: 2.3.4 Available version: 2.3.4	<a href="#">Download</a>  <a href="#">Restart</a> <a href="#">Uninstall</a> <a href="#">Set</a>	<b>CUX-Daemon 2.3.4</b> 

# Addon XML-API

## Information

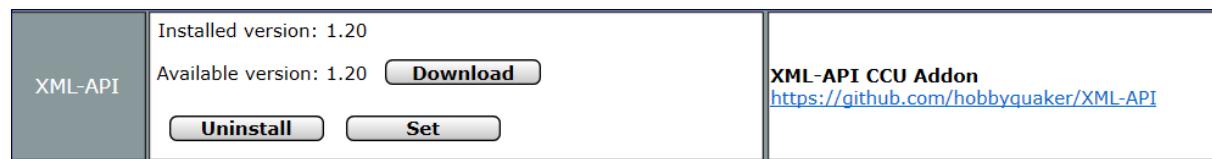
The [XML-API](#) CCU Addon is used to communicate between the CCU and Domoticz.

Installed via the RaspberryMatic WebUI > Home page > Settings > Control panel > Additional Software.

After installation, no further settings required.

The option *set* lists the available scripts with their parameter.

Understanding the XML-API concept is essential to communicate between Domoticz and the CCU v.v. = keep the [reference](#) at hand.



## XML-API Scripts

There is a variety of scripts available (see the XML-API documentation).

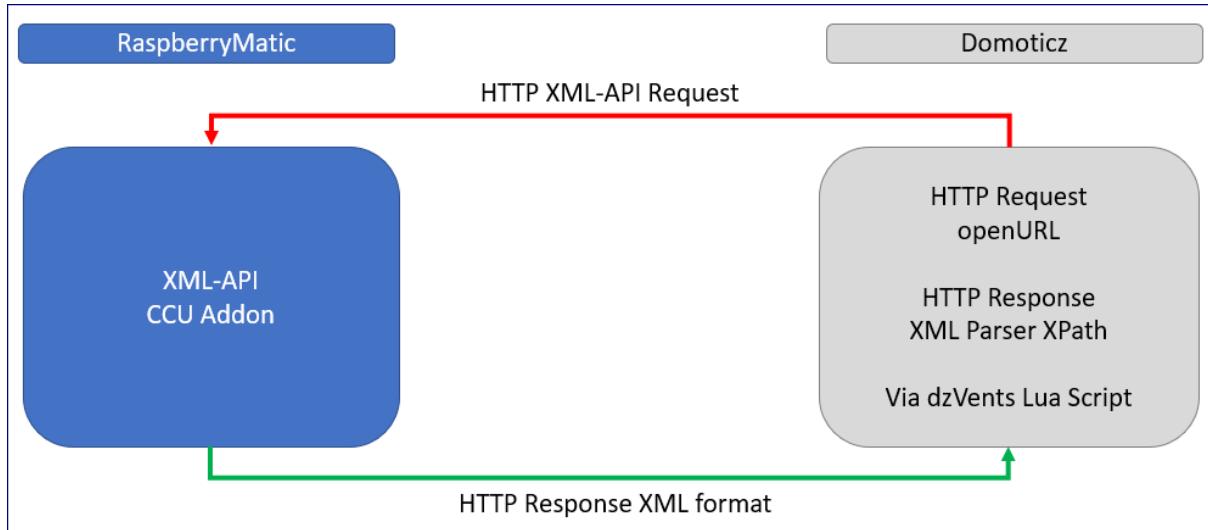
These are used by the automation scripts with relevant parameters.

Examples are

- Lists all devices with channels and current values: statelist.cgi  
This is a key script to get id's required by automation script to get or set values.  
Example:  
<http://ccu-ip/addons/xmlapi/statelist.cgi>
- List channels and values for single or multiple devices (1234,5678): state.cgi  
Example:  
[http://ccu-ip/addons/xmlapi/state.cgi?device\\_id=1541](http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541).
- Changes one or more channel states: statechange.cgi  
Example:  
[http://ccu-ip/addons/xmlapi/statechange.cgi?ise\\_id=1584&new\\_value=20](http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=20)

## Communication Flow

Domoticz triggers an HTTP XML-API request (with parameters depending script) to the CCU and parses the HTTP response in XML format using XPath.



### Note

In the next examples, the wording “push” and “pull” are used:

- CCU to push data to Domoticz (i.e. device data)
- Domoticz to pull data from the CCU (i.e. device value read)
- Domoticz to push data to the CCU (i.e. device value change)

## Example List Devices

To be able to communicate between Domoticz and the CCU, the device id's and datapoint ids are used. These can be obtained via the statelist.cgi script.

### HTTP XML-API Request

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

### HTTP Response (extract)

The response is an XML formatted list with all devices, their channels and datapoints with values.

The device name is also listed in the Homematic WebUI > Settings > Devices > Click on a device entry shows the general devices settings which enables to change the device name.

Each device has a unique serial number.

The default device name is “device type serial number”, i.e. “HmIP-eTRV-2 000A18A9A64DAC”.

For this device, changed the name to “Radiator Thermostat MakeLab”.

**RaspberryMatic – General Device Settings with Name change**

Example Radiator Thermostat with type HmIP-eTRV-2 and serial number 000A18A9A64DAC

HmIP-eTRV-2 000A18A9A64DAC	HmIP-eTRV-2		Homematic IP Radiator Thermostat	000A18A9A64DAC	HmIP-RF	Secured	Heating	MakeLab	2.8 V -68 dBm -80 dBm
-------------------------------	-------------	--	-------------------------------------	----------------	---------	---------	---------	---------	-----------------------------

↓

Radiator Thermostat MakeLab	HmIP-eTRV-2		Homematic IP Radiator Thermostat	000A18A9A64DAC	HmIP-RF	Secured	Heating	MakeLab	2.8 V -68 dBm -80 dBm
-----------------------------	-------------	--	-------------------------------------	----------------	---------	---------	---------	---------	-----------------------------

General device settings: 000A18A9A64DAC

Name: HmIP-eTRV-2 000A18A9A64DAC  
Type description: HmIP-eTRV-2  
Serial number: 000A18A9A64DAC  
Operable:   
Visible:   
Logged:   
Service messages:

**Functional test**

**Start test** →→→→→  
During the functional test the error-free communication to the device is tested. Therefore, switching commands will be sent to all actuators connected to the device. Only the first feedback message will be evaluated. The test is passed as soon as the first feedback will be received by the device.

**Cancel** **OK**

General device settings: 000A18A9A64DAC

Name: Radiator Thermostat MakeLab  
Type description: HmIP-eTRV-2  
Serial number: 000A18A9A64DAC  
Operable:   
Visible:   
Logged:   
Service messages:

**Functional test**

**Start test** →→→→→  
During the functional test the error-free communication to the device is tested. Therefore, switching commands will be sent to all actuators connected to the device. Only the first feedback message will be evaluated. The test is passed as soon as the first feedback will be received by the device.

**Cancel** **OK**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device sticky_unreach="false" unreach="false" ise_id="1596" name="HM-RC-19 CUX2801001">
  <channel ise_id="1597" name="HM-RC-19 CUX2801001:0" operate="" visible="" index="0">
    <datapoint type="LOWBAT" ise_id="1598" name="CUxD.CUX2801001:0.LOWBAT" operations="5"
    timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
    <datapoint type="UNREACH" ise_id="1606" name="CUxD.CUX2801001:0.UNREACH" operations="5"
    timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
    <datapoint type="STICKY_UNREACH" ise_id="1602" name="CUxD.CUX2801001:0.STICKY_UNREACH"
    operations="7" timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
  </channel>
  ...
</device>
<device ise_id="1011" name="HM-RCV-50 BidCoS-RF">
  <channel ise_id="1012" name="HM-RCV-50 BidCoS-RF:0" operate="" visible="" index="0">
    <datapoint type="INSTALL_MODE" ise_id="1013" name="BidCos-RF.BidCos-RF:0.INSTALL_MODE"
    operations="3" timestamp="1575570253" valueunit="" valuetype="2" value="false"/>
  </channel>
  ...

```

```

</device>
<device unreach="false" ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0" operate="true" visible="true" index="0">
    <datapoint type="LOW_BAT" ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOCAL_BAT" operations="5"
      timestamp="1575709506" valueunit="" valuetype="2" value="false"/>
    <datapoint type="OPERATING_VOLTAGE" ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE"
      operations="5" timestamp="1575709506" valueunit="" valuetype="4" value="2.800000"/>
    <datapoint type="OPERATING_VOLTAGE_STATUS" ise_id="1554" name="HmIP-
      RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" operations="5" timestamp="1575709506" valueunit=""
      valuetype="16" value="0"/>
    <datapoint type="UNREACH" ise_id="1557" name="HmIP-RF.000A18A9A64DAC:0.UNREACH" operations="5"
      timestamp="1575709506" valueunit="" valuetype="2" value="false"/>
  ...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
    <datapoint type="ACTUAL_TEMPERATURE" ise_id="1567" name="HmIP-
      RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1575709506" valueunit=""
      valuetype="4" value="21.600000"/>
    <datapoint type="SET_POINT_TEMPERATURE" ise_id="1584" name="HmIP-
      RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1575709506" valueunit="°C"
      valuetype="4" value="20.000000"/>
  </channel>
  ...
</device>
<device unreach="false" ise_id="1418" name="HMIP-PSM 0001D3C99C6AB3" config_pending="false">
  <channel ise_id="1419" name="HMIP-PSM 0001D3C99C6AB3:0" operate="true" visible="true" index="0">
    <datapoint type="OPERATING_VOLTAGE" ise_id="1426" name="HmIP-RF.0001D3C99C6AB3:0.OPERATING_VOLTAGE"
      operations="5" timestamp="0" valueunit="" valuetype="4" value="0.000000"/>
    <datapoint type="ACTUAL_TEMPERATURE" ise_id="3065" name="HmIP-
      RF.0001D3C99C6AB3:0.ACTUAL_TEMPERATURE" operations="5" timestamp="0" valueunit="" valuetype="4"
      value="0.000000"/>
  </channel>
  ...
</device>
</stateList>
```

*Note*

Device HmIP-eTRV-2 000A18A9A64DAC listed with default name.

```

<device unreach="false" ise_id="1541" name="Radiator Thermostat MakeLab" config_pending="false">
  <channel ise_id="1542" name="Radiator Thermostat MakeLab:0" operate="true" visible="true" index="0">
    <datapoint type="OPERATING_VOLTAGE" ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE"
      operations="5" timestamp="1575710600" valueunit="" valuetype="4" value="2.800000"/>
  ...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
    <datapoint type="ACTIVE_PROFILE" ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE"
      operations="7" timestamp="1575710600" valueunit="" valuetype="16" value="1"/>
  ...

```

*Note*

Device HmIP-eTRV-2 000A18A9A64DAC listed with name changed to “Radiator Thermostat MakeLab”.

## HmIP-eTRV-2 Device Request

Domoticz to request (“pull”) the setpoint & temperature of the HomematicIP radiator thermostat (HMIP-eTRV-2) with the device id=1541:

1. dzVents Lua script event sends, every minute, an HTTP XML-API request (with callback) for device information (using the device id) to the CCU  
[http://ccu-ip/addons/xmlapi/state.cgi?device\\_id=1541](http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541).
2. The CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents Lua script handles the callback, parses XML string using XPath to get the values of the setpoint & temperature by reading their datapoints.
4. dzVents Lua script updates the Domoticz text device with IDX=175.

### HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoints to pick, i.e. use in the automation script (dzVents Lua).

#### *HTTP URL*

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

#### *HTTP Response (extract)*

The datapoints required are the SET\_POINT\_TEMPERATURE (ise\_id=1584) and the ACTUAL\_TEMPERATURE (ise\_id=1567).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <device config_pending="false" unreach="false" ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
      ...
      <datapoint ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE" timestamp="1575647467"
      valueunit="" valuetype="4" value="2.800000" type="OPERATING_VOLTAGE"/>
      <datapoint ise_id="1554" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS"
      timestamp="1575647467" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
      ...
    </channel>
    <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
      ...
      <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1575647467"
      valueunit="" valuetype="4" value="19.300000" type="ACTUAL_TEMPERATURE"/>
      <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_MODE" timestamp="1575647467"
      valueunit="" valuetype="16" value="1" type="SET_POINT_MODE"/>
      <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
      timestamp="1575647467" valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
      ...
    </channel>
    ...
  </device>
</state>
```

## Automation Script

Event name: thermostat\_makelab\_info

```
-- Domoticz device text (named MakeLab Thermostat Info)
local IDX_MAKELAB_THERMOSTAT_INFO = 50;
local ID_DEVICE = 1541;
-- url of the raspmatic webserver to obtain device information
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
-- callback of the url request - must be unique across all automation events
local RES_RASPMATIC = 'RES_makelab_thermostat_info';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = {
        timer = {
            -- 'every minute' -- for tests
            'every 5 minutes'
        },
        httpResponses = {
            RES_RASPMATIC -- must match with the callback passed to the openURL command
        }
    },
    execute = function(domoticz, item)
        -- check if the item is a device, then request information
        if (item.isTimer) then
            domoticz.openURL({url = URL_RASPMATIC, method = 'GET', callback = RES_RASPMATIC})
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                -- multiple datapoints - domoticz.log(item.data);
                -- parse the response using XPath
                -- select the attribute value of the datapoint element (this is XPath syntax)
                -- SETPOINT °C
                local setpointvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
                -- TEMPERATURE °C
                local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')

                -- log
                local msg = 'Setpoint: ' .. tostring(round(setpointvalue,DECIMALS)) .. ', Temperature: ' ..
                tostring(round(temperaturevalue,DECIMALS))
                domoticz.log(msg)

                -- update the domoticz device
                domoticz.devices(IDX_MAKELAB_THERMOSTAT_INFO).updateText(msg);

            else
                domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

## HmIP-eTRV-2 Single Datapoint Request

Domoticz to request (“pull”) the temperature of the HomematicIP radiator thermostat (HMIP-eTRV-2) with the device id=1541, datapoint id=1567:

1. dzVents Lua script event sends, every minute, an HTTP XML-API request (with callback) for datapoint information (using datapoint id) to the CCU  
[http://ccu-ip/addons/xmlapi/state.cgi?datapoint\\_id=1567](http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=1567).
2. The CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents Lua script handles the callback, parses XML string using XPath to get the values of the temperature datapoint.
4. dzVents Lua script updates the Domoticz temp device with IDX=65.

### *Domoticz Device*

Created as Virtual Sensor Type Temperature, listed as device type Temp, SubType LaCrosse TX3.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
65	VirtualDevices	14091	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.4 C	

## HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoint to pick, i.e. use in the automation script (dzVents Lua).

### *HTTP URL*

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=1567
```

### *HTTP Response*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="22.400000" ise_id="1567"/>
</state>
```

## Automation Script

Event name: thermostat\_makelab\_temp

```

local IDX_MAKELAB_THERMOSTAT_TEMP = 65;
local ID_DATAPOINT = 1567;
-- url of the raspmatic webserver to obtain device information
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=' .. ID_DATAPOINT;
-- callback of the url request - must be unique across all automation events
local RES_RASPMATIC = 'RES_makelab_thermostat_temp';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
    on = {
        timer = {
            -- 'every minute' -- for tests
            'every 5 minutes'
        },
        httpResponses = {
            RES_RASPMATIC -- must match with the callback passed to the openURL command
        }
    },
    execute = function(domoticz, item)
        -- check if the item is a device, then request information
        if (item.isTimer) then
            domoticz.openURL({url = URL_RASPMATIC, method = 'GET', callback = RES_RASPMATIC})
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then

            if (item.statusCode == 200) then
                -- <datapoint value="22.400000" ise_id="1567"/>
                local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')
                -- log
                local msg = 'Temperature: ' .. tostring(round(temperaturevalue,DECIMALS))
                domoticz.log(msg)
                -- update the domoticz device
                domoticz.devices(IDX_MAKELAB_THERMOSTAT_TEMP).updateTemperature(round(temperaturevalue,1));
            else
                domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}

```

*Note*

By opening multiple URL's with unique callback string, it is possible to obtain multiple datapoints. This could be used to check the battery status of several HomematicIP devices.

## HmIP-eTRV-2 Multiple Datapoints Request

Builds upon previous example, but Domoticz to request, i.e. pull various datapoints for multiple devices.

The example below is taken from the function [Heating System Dashboard](#).

### *Principle*

Define a Lua table with properties for each device, i.e. the RaspberryMatic device datapoint and the corresponding Domoticz IDX for the devices getting RaspberryMatic datapoint attributes values assigned.

### *Example*

Get datapoints for the RaspberryMatic device with ise\_id=1541. Select the datapoint with type ACTUAL\_TEMPERATURE and assign the value to the Domoticz Temperature device with IDX=51.

### *Note*

It is also possible to let RaspberryMatic push device data triggered by value change. This requires a RaspberryMatic program (see function [Radiator Thermostat \(HmIP-eTRV\)](#)).

## Automation Script

Event name: heating\_system\_control.dzvents

```
-- Helper
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()

-- Create JSON object per heat exchanger (thermostat) for openURL and HTTP response
-- Device properties:
-- RaspberryMatic device datapoint ise_id
-- Domoticz IDX for TRCnnnnSP (Thermostat), TRCnnnnPV (Temperature), TCVnnnnPV (Percentage)
-- dzVents HTTP response (The response must be unique across all dzevents)
-- Keep the keys for the device properties to lowercase

-- E-6201 = Heat Exchanger MakeLab
local E6201 = JSON:decode('{"iseid":1541,"idxtrcsp":52,"idxtrcpv":51,"idxtcvpv":53,"response":"U6000E6201"}')
-- E-6301 = Heat Exchanger Wohnzimmer-1
local E6301 = JSON:decode('{"iseid":3543,"idxtrcsp":55,"idxtrcpv":54,"idxtcvpv":56,"response":"U6000E6301"}')

-- RaspberryMatic HTTP XMLAPI
-- URL of the CCU to obtain device information
local HTTPREQUEST = 'http://raspmatic-ip/config/xmlapi/state.cgi?device_id='
-- Define the XMLAPI attributes to get data from
local TYPE_SET_POINT_TEMPERATURE = "SET_POINT_TEMPERATURE" -- TRCnnnnSP
local TYPE_ACTUAL_TEMPERATURE = "ACTUAL_TEMPERATURE" -- TRCnnnnPV
local TYPE_LEVEL = "LEVEL" -- TCVnnnnPV

-- Update the domoticz device data with the values
-- Info: ----- Start internal script: u6000_control: HTTPResponse: "U6000E6201"
-- {[{"iseid":1541, ["level":100, ["temperature":20.7, ["setpoint":23.0}]}
-- Updating Device: 1541, SP: 23.0, PV: 20.7, L: 1.0
-- Info: ----- Finished u6000_control
local function updateDevices(domoticz, deviceProperties, deviceValues)
    domoticz.log(deviceValues)
    domoticz.log(('Updating Device: %s, SP: %s, PV: %s, L: %s'):format(
        deviceValues.iseid, deviceValues.setpoint, deviceValues.temperature, deviceValues.level))
    -- Update the devices using the p&id nomenclature
    domoticz.devices(deviceProperties.idxtrcsp).updateSetPoint(deviceValues.setpoint);
    domoticz.devices(deviceProperties.idxtrcpv).updateTemperature(deviceValues.temperature);
    domoticz.devices(deviceProperties.idxtcvpv).updatePercentage(deviceValues.level);
end

return {
    on = {
        timer = {'every minute'},
    }
}
```

```
httpResponses = {E6201.response, E6301.response,},  
execute = function(domoticz, item)  
    -- check if the item is a device, then request information  
    if (item.isTimer) then  
        -- Submit HTTP Requests to RaspMatic -- wait 1 sec after each request  
        domoticz.openURL({  
            url = HTTPREQUEST .. E6201.iseid, method = 'GET', callback = E6201.response,})  
        domoticz.openURL({  
            url = HTTPREQUEST .. E6301.iseid, method = 'GET', callback = E6301.response,}).afterSec(1)  
    end  
  
    -- check if the item is a httpresponse from the openurl callback  
    if (item.isHTTPResponse) then  
        if (item.statusCode == 200) then  
            -- From the HTTP response, check which device returned data  
            local deviceValues = {}  
  
            -- Select the datapoints by using the type name.  
            -- this makes selecting datapoints easy as no need for the datapoint ise_id per device  
            deviceValues.setpoint = tonumber(domoticz_applyXPath(item.data,  
                ('//datapoint[@type="%s"]/@value'):format(TYPE_SET_POINT_TEMPERATURE)))  
            deviceValues.temperature = tonumber(domoticz_applyXPath(item.data,  
                ('//datapoint[@type="%s"]/@value'):format(TYPE_ACTUAL_TEMPERATURE)))  
            deviceValues.level = tonumber(domoticz_applyXPath(item.data,  
                ('//datapoint[@type="%s"]/@value'):format(TYPE_LEVEL))) * 100  
  
            -- Select the device(s) to update  
            if deviceValues.iseid == E6201.iseid then updateDevices(domoticz, E6201, deviceValues) end  
            if deviceValues.iseid == E6301.iseid then updateDevices(domoticz, E6301, deviceValues) end  
  
        else  
            domoticz.log('[ERROR] Problem handling the request: ' .. item.statusText, domoticz.LOG_ERROR)  
        end  
    end  
end  
}
```

## HmIP-eTRV-2 Datapoint Change Value

Domoticz to request (“push”) changing the setpoint of the HomematicIP radiator thermostat (HMIP-eTRV-2) with the device id=1541, datapoint id=1584.

Solutions have been worked:

1. Device Switch Type Selector – Selector Levels Actions
2. Automation Event (dzVents Lua script)
3. Device Switch Type Push Off Button

My preferred solution is using the Automation Script as easier to maintain, enables to handle multiple devices, handle HTTP response (for example log or set alert message).

### Solution Device Switch Type Selector – Selector Level Actions

For this solution, Domoticz uses device level actions (as the switch type is selector), by submitting for each level the HTTP XML-API request to the CCU.

The HTTP Response is not handled.

**MakeLab Thermostat Setpoint** 20

Last Seen: 2019-12-06 13:57:20  
Type: Light/Switch, Switch, Selector

0 18 19 20 21

★ Log Edit Timers Notifications

Idx: 49  
Name: MakeLab Thermostat Setpoint  
Switch Type: Selector

Switch Icon: Heating (Heating device)

On Delay: 0 (Seconds) 0 = Disabled  
Off Delay: 0 (Seconds) 0 = Disabled  
Protected: ■

Selector Style:  Button set  Select menu  
Hide Off level: ■

Level	Level name	Order
0	0	<span style="color: green;">▼</span> <span style="color: red;">▲</span> <span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
10	18	<span style="color: green;">▼</span> <span style="color: red;">▲</span> <span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
20	19	<span style="color: green;">▼</span> <span style="color: red;">▲</span> <span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
30	20	<span style="color: green;">▼</span> <span style="color: red;">▲</span> <span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
40	21	<span style="color: red;">▲</span> <span style="color: blue;">✖</span> <span style="color: blue;">✚</span>

Level name:  Add

Level Action

Level	Action	Order
0	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=0	<span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
10	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=18	<span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
20	http://CCU-IP i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=19	<span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
30	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=20	<span style="color: blue;">✖</span> <span style="color: blue;">✚</span>
40	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=21	<span style="color: blue;">✖</span> <span style="color: blue;">✚</span>

Description: MakeLab homematicIP Radiator Thermostat change setpoint using Level Actions via HTTP XML-API requests.

Save Delete Replace

## HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The HTTP URL used is the same as the device level action defined.

This example request changes the setpoint to 18°C for datapoint=1584.

The response confirms the changed id and the new value.

### *HTTP URL*

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=18
```

### *HTTP Response*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<result>
  <changed id="1584" new_value="18"/>
</result>
```

## Solution Automation Script

Instead device level actions, an option is to use an Automation Script triggered by device changes (the level name is used as the new setpoint), which opens the URL (HTTP XML-API request) with a HTTP callback.

The callback handles the response.

The Automation Script could also handle multiple devices.

## Automation Script

```
-- makelab_thermostat_setpoint
-- set the setpoint of the radiator thermostat channel HmIP-eTRV-2 000A18A9A64DAC:1
-- set a new setpoint via url example:
-- http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=17
-- response:
-- <result><changed id="1584" new_value="17.0"/></result>

-- Domoticz device idx
local IDX_MAKELAB_THERMOSTAT_SETPOINT = 49;

-- raspmatic datapoint id taken from statelist
-- <datapoint type="SET_POINT_TEMPERATURE" ise_id="1584" name="HmIP-
RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1575709506" valueunit="°C"
valuetype="4" value="20.00000"/>
local ID_DATAPOINT_SETPOINT = 1584;

-- url of the raspmatic webserver to set the new setpoint
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_SETPOINT
.. '&new_value='

-- callback of the url request - must be unique across all dzevents - use prefix res + script name
local RES_RASPMATIC = 'res_makelab_thermostat_setpoint';

-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
  local power = 10^decimals
  return math.floor(number * power) / power
end

return {
  on = {
    devices = {
      IDX_MAKELAB_THERMOSTAT_SETPOINT
    },
  }
}
```

```

httpResponses = {
    RES_RASPMATIC
}
},
execute = function(domoticz, item)
    -- check if item is device to trigger setpoint change using the levelname (0,17,18,...)
    if (item.isDevice) then
        domoticz.log('Device ' .. domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).name .. ' was changed
to ' .. tostring(domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName), domoticz.LOG_INFO)
        -- get the new setpoint from the levelname
        -- 0%=0(OFF) ; 10%-40%=18-21 ; min=18, max=21
        local level = domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName;
        local newsetpoint = tonumber(level);
        -- set the new setpoint
        domoticz.log('New setpoint:' .. URL_RASPMATIC .. tostring(newsetpoint), domoticz.LOG_INFO);
        domoticz.openURL({url = URL_RASPMATIC .. tostring(newsetpoint), method = 'POST', callback =
RES_RASPMATIC});
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
        if (item.statusCode == 200) then
            -- the full http xml response: domoticz.log(item.data);
            -- parse the response using XPath
            -- select the attribute value of the changed element (this is XPath syntax)
            -- <changed id="1584" new_value="18"/>
            domoticz.log('CCU Response new value: ' ..
domoticz_applyXPath(item.data,'//changed[@id="1584"]/@new_value'))
            -- The response could also be logged to an alert or control message device to log changes
        else
            domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
        end
    end
end
}

```

## Domoticz Log

The log shows the HTTP request and the HTTP response.

```

2019-12-07 11:48:54.970 (VirtualDevices) Light/Switch (MakeLab Thermostat Setpoint)
2019-12-07 11:48:54.963 Status: User: Admin initiated a switch command (49/MakeLab Thermostat
Setpoint/Set Level)

2019-12-07 11:48:55.103 Status: dzVents: Info: Handling events for: "MakeLab Thermostat Setpoint",
value: "19"
2019-12-07 11:48:55.103 Status: dzVents: Info: ----- Start internal script:
makelab_thermostat_setpoint: Device: "MakeLab Thermostat Setpoint (VirtualDevices)", Index: 49
2019-12-07 11:48:55.103 Status: dzVents: Info: Device MakeLab Thermostat Setpoint was changed to 19
2019-12-07 11:48:55.103 Status: dzVents: Info: New setpoint:http://ccu-
ip addons/xmlapi/statechange.cgi?se_id=1584&new_value=19
2019-12-07 11:48:55.103 Status: dzVents: Info: ----- Finished makelab_thermostat_setpoint

2019-12-07 11:48:55.104 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2019-12-07 11:48:56.019 Status: dzVents: Info: Handling httpResponse-events for:
"res_makelab_thermostat_setpoint"
2019-12-07 11:48:56.019 Status: dzVents: Info: ----- Start internal script:
makelab_thermostat_setpoint: HTTPResponse: "res_makelab_thermostat_setpoint"
2019-12-07 11:48:56.024 Status: dzVents: Info: CCU Response new value: 19
2019-12-07 11:48:56.024 Status: dzVents: Info: ----- Finished makelab_thermostat_setpoint

```

## Device Switch Type Push Off Button

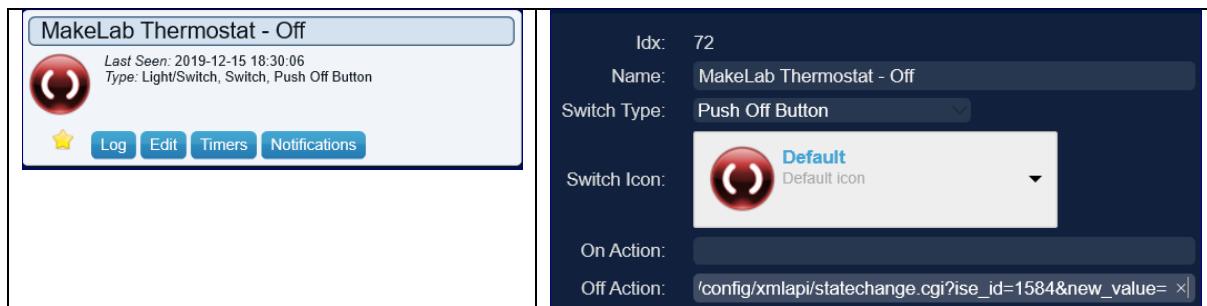
The state change action can also be assigned to a Switch to turn the thermostat Off.

Define a VirtualSensor device with

- Name, i.e. “MakeLab Thermostat - Off”
- Sensor Type Switch.

After adding the device, select the widget and change the properties:

- Switch Type: Push Off Button
- Off Action:  
`http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=0`



### Domoticz Log

Test by pushing the icon.

```
2019-12-15 18:30:06.767 (VirtualDevices) Light/Switch (MakeLab Thermostat - Off)
2019-12-15 18:30:06.758 Status: User: Admin initiated a switch command (72/MakeLab Thermostat - Off/Off)
```

## HmIP-eTRV-2 Update Domoticz Device Value

The CCU to update (“push”) the Domoticz device “MakeLab Thermostat Temperature” (IDX=65), if the “Actual Temperature Value” from the HomematicIP radiator thermostat (HmIP-eTRV-2) changes. So, every change in temperature is sent to Domoticz. For this example, the Domoticz development system is used.

1. The RaspMatic script “HmIP-eTRV-2-MakeLab” immediately runs if the following rule applies:  
“Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature within value range / with value from 1.00 and less than 0.00 trigger when changed”
2. A script reads the actual state of the datapoint “HmIP-RF.000A18A9A64DAC:1.ACTUAL\_TEMPERATURE”, builds & submits the URL of the Domoticz API request to update the value of the Domoticz device “MakeLab Thermostat Temperature” (IDX=65). The HTTP response is logged.

### Note

The solution is rather device bound, means for every device an XML-API script is required. If this is not required, to measure every value change, then consider using a Domoticz Automation Script Event to pull value(s) [of one or more devices] in regular intervals (i.e. every minute or 5 minutes). See previous Example Datapoint Request.

### XML-API Script

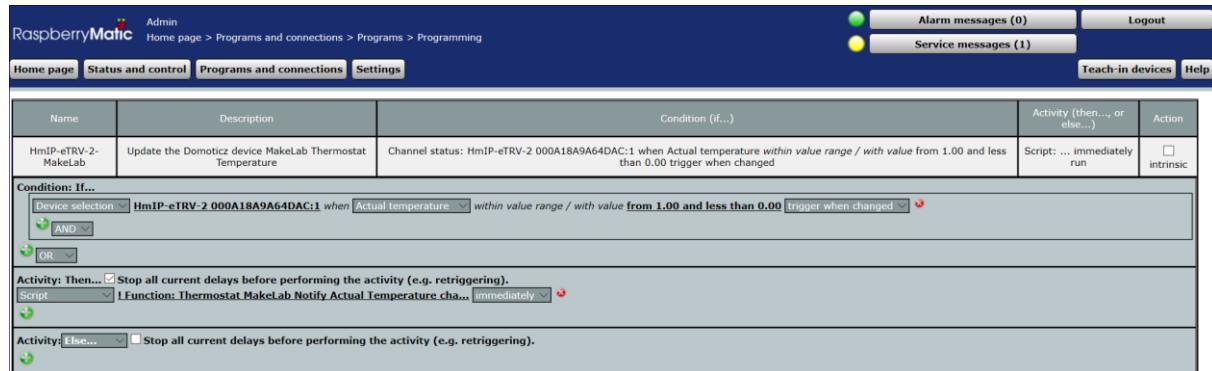
The script makes use of a CUxD device (see Addon CUxD) to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz device. Example:

```
wget -q -O - 'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=20.89'
```

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

HmIP-eTRV-2-MakeLab	Update the Domoticz device MakeLab Thermostat Temperature	Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature <i>within value range / with value from 1.00 and less than 0.00 trigger when changed</i>	
			Script: ... immediately run

The screenshot shows the RaspberryMatic programming interface. At the top, there's a navigation bar with 'Admin' and links to 'Home page', 'Status and control', 'Programs and connections', 'Programs', and 'Programming'. On the right, there are buttons for 'Alarm messages (0)', 'Logout', 'Service messages (1)', 'Teach-in devices', and 'Help'. Below the navigation, there's a table for defining a script. The table has columns for 'Name', 'Description', 'Condition (if...)', 'Activity (then..., or else...)', and 'Action'. The 'Name' column contains 'HmIP-eTRV-2-MakeLab'. The 'Description' column contains 'Update the Domoticz device MakeLab Thermostat Temperature'. The 'Condition (if...)' column contains the condition 'Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature *within value range / with value from 1.00 and less than 0.00 trigger when changed*'. The 'Activity (then..., or else...)' column contains 'Script: ... immediately run'. The 'Action' column contains 'intrinsic'. Below the table, there's a 'Condition: If...' section with a dropdown for 'Device selection' set to 'HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature *within value range / with value from 1.00 and less than 0.00 trigger when changed*'. There are also sections for 'Activity: Then...' and 'Activity: Else...'.

## RaspberryMatic Script: thermostat\_makelab\_temp.script

```

! Function: Thermostat MakeLab Notify Actual Temperature change
! 20191208 by rwbl
string function = "Thermostat MakeLab Temperature";
string actdate = system.Date("%d.%m"); ! sDate = "09.08";
! string actDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string acttime = system.Date("%H:%M"); ! sTime = "07:32";
! string actime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";

! A domoticz virtual temperature device is used
string idx = 65; ! Type: Temp; SubType: LaCrosse TX3
! Define the parameter nvalue and svalue - see domoticz api documentation
string nvalue = 0;
! The object string is taken from the raspmatic XML-API script statelist.cgi
string svalue = dom.GetObject("HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE").State();
! for tests: string svalue = 20;

! Build the domoticz http api request url to set the actual temperature on the domoticz device
! Example: http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=20.89
string amp = "&";
string urldom = "'http://domoticz-ip:8080/json.htm'; ! Production=60,Development=179
string urlrequest =
urldom"?type=command#amp#param=udevice#amp#idx="#idx#amp#nvalue="#nvalue#amp#svalue="#svalue#"';
! For Tests.
! Output'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=21.700000'
WriteLine(urlrequest);

! OPTION: Run the command without a return result
! res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);
! OPTION: Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#urlrequest);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: The script running on the CCU waits until the completion - ensure not to execute commands which
take long time.
string res = dom.GetObject("CUxD.CUX2801001:1.CMD_RETs").State();
! WriteLine("VT="#res.VarType()#/##res); ! VT=4, {"status" : "OK","title" : "Update Device"}

! handle result
var domlog = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (res.Find("OK") > 0) {
  domlog.Variable(function#"-Last update OK:#actdate#" "#acttime ")
}
else {
  domlog.Variable(function#"-Last update ERROR:#actdate#" "#acttime")
};
! WriteLine("VT="#domlog.VarType()); ! VT=9
WriteLine(domlog.Variable()); ! shows the last update string

```

## RaspberryMatic Log Entry

Menu: Home page > Status and control > System variables

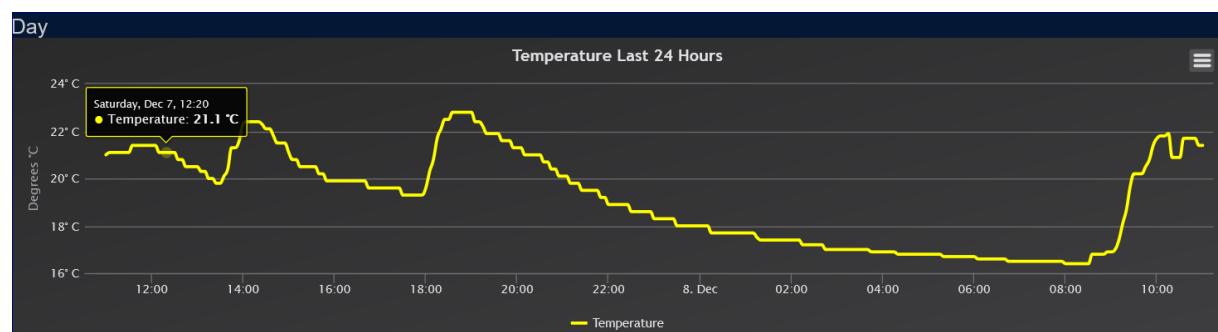
DomoticzLog	Domoticz Log Messages	03.12.2019 23:44:44	Thermostat MakeLab Temperature- Last update OK:08.12 11:14
-------------	-----------------------	---------------------	---

## Domoticz Device

The Domoticz device is a virtual device (see below properties).

The temperature is updated by the HTTP API request triggered by the CCU (“push”), there is no need to define an automation event.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
65	VirtualDevices	14091	1	Makelab Thermostat Temperature	Temp	LaCrosse TX3	21.4 C



## HmIP-SWDO Datapoint Request

Another example of a Domoticz Automation Event to request (“pull”) the value of a HomematicIP device datapoint value.

The device is a SWDO from which he datapoint Operating Voltage (id=2543) is pulled. The datapoint required information, i.e. ise\_id) is taken from the statelist request URL:

```
http://ccu-ip/addons/xmlapi/state.cgi
```

```
<datapoint ise_id="2543" name="HmIP-RF.0000DA498D5859:0.OPERATING_VOLTAGE" operations="5" timestamp="1575914670" valueunit="" valuetype="4" value="1.500000" type="OPERATING_VOLTAGE"/>
```

### HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.

The response lists the datapoint information value and id, to be used in the automation script .

#### HTTP URL

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=2543
```

#### HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="1.400000" ise_id="2543"/>
</state>
```

## Automation Script

Event name: postbox\_voltage.lua

```
--[[[

postbox_voltage.lua
Check the voltage of the HomematicIP device SWDO built into the postbox.
Log if voltage is below threshold.

]]-
local URL_RASPMATIC = 'http:// ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux
local DATAPOINT2543 = JSON:decode('{"name":"Postbox
Voltage","id":2543,"xpath":"/datapoint[@ise_id=2543]/@value","response":"RESPONSEBOXVOLTAGE"}');
-- swdo device: set low voltage threshold (same as the RaspberryMatic device parameter.
local TH_SWDO_LOW_VOLTAGE = 1.0;

return {
  on = {
    timer = {
      'every minute' -- for tests
      -- 'every 30 minutes'
    },
    httpResponses = {
      DATAPOINT2543.response,
    }
  },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({url = URL_RASPMATIC .. DATAPOINT2543.id, method = 'GET', callback =
DATAPOINT2543.response,})
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        domoticz.log(item.data);
      end
    end
  end
}
```

```

-- Select the callback - in case several datapoints
if (item.callback == DATAPOINT2543.response) then
    local voltage = tonumber(domoticz_applyXPath(item.data, DATAPOINT2543.xpath))
    domoticz.log(DATAPOINT2543.name .. ':' .. voltage)
    if voltage < TH_SWDO_LOW_VOLTAGE then
        local message= DATAPOINT2543.name .. ':Low Voltage ' .. voltage .. ' ' ..
domoticz.helpers.isnowhhmm(domoticz)
        domoticz.log(message)
        -- in production system the set alert
        -- domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
    end
end
else
    domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
end
end
}

```

## Domoticz Log

```

2019-12-12 09:46:00.223 Status: dzVents: Info: ----- Start internal script: postbox_voltage:, trigger:
every minute
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Finished postbox_voltage
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Start internal script: thermostat_makelab_temp:, trigger:
every minute
2019-12-12 09:46:00.224 Status: dzVents: Info: ----- Finished thermostat_makelab_temp

2019-12-12 09:46:00.224 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2019-12-12 09:46:00.362 Status: dzVents: Info: Handling httpResponse-events for: "RESPONSEBOXVOLTAGE"
2019-12-12 09:46:00.362 Status: dzVents: Info: ----- Start internal script: postbox_voltage:
HTTPResponse: "RESPONSEBOXVOLTAGE"
2019-12-12 09:46:00.367 Status: dzVents: Info: <?xml version="1.0" encoding="ISO-8859-1"
?><state><datapoint ise_id='2543' value='1.400000'/'></state>
2019-12-12 09:46:00.367 Status: dzVents: Info: Postbox Voltage: 1.4
2019-12-12 09:46:00.367 Status: dzVents: Info: ----- Finished postbox_voltage

```

# Addon CUxD

## Information

The [CUxD CCU Addon](#) is used for communication triggered by the CCU to Domoticz. The CCU submits Domoticz HTTP API Requests via system command wget.

This concept can be used for example to

- handle Window/Door Contact state changes and update Domoticz Alert Device
- handle manual Radiator Thermostat changes and sync the Domoticz Setpoint Device

Get started, by reading [here](#) and [download](#) latest version.

Install via the RaspberryMatic WebUI > Home page > Settings > Control panel > Additional Software.

## CUxD Configure with Set



## CUxD Set Menu



### Add a new “Geräte” device

Select CUxD Gerätetyp (28) System with Function Exec, Leave Name empty, Use Gerät Icon “Fernbedienung 19 Tasten”.

Press Button “Gerät auf CCU erzeugen !”.



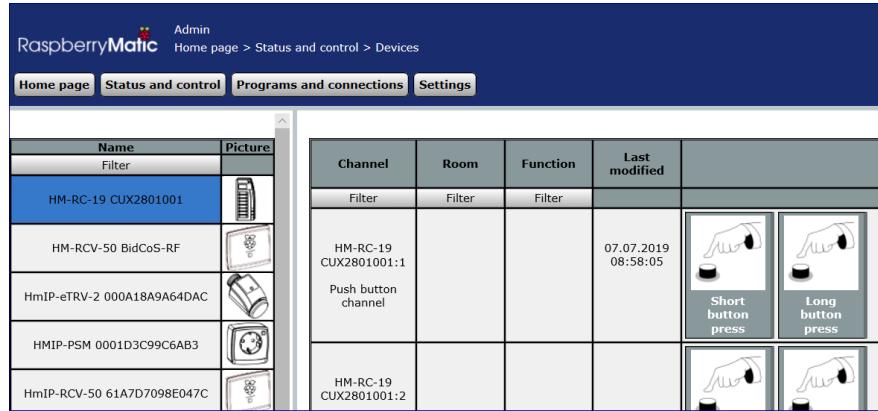
If all goes well, the RaspberryMatic WebUI Device Inbox lists the new device.

Switch to the RaspberryMatic WebUI.

## RaspberryMatic WebUI

Select Settings > Device Inbox and add the CUxD.

The CUxD device created is **HM-RC-19 CUX2801001** with 16 push-button channels (called “Push button channel”).



Check the device channels and datapoints with a XML-API HTTP Request:

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

Resulting in:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
  - <device sticky_unreach="false" unreach="false" ise_id="1596" name="HM-RC-19 CUX2801001">
    - <channel ise_id="1597" name="HM-RC-19 CUX2801001:0" operate="" visible="" index="0">
      <datapoint ise_id="1598" name="CUxD.CUX2801001:0.LOWBAT" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="LOWBAT"/>
      <datapoint ise_id="1606" name="CUxD.CUX2801001:0.UNREACH" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="UNREACH"/>
      <datapoint ise_id="1602" name="CUxD.CUX2801001:0.STICKY_UNREACH" operations="7" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="STICKY_UNREACH"/>
    </channel>
    - <channel ise_id="1610" name="HM-RC-19 CUX2801001:1" operate="true" visible="true" index="1">
      <datapoint ise_id="1628" name="CUxD.CUX2801001:1.PRESS_SHORT" operations="6" timestamp="1562482685" valueunit="" valuetype="2" value="false" type="PRESS_SHORT"/>
      <datapoint ise_id="1627" name="CUxD.CUX2801001:1.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
      <datapoint ise_id="1612" name="CUxD.CUX2801001:1.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_KILL"/>
      <datapoint ise_id="1611" name="CUxD.CUX2801001:1.CMD_EXEC" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_EXEC"/>
      <datapoint ise_id="1634" name="CUxD.CUX2801001:1.WORKING" operations="5" timestamp="1562428036" valueunit="" valuetype="2" value="false" type="WORKING"/>
    </channel>
    - <channel ise_id="1636" name="HM-RC-19 CUX2801001:2" operate="true" visible="true" index="2">
      <datapoint ise_id="1654" name="CUxD.CUX2801001:2.PRESS_SHORT" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_SHORT"/>
      <datapoint ise_id="1653" name="CUxD.CUX2801001:2.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
    </channel>
  </device>
</stateList>
```

## Example Update Domoticz Text Device

Update the text of the Domoticz Text Device (IDX=36) by a short button press of channel:1 (name HM-RC-19 CUX2801001:1) of the device HM-RC-19 CUX2801001.

See also function [Postbox Notifier \(RaspberryMatic\)](#) for live example.

### RaspMatic Program

Logic: If CHANNEL:1 button press short THEN execute script.

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
CUX2801001Button1	Send Message to Domoticz Text Device	Channel status: HM-RC-19 CUX2801001:1 when Button press short	Script: ... immediately run	<input type="checkbox"/> intrinsic
<b>Condition: If...</b>				
Device selection <input type="button" value="HM-RC-19 CUX2801001:1"/> when <input type="button" value="Button press short"/> <input checked="" type="radio"/> AND <input type="radio"/> OR				
Activity: Then... <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). Script <input !="" %d.%m.%y\");="" sdate='\"09.08.201..."/' type="button" value="string sDate = system.Date(\"/> <input type="button" value="immediately"/> <input type="button" value="..."/>				
Activity: Else... <input type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering).				

### RaspMatic Script

Submit Domoticz HTTP API Request via system command wget, to update the Domoticz Text Device.

```
wget -q -O - URL
```

The URL is the HTTP API Request as defined per Domoticz HTTP API/JSON documentation.

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=iDX&nvalue=0&svalue=MESSAGE
```

```
string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";
string nIdx = 36;
string sMsg = sDate # " # sTime # " - Threshold reached! Take action...";
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm'";
string sUrl =
sDomUrl#"?type=command#cAmp#param=udevice#cAmp#idx="#nIdx#cAmp#"nvalue=0#cAmp#"svalue="#sMsg#"";
! Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);
! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: script running on CCU waits until completion - use commands which do not take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RET").State();
! WriteLine("VT="#sRes.VarType()#"/#sRes); ! VT=4, {"status" : "OK","title" : "Update Device"}
! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
    dl.Variable("Last update OK")
}
else {
    dl.Variable("Last update ERROR")
};
! WriteLine("VT="#dl.VarType()); ! VT=9
! WriteLine(dl.Variable()); ! shows the last update string
```

*Note*

The Domoticz HTTP response string is checked against error and a RaspMatic systemvar “DomoticzLog” is updated.

## Button pressed and Domoticz Device Updated

Channel	Room	Function	Last modified		IAQM Status	
Filter	Filter	Filter			Text	
HM-RC-19 CUX2801001:1  Push button channel			09.07.2019 10:21:57	  Short button press	09.07.2019 10:21:57 - Threshold reached! Take action... Last Seen: 2019-07-09 10:21:56 Type: General, Text  ★ Log Edit	

## Plugin Considerations

Initially started testing by adding HomematicIP devices to the CCU, determined the required device, channel and datapoint id's, triggered HTTP REST requests (XML-API) from Domoticz and parsed the CCU XML formatted HTTP response to newly created Domoticz devices as triggered by dzVents Lua script (timer).

These steps are defined next.

Then considered, that for additional devices, it requires effort to setup devices and dzVents Lua scripts ... which could get unclear or even confusing.

A better approach would be, to develop for each of the HomematicIP devices a Domoticz Python Plugin.

After various iterations, developed plugins for the HomematicIP devices Pluggable Switch and Meter (HMIP-PSM) and Radiator Thermostat (HMIP-eTRV-2).

These can be found in my GitHub [repositories](#) starting with domoticz-plugin-hmip-<short description>, i.e. domoticz-plugin-hmip-psm, domoticz-plugin-hmip-etrv-2.

More under consideration ...

## CCU Device(s) Information

### All Devices List

HTTP request to obtain the list of all devices.

```
http://ccu-ip/addons/xmlapi/devicelist.cgi
```

Example response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<deviceList>
  <device ready_config="true" device_type="HmIP-eTRV-2" interface="HmIP-RF" ise_id="1541"
address="000A18A9A64DAC" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ready_config="true" ise_id="1542" address="000A18A9A64DAC:0" name="HmIP-eTRV-2
000A18A9A64DAC:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
group_partner="" index="0" parent_device="1541" direction="UNKNOWN" type="30"/>
...
  </device>

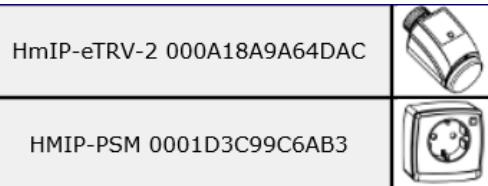
  <device ready_config="true" device_type="HMIP-PSM" interface="HmIP-RF" ise_id="1418"
address="0001D3C99C6AB3" name="HMIP-PSM 0001D3C99C6AB3">
    <channel ready_config="true" ise_id="1419" address="0001D3C99C6AB3:0" name="HMIP-PSM
0001D3C99C6AB3:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
...
...
```

#### Notes

The example response extract lists two devices

- HmIP-eTRV-2  
(device id=1541)
- HMIP-PSM  
(device id=1418)

with their device id and serial number (address).



### All Devices State Information

To get detailed information about the state of all the devices, submit

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="1011" name="HM-RCV-50 BidCoS-RF">
<device ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false" unreach="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0" operate="true" visible="true" index="0">
    <datapoint ise_id="1543" name="HmIP-RF.000A18A9A64DAC:0.CONFIG_PENDING" type="CONFIG_PENDING"
operations="5" timestamp="1561796713" valueunit="" valuetype="2" value="false"/>
...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
    <datapoint ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE" type="ACTIVE_PROFILE"
operations="7" timestamp="1561796714" valueunit="" valuetype="16" value="1"/>
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
type="ACTUAL_TEMPERATURE" operations="5" timestamp="1561796714" valueunit="" valuetype="4"
value="22.900000"/>
```

#### Notes

The response lists for each device the channels and the datapoints with unique ids.

## Single Device Information

Submit HTTP request to obtain detailed information for a single device, i.e. the thermostat.

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
<device ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false" unreach="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
    ...
    <datapoint ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" type="LOW_BAT"
      timestamp="1561799020" valueunit="" valuetype="2" value="false"/>
    ...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
    ...
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
      type="ACTUAL_TEMPERATURE" timestamp="1561799020" valueunit="" valuetype="4" value="22.900000"/>
    ...
    <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
      type="SET_POINT_TEMPERATURE" timestamp="1561799020" valueunit="°C" valuetype="4" value="4.500000"/>
    ...
  </channel>
  ...
</device>
</state>
```

### Notes

The response lists all the device datapoints, which can be used to obtain information or change a value via HTTP URL XML-API request.

These requests are used by the Domoticz dzVents Lua scripts.

To test, submit the HTTP URL and check the response on a browser, but also in the Homematic Web-UI.

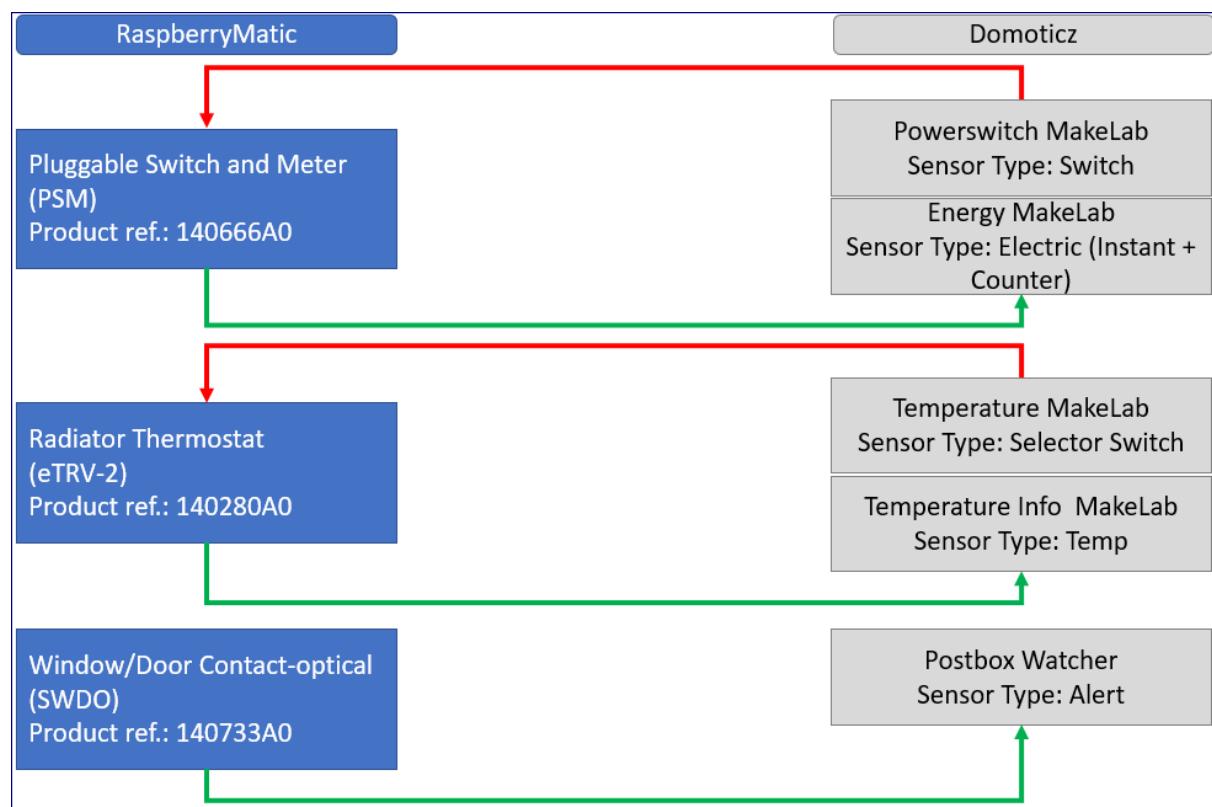
## CCU Devices HomematicIP

To get started with HomematicIP (controlled by RaspberryMatic via Domoticz), these three devices are used – for details see the respective functions.

### List

Device	Function	Notes
Pluggable Switch and Meter (PSM)  Product ref.: 140666A0 Device ID: 1418	<ul style="list-style-type: none"> <li>• Room MakeLab</li> <li>• Switch main devices</li> <li>• Measure energy consumption (Wh) &amp; power (W)</li> </ul>	Max. switching capacity 3680 W
Radiator Thermostat (eTRV-2)  Product ref.: 140280A0 Device ID: 1541	<ul style="list-style-type: none"> <li>• Room MakeLab</li> <li>• Control room temperature</li> <li>• Measure room temperature</li> </ul>	
Window / Door Contact – optical (SWDO)  Product ref.: 140733A0 Device ID: 2530	<ul style="list-style-type: none"> <li>• Postbox Watcher</li> <li>• Notify if the hatch of the postbox has been opened</li> </ul>	

### Communication Devices





# Experiments

Next are just a few experiments briefly described, to test connecting and using devices.

## HMIP-PSM

To switch the power on/off for all devices and measure the Power (W) and Energy Consumption (Wh) of the room MakeLab.

A HomematicIP Pluggable Switch and Meter (HmIP-PSM, Device ID: 1418) connected to a RaspberryMatic operating system running a Homematic Central-Control-Unit (CCU).

Domoticz requests, in regular intervals, data from the CCU via a dzVents Lua script and updates Domoticz device information.

## RaspberryMatic

RaspberryMatic Device Configuration					Domoticz Device Configuration (Name/Type/SubType)
Channel	Room	Function	Last modified	Control	Device
					Powerswitch MakeLab
HmIP-PSM 0001D3C9C9CAB3:3 Switch actuator			30.06.2019 09:10:57	Off      On	Light/switch Switch
HmIP-PSM 0001D3C9C9CAB3:6 Status report measured value channel	Home office		30.06.2019 09:10:57	Energy counter RaspMatic 716.60 Wh  Energy counter device 0.00 Wh  Voltage 230.60 V  Power 0.00 W  Frequency 50.01 Hz	Electric Usage MakeLab General kWh

## Domoticz

### Automation Scripts

Event name: powerswitch\_makelab

```
local IDX_POWERSWITCH_MAKELAB = 177;
local ID_DATAPOINT_STATE = 1451
local URL_CCU = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_STATE .. '&new_value='
local RES_CCU = 'powerswitchmakelab';
return {
    on = {
        devices = {IDX_POWERSWITCH_MAKELAB},
        httpResponses = {RES_CCU}
    },
    execute = function(domoticz, item)
        if (item.isDevice) then
            local state = 'true';
            if domoticz.devices(IDX_POWERSWITCH_MAKELAB).state == 'Off' then
                state = 'false';
            end
            domoticz.openURL({ url = URL_CCU .. state, method = 'GET', callback = RES_CCU });
        end;
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                domoticz.log('[INFO] Switch : ' .. item.statusText)
            else
                domoticz.log('[ERROR] Cannot switch : ' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

**Event name: electric\_usage\_makelab**

```
local IDX_ELECTRICUSAGEMAKELAB = 174;
local ID_DEVICE = 1418;
local URL_CCU = 'http://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
local RES_CCU = 'electricusagemakelab';
local DECIMALS = 2;

return {
  on = {
    timer = {'every minute'},
    httpResponses = {RES_CCU}
  },
  execute = function(domoticz, item)
    if (item.isTimer) then
      domoticz.openURL({ url = URL_CCU, method = 'GET', callback = RES_CCU })
    end
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        local powervalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1471"]/@value') -- W
        local energyvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1467"]/@value') -- Wh
        domoticz.devices(IDX_ELECTRICUSAGEMAKELAB).updateElectricity(
          domoticz.helpers.roundnumber(tonumber(powervalue),2), -- W
          domoticz.helpers.roundnumber(tonumber(energyvalue),2) ) -- Wh
      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

## HMIP-eTRV-2

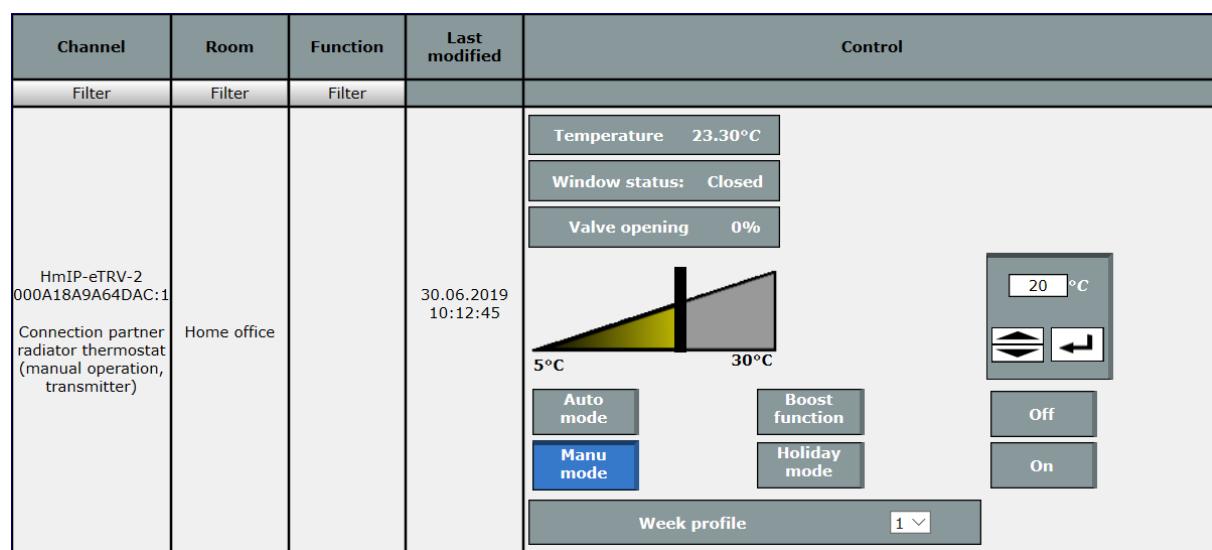
To measure and control the MakeLab room temperature.

A HomematicIP Radiator Thermostat (HmIP-eTRV-2, Device ID: 1541) connected to a RaspberryMatic operating system running a Homematic Central-Control-Unit (CCU). The thermostat setpoint is set by a Domoticz Selector Switch.

Domoticz requests, in regular intervals, the setpoint and room temperature from the CCU via a dzVents Lua script and updates Domoticz device information.

## RaspberryMatic

### Device Configuration



## Domoticz

### Domoticz Device Configuration

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
176	VirtualSensors	00014100	1	Thermostat MakeLab Setpoint	Light/Switch	Switch	Off
175	VirtualSensors	00082175	1	Thermostat MakeLab Info	General	Text	Setpoint: 4.5, Temperature: 23.2

**Thermostat MakeLab Setpoint** 20

Last Seen: 2019-06-30 10:12:44  
Type: Light/Switch, Switch, Selector

Off 18 19 20 21

Log Edit Timers Notifications

**Thermostat MakeLab Info**

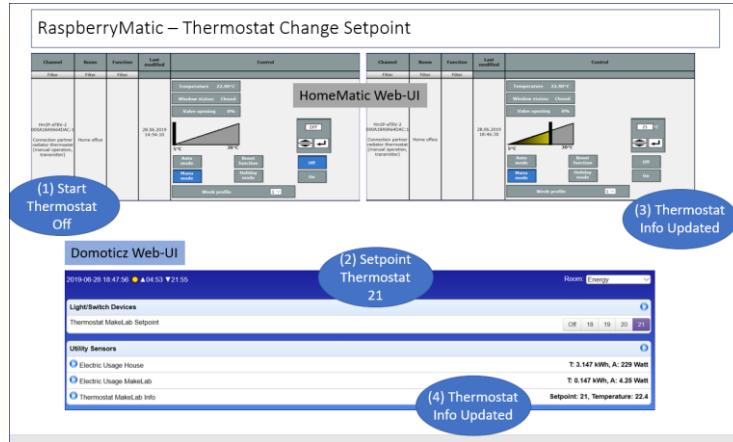
Setpoint: 20, Temperature: 23.3  
Last Seen: 2019-06-30 10:14:01  
Type: General, Text

Text

Log Edit

### Change Thermostat Setpoint

The setpoint of the thermostat is changed via dzVents Lua Script.



## Domoticz Automation Script

Event name: thermostat\_makelab\_setpoint

```

local IDX_THERMOSTAT_MAKELAB_SETPOINT = 176;
local ID_DATAPOINT_SETPOINT = 1584;
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_SETPOINT
.. '&new_value='
local RES_RASPMATIC = 'thermostatmakelabsetpoint';
local DECIMALS = 2;

return {
on = {
    devices = {IDX_THERMOSTAT_MAKELAB_SETPOINT},
    httpResponses = {RES_RASPMATIC}
},
execute = function(domoticz, item)
    if (item.isDevice) then
        -- get the new SETPOINT 0%=0(OFF); 10%-40%=18-21; min=18, max=21
        local level = domoticz.devices(IDX_THERMOSTAT_MAKELAB_SETPOINT).level;
        local newsetpoint = 0;
        if level > 0 then
            newsetpoint = 17 + (level * 0.1);
        end
        -- set the new setpoint
        domoticz.log('New setpoint:' .. URL_RASPMATIC .. tostring(newsetpoint), domoticz.LOG_INFO);
        domoticz.openURL({url = URL_RASPMATIC .. tostring(newsetpoint), method = 'POST', callback =
RES_RASPMATIC});
        end

        -- check if the item is a httpresponse from the openurl callback
        if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
                -- multile datapoints
                domoticz.log(item.data);
                -- parse the response using XPath
                -- select the attribute value of the datapoint element (this is XPath syntax)
                -- domoticz.log(domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value'))
            else
                domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
}

```

## Node-RED Dashboard Thermostat

State upfront: Domoticz is not involved in this experiment. There is direct communication between RaspberryMatic and Node-RED.

To create a Node-RED dashboard showing thermostat datapoints

- ACTUAL\_TEMPERATURE – measured at the thermostat (°C)
- SET\_POINT\_TEMPERATURE – at the thermostat (°C)
- LEVEL - the valve position (%)

The Node-RED dashboard add-on enables to create live dashboards (see [Node-RED](#)).

This experiment is developed on the Domoticz development system running Node-RED.

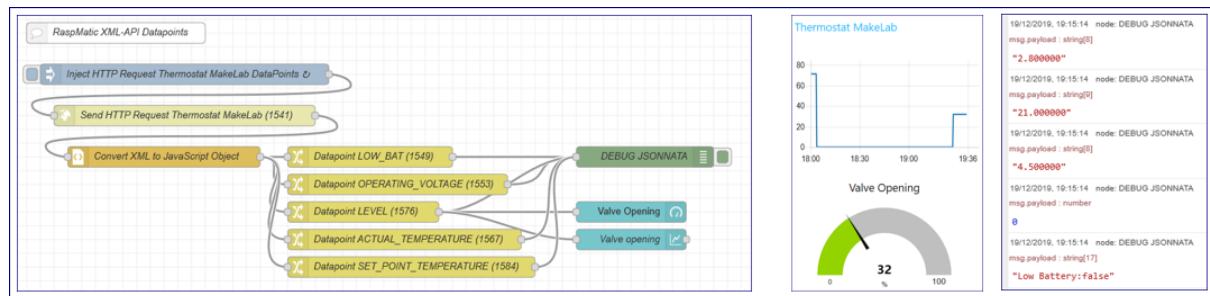
As already mentioned: a Domoticz system is not required.

Node-RED communicates directly with the RaspberryMatic system to obtain the value of the datapoints by triggering HTTP XML-API requests (using the state.cgi script).

As a first test, several datapoints of the thermostat MakeLab are selected.

### Node-RED

The picture below shows the Node-RED flow, a simple dashboard with chart & gauge and the debug log.



#### Note

In a browser, enter URL <http://domoticz-ip:1880/ui> to view the Node-RED dashboard tab Homematic > Group Thermostat MakeLab.

### Flow

The **Inject node** triggers every 30s, the action for the **HTTP request node** to submit the URL

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

to get the state information for the device with datapoint id 1541.

The HTTP request returns XML structured string with all device datapoints.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<states>
  <><device config_pending="false" unreach="false" ise_id="1541" name="Thermostat MakeLab">
    <channel ise_id="1542" name="Thermostats MakeLab:0">
      <datapoint ise_id="1543" name="HmIP-RF.000A18A9A64DAC:0.CONFIG_PENDING" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
      <datapoint ise_id="1544" name="HmIP-RF.000A18A9A64DAC:0.DUTY_CYCLE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
      <datapoint ise_id="1545" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" timestamp="1576843927" valueunit="" valuetype="2" value="LOW_BAT" type="OPERATING_VOLTAGE_STATUS"/>
      <datapoint ise_id="1546" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
      <datapoint ise_id="1547" name="HmIP-RF.000A18A9A64DAC:0.RSSI_PEER" timestamp="1576843927" valueunit="" valuetype="3" value="0" type="RSSI_PEER"/>
      <datapoint ise_id="1548" name="HmIP-RF.000A18A9A64DAC:0.UNREACH" timestamp="1576843927" valueunit="" valuetype="8" value="179" type="UNREACH"/>
      <datapoint ise_id="1561" name="HmIP-RF.000A18A9A64DAC:0.UPDATE_PENDING" timestamp="1575413140" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
    </channel>
    <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
      <datapoint ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE" timestamp="1576843927" valueunit="" valuetype="16" value="1" type="ACTIVE_PROFILE"/>
      <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927" valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
      <datapoint ise_id="1568" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
      <datapoint ise_id="1569" name="HmIP-RF.000A18A9A64DAC:1.BOOT_MODE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="BOOT_MODE"/>
      <datapoint ise_id="1570" name="HmIP-RF.000A18A9A64DAC:1.BOOST_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="BOOST_TIME"/>
      <datapoint ise_id="1571" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_DIFFERENTIAL_TEMPERATURE" timestamp="0" valueunit="" valuetype="4" value="0" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
      <datapoint ise_id="1572" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_MODE" timestamp="0" valueunit="" valuetype="16" value="0" type="CONTROL_MODE"/>
      <datapoint ise_id="1573" name="HmIP-RF.000A18A9A64DAC:1.DURATION_UNIT" timestamp="0" valueunit="" valuetype="16" value="0" type="DURATION_UNIT"/>
      <datapoint ise_id="1574" name="HmIP-RF.000A18A9A64DAC:1.FROST_PROTECTION" timestamp="0" valueunit="" valuetype="16" value="0" type="FROST_PROTECTION"/>
      <datapoint ise_id="1575" name="HmIP-RF.000A18A9A64DAC:1.FROST_PROTECTION" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="FROST_PROTECTION"/>
      <datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" timestamp="1576843927" valueunit="" valuetype="0.270000" type="LEVEL"/>
      <datapoint ise_id="1577" name="HmIP-RF.000A18A9A64DAC:1.LEVEL_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="LEVEL_STATUS"/>
      <datapoint ise_id="1578" name="HmIP-RF.000A18A9A64DAC:1.PARTY_MODE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="PARTY_MODE"/>
      <datapoint ise_id="1579" name="HmIP-RF.000A18A9A64DAC:1.PARTY_SET_POINT_TEMPERATURE" timestamp="0" valueunit="" valuetype="0.000000" type="PARTY_SET_POINT_TEMPERATURE"/>
      <datapoint ise_id="1580" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_END" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_END"/>
      <datapoint ise_id="1581" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_START" timestamp="0" valueunit="" valuetype="20" value="0" type="PARTY_TIME_START"/>
      <datapoint ise_id="1582" name="HmIP-RF.000A18A9A64DAC:1.QUICK_VETO_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="QUICK_VETO_TIME"/>
      <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SETPOINT_VETO_OCCURRED" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="SETPOINT_VETO_OCCURRED"/>
      <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" timestamp="1576843927" valueunit="" valuetype="4" value="0.000000" type="SET_POINT_TEMPERATURE"/>
      <datapoint ise_id="1585" name="HmIP-RF.000A18A9A64DAC:1.SWITCH_POINT_OCCURRED" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="SWITCH_POINT_OCCURRED"/>
      <datapoint ise_id="1586" name="HmIP-RF.000A18A9A64DAC:1.VALVE_ADAPTION" timestamp="0" valueunit="" valuetype="2" value="false" type="VALVE_ADAPTION"/>
      <datapoint ise_id="1587" name="HmIP-RF.000A18A9A64DAC:1.VALVE_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="VALVE_STATE"/>
      <datapoint ise_id="1588" name="HmIP-RF.000A18A9A64DAC:1.WINDOW_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="WINDOW_STATE"/>
    </channel>
    <channel ise_id="1589" name="HmIP-eTRV-2 000A18A9A64DAC:2">
    <channel ise_id="1590" name="HmIP-eTRV-2 000A18A9A64DAC:3">
    <channel ise_id="1591" name="HmIP-eTRV-2 000A18A9A64DAC:4">
    <channel ise_id="1592" name="HmIP-eTRV-2 000A18A9A64DAC:5">
    <channel ise_id="1593" name="HmIP-eTRV-2 000A18A9A64DAC:6">
  </channel>
  <channel ise_id="1594" name="HmIP-eTRV-2 000A18A9A64DAC:7">
</states>
```

### Note

All datapoints with the example datapoint ACTUAL\_TEMPERATURE highlighted as used in the example next

The **XML node** converts the XML structure to a Javascript object which is used by the various **Change nodes** to get the value of a specific datapoint by using JSONata expressions. JSONata makes it simple to parsing this kind of data.

### Example JSONata expressions

Get the ACTUAL\_TEMPERATURE value for datapoint 1567.  
The payload returned is a string.

```
payload.**[ise_id="1567"].value
```

### Notes

The multi-level wildcard \*\* traverses all descendants. In this case, select all “ise\_id equals 1567” values, regardless of how deeply nested the information is in the XML structure.

The XML structure snippet parsed:

```
...
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927"
valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
...
```

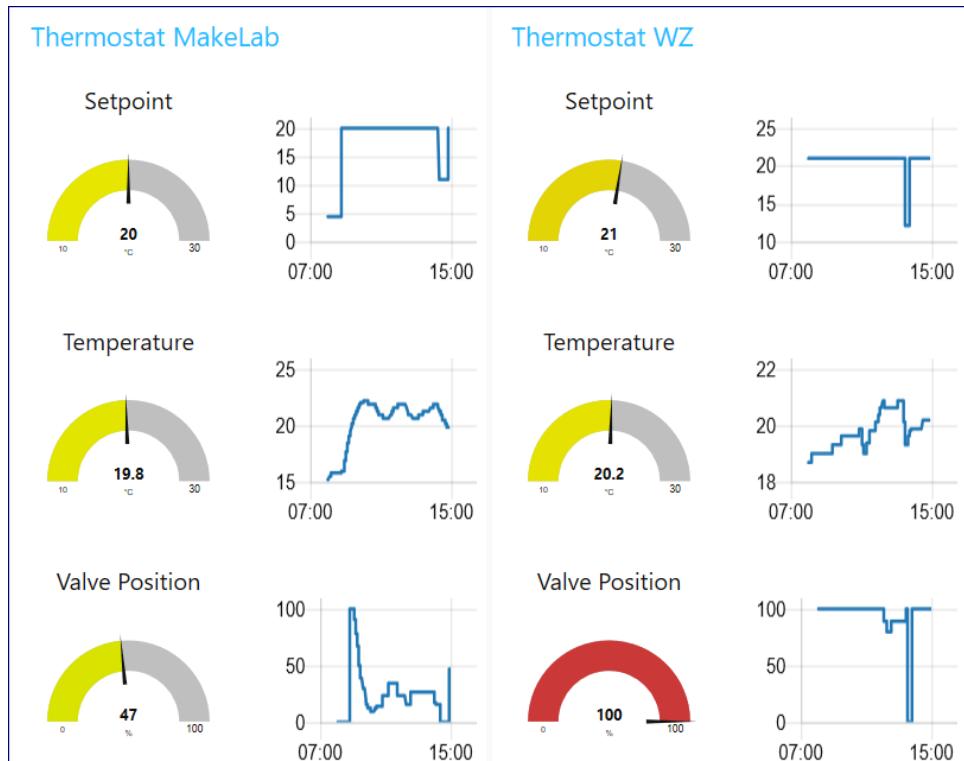
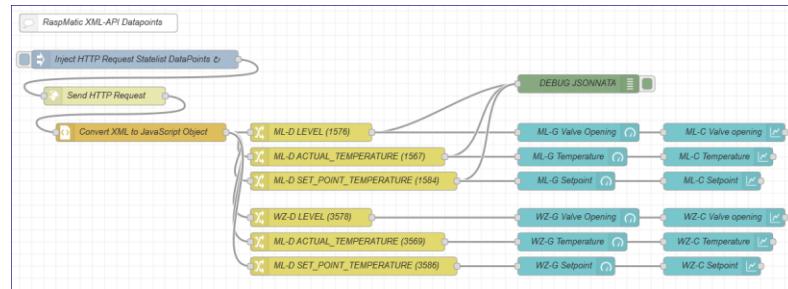
JSONata enables also to change the value, like for example the LEVEL value (=the valve position) from 0-1 to 0-100 (displayed with unit %).

The payload returned is a number.

```
$number(payload.**[ise_id=\\"1576\\"].value) * 100
```

The value is used by Node-Red Dashboard nodes like the Gauge (ui\_gauge) or Chart (ui\_chart).

The next flow has the two thermostats MakeLab and LivingRoom based on previous flow explained.



## RaspberryMatic triggers Node-RED flow (Push)

State upfront: Domoticz is not involved in this experiment, i.e. direct communication between RaspberryMatic and Node-RED.

To handle the state change of a window and door contact with magnet device (HmIP-SWDM).

Every state change is handled by a RaspberryMatic script which triggers a Node-RED HTTP GET request (RaspberryMatic pushes the state information the Node-RED flow).

The HTTP GET request parameter are used by Node-RED to take action accordingly.

### Raspberry Matic

The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE

If the SWDM device state changes, an HTTP request is sent (pushed) to Node-RED.

Example:

```
http://domoticz-ip:1880/3597?dp=3622&state=1
```

The parameters are the device id SWDM, datapoint id STATE and current state (0=closed, 1=open).

The request is handled by Node-RED.

### RaspberryMatic Script

The script is triggered if the state of the device changes either open or close.

The screenshot shows the RaspberryMatic programming interface. The top navigation bar includes 'Admin', 'Home page > Programs and connections > Programs > Programming', 'Logout', 'Alarm messages (0)', 'Service messages (0)', 'Teach-in devices', and 'Help'. Below the navigation is a toolbar with buttons for 'Home page', 'Status and control', 'Programs and connections', and 'Settings'. The main area displays a table for a script named 'Eingang Monitor'. The table columns are 'Name', 'Description', 'Condition (If...)', 'Activity (then..., or else...)', and 'Action'. The 'Name' row contains 'Eingang Monitor'. The 'Description' row contains 'Monitor Eingangstür auf geschlossen oder geöffnet'. The 'Condition (If...)' row contains 'Channel status: HmIP-SWDM 001558A99D5A78:1 when open trigger when changed' and 'Script: ... immediately run'. The 'Action' row has an unchecked checkbox for 'intrinsic'. Below the table, there are sections for 'Condition: If...' and 'Activity: Then...'. The 'Condition: If...' section shows two AND conditions: 'Device selection HmIP-SWDM 001558A99D5A78:1 when open trigger when changed' and 'Device selection HmIP-SWDM 001558A99D5A78:1 when closed trigger when changed'. The 'Activity: Then...' section shows an 'Activity' dropdown set to 'Script' and a 'Function' dropdown set to '! Function: Door Monitor ! 20191226 by rwbl string sFunction... immediately'. There is also an 'Else...' section.

### Script Source Code

```
string sFunction = "Eingang Monitor";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! The object string is taken from the raspmatic XML-API script statelist.cgi
string sState = dom.GetObject("HmIP-RF.001558A99D5A78:1.STATE").State();
! Node-RED Test
string cQMark = "?";
string cAmp = "&";
! XMLAPI datapoints Device & STATE
string sIDDevice = "3597";
string sIDState = "3622";
! Datapoint STATE property
string sUrlNR = "'http://domoticz-ip:1880/'; ! Development
string sUrl = sUrlNR#sIDDevice#cQMark#"dp="#sIDState#cAmp#"state="#sState#"";;
WriteLine(sUrl);
```

```

! Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

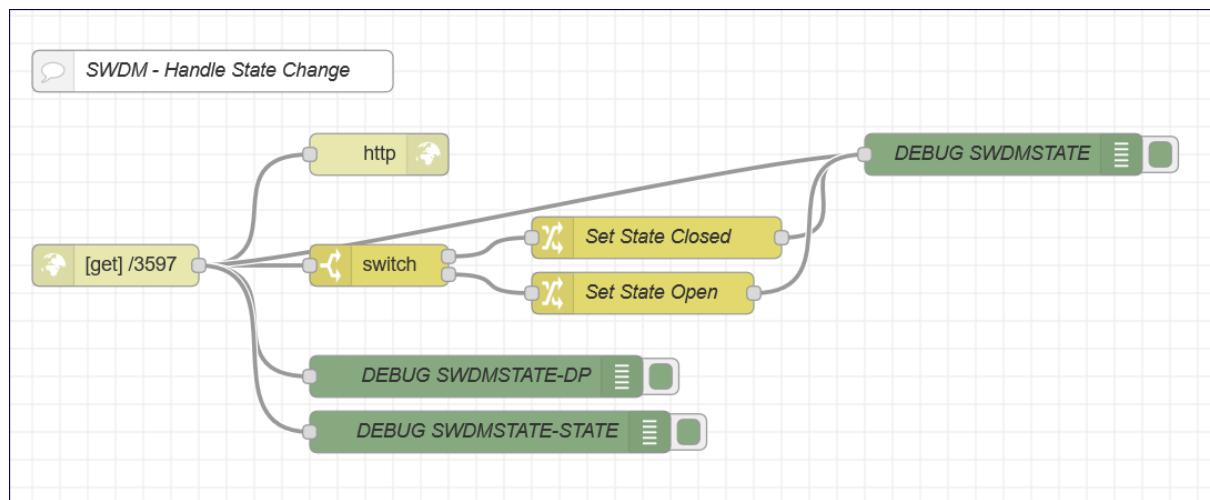
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command
! NOTE: script running on CCU waits until completion - ensure not to execute commands which take long
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RETs").State();
! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
    dl.Variable(sFunction#"-Last update OK:#sDate#" "#sTime")
}
else {
    dl.Variable(sFunction#"Last update ERROR:#sDate#" "#sTime")
};
! WriteLine("VT="#dl.VarType()); ! VT=9
! WriteLine(dl.Variable()); ! shows the last update string

```

## Node-RED

### Flow



### Flow Description

Node	Comments
<b>http in</b>	<p>Method: GET URL: /3597</p> <p>Output to nodes: http response, switch, various debug</p> <p><i>Notes</i></p> <p>The output of the http in node is a JSON string containing the parameter. Example: { "dp": "3622", "state": "1" } from the URL <a href="http://domoticz-ip:1880/3597?dp=3622&amp;state=1">http://domoticz-ip:1880/3597?dp=3622&amp;state=1</a> with device id 3597 and datapoint 3622.</p>
<b>http response</b>	No changes made, kept the defaults
<b>switch</b>	Property: msg.req.query.state == a/z: 0 == a/z: 1

	<p>Output to nodes: switch, debug There are two outputs covering 0 and 1</p> <p>Notes: The property "state" from the URL query is used with value 0   1 (req.query.state). The property "dp" has value "3622 (req.query.dp).</p>
<b>change</b>	There are two change nodes handling 0 and 1. For 0, the value is to the string Closed. For 1, the value is to the string Open.
<b>debug</b>	Outputs the value of various nodes, i.e. http in, change

### Node-RED Debug log from testing the RaspberryMatic script

The script sends HTTP request: <http://domoticz-ip:1880/3597?dp=3622&state=1>

```

debug
  i  f  l  v
  all nodes  delete

13/01/2020, 11:32:58 node: DEBUG SWDMSTATE
msg.payload : Object
▶ { dp: "3622", state: "1" }

13/01/2020, 11:32:58 node: DEBUG SWDMSTATE-
DP
msg.req.query.dp : string[4]
"3622"

13/01/2020, 11:32:59 node: DEBUG SWDMSTATE-
STATE
msg.req.query.state : string[1]
"1"

13/01/2020, 11:33:00 node: DEBUG SWDMSTATE
msg.payload : string[4]
"Open"

```

## Node-RED triggers HTTP GET Request (Pull)

State upfront: Domoticz is not involved in this experiment, i.e. direct communication between RaspberryMatic and Node-RED.

To request the state change of a window and door contact with magnet device (HmIP-SWDM).

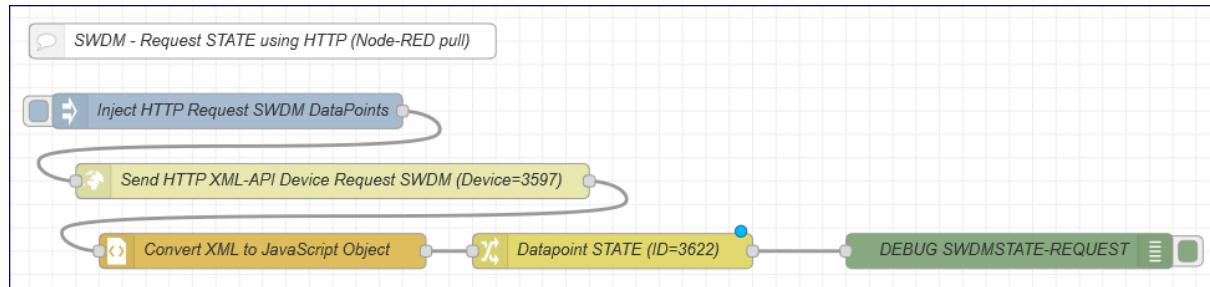
The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE

### Node-RED

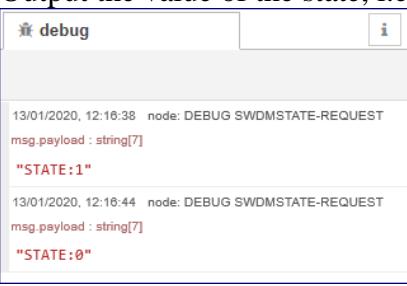
Node-RED to pull from RaspberryMatic the state of the SWDM device.

This is an example how to trigger, by Node-RED, the value of a datapoint via the XML-API.

### Flow



### Flow Description

Node	Comments
<b>Inject</b>	Start the request. An interval can be set to regular request device information. For this example, does not make sense as the state can be changed within an interval.
<b>http request</b>	Send the HTTP request using the defined URL. Example: http://ccu-ip/config/xmlapi/state.cgi?device_id=3597
<b>Json</b>	Convert the HTTP XML response to a JavaScript object.
<b>change</b>	Get the value from the XML tree using JSONata & xpath: \$join(["STATE:", msg.payload.**[ise_id="3622"].value])
<b>Debug</b>	Output the value of the state, i.e. 0   1 

## Node-RED triggers HTTP GET Request All Datapoints

This concept is based on the tool [RaspberryMatic Statelist](#) and is enhanced further to:

- get, in regular intervals, using HTTP XML-API request (i.e. `http://ccu-ip/config/xmlapi/statelist.cgi`),
- a device datapoint value and
- update the Domoticz device via an HTTP API request  
`http://domoticz-ip:port/json.htm?type=command&param=udevice&idx=113&nvalue=0&svalue=24.1`

Instead of setting a single device, there is also an example how to set multiple devices from multiple datapoints.

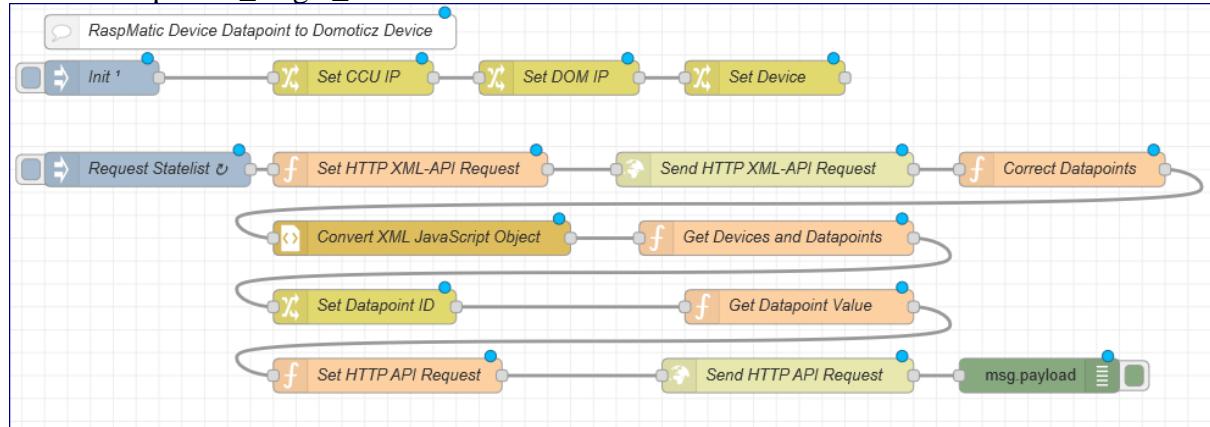
### Example Single Device

Get the RaspberryMatic datapoint 1567 (Thermostat MakeLab, Type ACTUAL\_TEMPERATURE, Value °C) and assign to the Domoticz device with IDX=113.

	<b>IDX=113</b> Hardware=VirtualDevices, Name=MakeLab Temperature Type=Temp SubType=LaCrosse TX3
--	---

### Node-RED Flow

Source: `raspmatic_single_device.flow`



### Function Node Get Devices and Datapoints

The purpose of this function is to

- return a json array with devices used for the ui\_dropdown
- set flow context "datapoints" for the ui\_template to display the datapoints for the selected device id

#### Notes

- Example device entry: `deviceobj["Briefkasten Status"] = 2530;`
- Example datapoint entry:  
`device:"2530",datapoint:{ "id":"2532","type":"CONFIG_PENDING","value":false,"name":"HmIP-RF.0000DA498D5859:0.CONFIG_PENDING"}`

- the second array below is not used as replaced by the flow context "datapoints" - but is kept in case any enhancements planned

```

statelist = msg.payload.stateList;
devices = statelist.device
arrdevices = [];
arrdatapoints = [];
devices.forEach(function(device) {
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename]=deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            channeldatapoints.forEach(function(datapoint){
                obj = datapoint.$;
                if (obj !== undefined) {
                    dpobj = {
                        "device":deviceise_id,
                        "datapoint": {
                            "id":obj.ise_id,"type":obj.type,"value":obj.value,"name":obj.name
                        }
                    };
                    arrdatapoints.push(dpobj);
                }
            });
        }
    });
});
flow.set("datapoints",arrdatapoints);

msgdevices = {};
msgdevices.options = arrdevices;

// not used
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices,msgdatapoints];

```

## Function Node Get Datapoint Value

Get the value from the flow context “datapoints” using the datapoint id and returns the value property.

```

// get the datapoint id, i.e. 1657 for the selected channel
var datapointid = msg.payload;
// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":false,"name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");
datapoints.forEach(function(datapoint){
    if (datapoint.datapoint.id == datapointid) {
        msg.payload = datapoint.datapoint.value;
    }
});
return msg;

```

### Function Node Set HTTP API request

Set the msg.url to request Domoticz to update the device with IDX=113.  
The flow context “domip” holds the Domoticz system IP address.

```
const temp = msg.payload;
const idx = flow.get("device").idx;

// Define the url to set the temperature of the Domoticz device
// /json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP

msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx=" + idx +
"&nvalue=0&svalue=" + temp;
node.warn(msg.url);

return msg;
```

## Example Multiple Devices

To get & set the value for multiple Domoticz devices a JSON array is defined in the Node-RED flow change node “Set Devices”. This array is used by the function node “Get Devices and Set Value”.

For this example, get three RaspMatic datapoints and set the value of three Domoticz devices.

Extract from the RaspMatic full statelist obtained using HTTP XML-API request:

```
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1583752872" valueunit="" valuetype="4" value="22.800000" type="ACTUAL_TEMPERATURE"/>

<datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" operations="7" timestamp="1583752872" valueunit="" valuetype="4" value="0.310000" type="LEVEL"/>

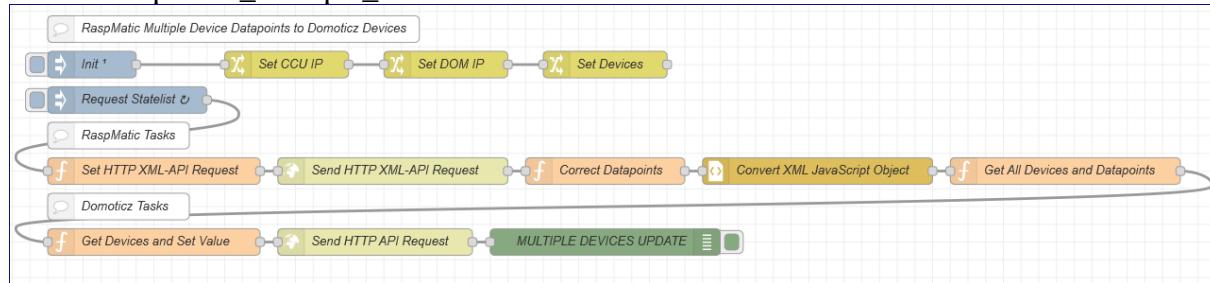
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1583752872" valueunit="°C" valuetype="4" value="21.000000" type="SET_POINT_TEMPERATURE"/>
```

The Node-RED change node to configure the datapoints in JSON format (array with several key:value pairs) and the three Domoticz devices (from the GUI > Setup > Devices list).

<div style="border: 1px solid #ccc; padding: 5px;"> <p>Edit change node &gt; JSON editor</p> <p>Edit JSON      Visual editor</p> <pre>1 * [ 2 *   { 3 *     "datapoint": 1567, 4 *     "type": "ACTUAL_TEMPERATURE", 5 *     "idx": 113 6 *   }, 7 *   { 8 *     "datapoint": 1576, 9 *     "type": "LEVEL", 10 *    "idx": 114 11 *  }, 12 *  { 13 *    "datapoint": 1584, 14 *    "type": "SET_POINT_TEMPERATURE", 15 *    "idx": 115 16 *  } 17 * ]</pre> </div>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Idx</th> <th>Hardware</th> <th>ID</th> <th>Unit</th> <th>Name</th> <th>Type</th> <th>SubType</th> </tr> </thead> <tbody> <tr> <td>115</td> <td>VirtualDevices</td> <td>140C3</td> <td>1</td> <td>MakeLab Thermostat Setpoint</td> <td>Temp</td> <td>LaCrosse TX3 6.0 C</td> </tr> <tr> <td>114</td> <td>VirtualDevices</td> <td>00082114</td> <td>1</td> <td>MakeLab Thermostat Valve</td> <td>General</td> <td>Percentage 0%</td> </tr> <tr> <td>113</td> <td>VirtualDevices</td> <td>140C1</td> <td>1</td> <td>MakeLab Thermostat Temperature</td> <td>Temp</td> <td>LaCrosse TX3 21.8 C</td> </tr> </tbody> </table> <p>RaspMatic Datapoint = Domoticz Idx - see left.</p>	Idx	Hardware	ID	Unit	Name	Type	SubType	115	VirtualDevices	140C3	1	MakeLab Thermostat Setpoint	Temp	LaCrosse TX3 6.0 C	114	VirtualDevices	00082114	1	MakeLab Thermostat Valve	General	Percentage 0%	113	VirtualDevices	140C1	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3 21.8 C
Idx	Hardware	ID	Unit	Name	Type	SubType																							
115	VirtualDevices	140C3	1	MakeLab Thermostat Setpoint	Temp	LaCrosse TX3 6.0 C																							
114	VirtualDevices	00082114	1	MakeLab Thermostat Valve	General	Percentage 0%																							
113	VirtualDevices	140C1	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3 21.8 C																							

## Node-RED Flow

Source: raspmatic\_multiple\_devices.flow



## Function Node Get Devices and Datapoints

Same as for the single device.

## Function Node Get Devices and Set Value

```

/*
The purpose of this function is to set the value for multiple domoticz devices.
This function does not return a value as msg are send during device json array loop.
*/
const DEBUG = true;
// get all the datapoints by device
// device:"2530",datapoint:{id:"2532",type:"CONFIG_PENDING",value:"false",name:"HmIP-RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");
// devices: [{"datapoint": 1567,type: "ACTUAL_TEMPERATURE",idx: 113}, ...]
devices = flow.get("devices");
//
devices.forEach(function(device){
    datapoints.forEach(function(datapoint){
        if (device.datapoint == datapoint.datapoint.id) {
            idx = device.idx;
            value = datapoint.datapoint.value;
            if (idx == "114") value = value * 100;
            msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx=" +
+ idx + "&value=0&value=" + value;
            node.send(msg);
            if (DEBUG) node.warn("URL: " + msg.url);
        }
    });
});
}
  
```

# SQL

## Purpose

To execute CRUD operations on the Domoticz SQLite Database domoticz.db.

CRUD = Create (Create, Insert), Read (Select), Update and Delete.

The operations are performed via SQL statements:

- A statement is any text that the database engine recognizes as a valid command.
- A query is a statement that returns a recordset or nothing if empty.

## Thoughts on using SQL

- Python3 scripts triggered by a Switch or Node-RED Dashboard UI (and input field)
- Python3 with guizero – simple UI
- Node-RED SQLite3 Nodes
- SQLite3 package from a terminal
- B4J application

## Approach

- **Explore Database Structure & SQL Statements**

To analyse the database, external tools like the SQLiteSpy are useful helpers, but also using SQLite3 from the terminal command-line (Command Line Interface CLI).

- **For Tests**

The Domoticz database is exported from the Domoticz production system to the Development Device:

GUI > Setup > Settings > Backup/Restore > Backup Database.

The CRUD operations are there for executed in offline mode.

- **For Production**

Ensure Domoticz is stopped first, then perform CRUD operations on the Domoticz database and start Domoticz after completing.

This is only required if changes (Update or Delete) to the database are made. For Read (Select), no need to stop Domoticz.

## SQLite Shell

The sqlite3 package (include SQLite libraries) enables to perform SQL operations from the command-line shell - SQLite shell or SQLite CLI (Command-Line-Interface) Shell.

### Install SQLite3 Package

Login as user “pi” and install the “sqlite3” package.

*Note*

Prior installing the sqlite3 package, run an update (i.e. on a Raspberry Pi):

```
sudo apt-get update  
sudo apt install sqlite3
```

```
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Suggested packages:  
  sqlite3-doc  
The following NEW packages will be installed:  
  sqlite3  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 709 kB of archives.  
After this operation, 1,991 kB of additional disk space will be used.  
Get:1 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf sqlite3 armhf 3.16.2-5+deb9u1  
[709 kB]  
Fetched 709 kB in 2s (325 kB/s)  
Selecting previously unselected package sqlite3.  
(Reading database ... 96101 files and directories currently installed.)  
Preparing to unpack .../sqlite3_3.16.2-5+deb9u1_armhf.deb ...  
Unpacking sqlite3 (3.16.2-5+deb9u1) ...  
Setting up sqlite3 (3.16.2-5+deb9u1) ...  
Processing triggers for man-db (2.7.6.1-2) ...
```

### Prepare SQL Statements

For the examples, the Domoticz production database “domoticz.db”, located on the Raspberry Pi in folder “/home/pi/domoticz”, is used.

Login as user “pi”, change directory to “domoticz”, start sqlite3 and connect to the database domoticz.db.

```
Login username "pi"  
$cd domoticz  
$sqlite3  
SQLITE version 3.16.2 2017-01-06 16:32:41  
Enter ".help" for usage hints.  
Connected to a transient in-memory database.  
Use ".open FILENAME" to reopen on a persistent database.  
sqlite> .open domoticz.db
```

#### Notes

Usage of the semicolon “;” in SQL statements.

- On dot “.” commands, to control the SQLite shell, do NOT use the semicolon to close the command off. This gives an error or non-expected results.  
Example: “.open domoticz.db;” creates a new database with filename “domoticz.db;”.
- Use the semicolon on SQL statements, i.e. SELECT, UPDATE etc.

## SQL Statement Examples

Some examples SQL statement. See also in several function, like function Notes section [Database Records](#).

### Get All Tables

```
sqlite> .tables
BackupLog          Meter           SceneTimers
Cameras            Meter_Calendar  Scenes
CamerasActiveDevices MobileDevices SetpointTimers
CustomImages       MultiMeter      SharedDevices
DeviceStatus       MultiMeter_Calendar Temperature
DeviceToPlansMap   MySensors      Temperature_Calendar
EnoceanSensors    MySensorsChilds TimerPlans
EventMaster        MySensorsVars  Timers
EventRules         Notifications  ToonDevices
Fan                Percentage    UV
Fan_Calendar      Percentage_Calendar UV_Calendar
FibaroLink         Plans          UserSessions
Floorplans         Preferences   UserVariables
GooglePubSubLink  PushLink      Users
Hardware           Rain           WOLNodes
HttpLink           Rain_Calendar Wind
LightSubDevices   SceneDevices  Wind_Calendar
LightingLog        SceneLog      ZWaveNodes
```

#### *Note*

The various tables holding measurement data are

```
'MultiMeter','MultiMeter_Calendar','Percentage','Percentage_Calendar','Rain','Rain_Calendar','Temperature','UV','UV_Calendar','Wind','Wind_Calendar'
```

### Get Structure DeviceStatus

There are several ways to get the structure of a table.

#### Command: .schema

```
sqlite> .schema devicestatus
CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT NULL, [DeviceID]
VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(100) DEFAULT Unknown, [Used] INTEGER
DEFAULT 0, [Type] INTEGER NOT NULL, [SubType] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0,
[Favorite] INTEGER DEFAULT 0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0,
[nValue] INTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFAULT
(datetime('now','localtime')), [Order] INTEGER BIGINT(10) default 0, [AddjValue] FLOAT DEFAULT 0,
[AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0, [AddjMulti2] FLOAT DEFAULT 1, [StrParam1]
VARCHAR(200) DEFAULT '', [StrParam2] VARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0,
[Protected] INTEGER DEFAULT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '',
[Options] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL);
CREATE INDEX ds_hduts_idx    on DeviceStatus(HardwareID, DeviceID, Unit, Type, SubType);
CREATE TRIGGER devicestatusupdate AFTER INSERT ON DeviceStatus
BEGIN
    UPDATE DeviceStatus SET [Order] = (SELECT MAX([Order]) FROM DeviceStatus)+1 WHERE DeviceStatus.ID =
NEW.ID;
END;
CREATE INDEX ds_hduts_idx    on DeviceStatus(HardwareID, DeviceID, Unit, Type, SubType);
CREATE TRIGGER devicestatusupdate AFTER INSERT ON DeviceStatus
BEGIN
    UPDATE DeviceStatus SET [Order] = (SELECT MAX([Order]) FROM DeviceStatus)+1 WHERE DeviceStatus.ID =
NEW.ID;
END;
```

## Command: Pragma

```
sqlite> .header on
sqlite> .mode column
sqlite> pragma table_info('devicestatus');

cid      name        type       notnull    dflt_value   pk
-----  -----
0        ID          INTEGER    0           1
1        HardwareID  INTEGER    1           0
2        DeviceID    VARCHAR(25) 1           0
3        Unit         INTEGER    0           0
4        Name         VARCHAR(10) 0           Unknown
5        Used         INTEGER    0           0
6        Type         INTEGER    1           0
7        SubType      INTEGER    1           0
8        SwitchType   INTEGER    0           0
9        Favorite     INTEGER    0           0
10       SignalLeve  INTEGER    0           0
11       BatteryLev  INTEGER    0           0
12       nValue       INTEGER    0           0
13       sValue       VARCHAR(20) 0           null
14       LastUpdate   DATETIME   0           datetime('0
15       Order        INTEGER BI 0           0
16       AddjValue    FLOAT      0           0
17       AddjMulti    FLOAT      0           1
18       AddjValue2   FLOAT      0           0
19       AddjMulti2   FLOAT      0           1
20       StrParam1   VARCHAR(20) 0           ''
21       StrParam2   VARCHAR(20) 0           ''
22       LastLevel   INTEGER    0           0
23       Protected    INTEGER    0           0
24       CustomImage INTEGER    0           0
25       Descriptio  VARCHAR(20) 0           ''
26       Options      TEXT       0           null
27       Color        TEXT       0           NULL
```

## Command: Select Query

```
sqlite> SELECT sql FROM sqlite_master WHERE name = "DeviceStatus";
sql
-----
CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT
NULL, [DeviceID] VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(
100) DEFAULT Unknown, [Used] INTEGER DEFAULT 0, [Type] INTEGER NOT NULL, [SubTyp
e] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0, [Favorite] INTEGER DEFAULT
0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0, [nValue] I
NTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFA
ULT (datetime('now','localtime')), [Order] INTEGER BIGINT(10) default 0, [AddjVal
ue] FLOAT DEFAULT 0, [AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0,
[AddjMulti2] FLOAT DEFAULT 1, [StrParam1] VARCHAR(200) DEFAULT '', [StrParam2] V
ARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0, [Protected] INTEGER DEFAU
LT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '', [O
ptions] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL)
```

## Other Examples

```
sqlite> .schema Temperature
CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill]
FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT
0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));
CREATE INDEX t_id_idx      on Temperature(DeviceRowID);
CREATE INDEX t_id_date_idx on Temperature(DeviceRowID, Date);
```

```
sqlite> .schema LightingLog
```

```
CREATE TABLE [LightingLog] ([DeviceRowID] BIGINT(10) NOT NULL, [nValue] INTEGER DEFAULT 0, [sValue]
VARCHAR(200), [Date] DATETIME DEFAULT (datetime('now','localtime')), [User] VARCHAR(100) DEFAULT (''));
CREATE INDEX ll_id_idx      on LightingLog(DeviceRowID);
CREATE INDEX ll_id_date_idx on LightingLog(DeviceRowID, Date);
```

## Select DeviceStatus ID,Name

```
sqlite> select ID, Name from DeviceStatus;
1|RPi RAM Usage
2|RPi HDD /boot
... and many more
12|Blind Living Room
207|Hzg Bad Sollwert
208|Hzg Bad Temperatur
209|Hzg Bad Batterie
...
```

## Select DeviceStatus ID,Name,nValue,sValue,LastUpdate

For the device with id=201. The ID is the Idx of the device.

```
sqlite> select ID,Name,nValue,sValue,LastUpdate from DeviceStatus where ID=201;
201|Hzg WZ Sollwert|1|21.0|2019-12-17 06:15:02
```

## Select LightingLog Device IDX=161

```
sqlite> select * from lightinglog where devicerowid=161;
161|0|Security Main Door changed to Alarm + Tamper|2019-05-21 09:01:07|
161|0|Security Main Door changed to Normal|2019-05-21 09:05:00|
161|0|Hue MakeLab changed to Off|2019-05-21 16:16:11|
... and many more ...
sqlite>
```

## Select Temperature Device IDX=120

```
sqlite> select * from Temperature where devicerowid=120;
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:40:00
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:45:00
... and many more ...
```

## Select Temperature,Name Device IDX=120

Show the device Name and limit the number of entries to 5.

```
sqlite> select * from Temperature, (select Name from devicestatus where id=120) as Name where
devicerowid=120 limit 5;
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:45:00|Aussen
120|16.2|0.0|91|0|14.73|0.0|2019-05-21 09:50:00|Aussen
120|16.2|0.0|91|0|14.73|0.0|2019-05-21 09:55:00|Aussen
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 10:00:00|Aussen
120|16.4|0.0|91|0|14.93|0.0|2019-05-21 10:05:00|Aussen
```

Another way to get the result using mode csv and nested tables.

```
sqlite> .mode csv
sqlite> select d.name, t.* from devicestatus d, temperature t where d.ID=t.devicerowid and
t.devicerowid=120 limit 5;
Name,DeviceRowID,Temperature,Chill,Humidity,Barometer,DewPoint,SetPoint,Date
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:15:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:20:00"
```

```
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:25:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:30:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:35:00"
```

## Select Temperature Device IDX=120 specific day.

If using select with between datetime('2019-05-21') and datetime('2019-05-22');, the first entry of the last date (i.e. 2019-05-22) is also shown.

```
sqlite> select * from Temperature, (select Name from devicestatus where id=120) as Name where devicerowid=120 and date >=datetime('2019-05-21') and date < datetime('2019-05-22');

120|17.5|0.0|84|0|14.77|0.0|2019-05-21 13:35:00|Aussen
120|17.5|0.0|84|0|14.77|0.0|2019-05-21 13:40:01|Aussen
...
```

## Select Plans

The .header on and .mode column mode are set prior select command.

```
sqlite> .header on
sqlite> .mode column
sqlite> select * from plans;

ID      Order     Name          FloorplanID Area
-----  -----  -----
1       1         $Hidden Devices  0
2       9         Information    1
4       12        Termine       0
5       18        Volumio       3           123,118
8       17        Hardware Monito 0
11      7          Esszimmer (EZ) 0
12      13        Wohnzimmer (WZ) 0
13      4          Bad           0
14      5          Dusche         0
15      11        MakeLab       0
16      8          Haus          0
17      10        Keller         0
```

## Select Roomplans with associated Devices

```
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan      PlanID     Device      Idx
-----  -----
Bad       13         Hzg Bad Batterie 209
Bad       13         Hzg Bad Sollwert 207
Bad       13         Hzg Bad Temperat 208
Dusche    14         Hzg Dusche Batte 200
Dusche    14         Hzg Dusche Sollw 198
Dusche    14         Hzg Dusche Tempe 199
... and more ...
```

Another output in CSV format with header

*Note*

If the device name contains blanks, the name is put between "", i.e. "Device Name"

```
cd /home/pi/domoticz
sqlite3

sqlite> .open domoticz.db
sqlite> .header on
sqlite> .mode csv
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan,PlanID,Device,Idx
Bad,13,"Hzg Bad Batterie",209
Bad,13,"Hzg Bad Sollwert",207
Bad,13,"Hzg Bad Temperatur",208
Dusche,14,"Hzg Dusche Batterie",200
Dusche,14,"Hzg Dusche Sollwert",198
```

```
Dusche,14,"Hzg Dusche Temperatur",199
"Esszimmer (EZ)",11,"Hzg EZ Batterie",206
"Esszimmer (EZ)",11,"Hzg EZ Sollwert",204
"Esszimmer (EZ)",11,"Hzg EZ Temperatur",205
"Esszimmer (EZ)",11,"Hue EZ",111
... and more ...
```

## Select Setpoint Timers

Each radiator thermostat has one or more timers to control the temperature.  
The timers are defined in the Domoticz Database table SetpointTimers.

```
sqlite> .header on
sqlite> .mode csv
sqlite> SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID;

Name, ID, Active, DeviceRowID, Date, Time, Type, Temperature, TimerPlan, Days, Month, MDay, Occurrence
"Hzg Bad Sollwert", 1, 1, 207, 2019-12-15, 06:00, 2, 20.0, 0, 128, 0, 0, 0
"Hzg Bad Sollwert", 2, 1, 207, 2019-12-15, 11:00, 2, 19.0, 0, 128, 0, 0, 0
"Hzg Bad Sollwert", 3, 1, 207, 2019-12-14, 23:00, 2, 17.0, 0, 128, 0, 0, 0
"Hzg EZ Sollwert", 4, 1, 204, 2019-12-14, 06:00, 2, 20.0, 0, 128, 0, 0, 0
"Hzg EZ Sollwert", 5, 1, 204, 2019-12-14, 22:00, 2, 17.0, 0, 128, 0, 0, 0
"Hzg WZ Sollwert", 6, 1, 201, 2019-12-15, 06:15, 2, 21.0, 0, 128, 0, 0, 0
"Hzg WZ Sollwert", 7, 1, 201, 2019-12-14, 22:00, 2, 0.0, 0, 128, 0, 0, 0
"Hzg Dusche Sollwert", 8, 1, 198, 2019-12-14, 21:30, 2, 21.0, 0, 128, 0, 0, 0
"Hzg Dusche Sollwert", 9, 1, 198, 2019-12-14, 23:00, 2, 0.0, 0, 128, 0, 0, 0
"Hzg MakeLab Sollwert", 10, 1, 195, 2019-12-14, 19:00, 2, 0.0, 0, 128, 0, 0, 0
```

## Delete LightingLog Rows Device IDX=161

Delete all entries from table LightingLog for the device with IDX=161.

```
Check if there are entries:
sqlite> select * from lightinglog where devicerowid=161;
161|0|Security Main Door changed to Alarm + Tamper|2019-05-21 09:01:07|
161|0|Security Main Door changed to Normal|2019-05-21 09:05:00|
161|0|Hue MakeLab changed to Off|2019-05-21 16:16:11|
... and many more ...
```

```
Delete the rows:
sqlite> delete from lightinglog where devicerowid=161;
```

```
Check again if there are entries = select returns no entries:
sqlite> select * from lightinglog where devicerowid=161;
sqlite>
```

## Delete Temperature Rows Date < 2018-08-28

Delete all entries from table [Temperature] with Date < 2018-08-28.

```
sqlite> DELETE FROM [Temperature] WHERE [DeviceRowID] = 28 AND [Date] < date('2018-08-28');
```

```
Check if select returns no entries:
sqlite> SELECT * FROM [Temperature] WHERE [DeviceRowID] = 28;
sqlite>
```

## Delete all Device Data

Delete all data from various tables (Meter, MultiMeter-Calendar) holding data for the device with IDX=44 (Name “Stromverbrauch”).

Use a “Select” statement to check if all data has been deleted.

```
sqlite> DELETE FROM Meter WHERE DeviceRowID=44
sqlite> SELECT * FROM Meter WHERE DeviceRowID=44
sqlite> DELETE FROM MultiMeter_Calendar WHERE DeviceRowID=44
sqlite> SELECT * FROM MultiMeter_Calendar WHERE DeviceRowID=44
```

*Note*

The SQL statements are DELETE followed by a SELECT Query to check if no result.

## Updating Device Data

Be careful on using – this is just an example – NOT recommended to use.  
If doing so, make a database backup first!

### Change the device type and subtype in table DeviceStatus

```
UPDATE DeviceStatus SET Type="32" WHERE ID=163;
UPDATE DeviceStatus SET SubType="0" WHERE ID=163;
```

### Change the DeviceRowID from one Device to another Device

In this example the device with the DeviceRowID 120 is replaced a new device with DeviceRowID 300. Both are temperature & humidity devices.

The device with the idx 120 will be deleted, but the historical (log) data is to be kept in the new device idx 300.

In table Temperature\_Calendar, the DeviceRowID = 120 is updated by DeviceRowID = 300.

```
sudo service domoticz stop
cd /home/pi/domoticz
sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
Connected to a transient in-memory database.
sqlite> .open domoticz.db
sqlite> UPDATE Temperature_Calendar SET DeviceRowID = 300 WHERE DeviceRowID = 120;
sqlite> SELECT * FROM Temperature_Calendar WHERE DeviceRowID = 120;
sqlite> .exit
sudo service domoticz start
```

## Steps

1. Open CLI
2. Stop Domoticz and set domoticz folder
3. Start sqlite3 and open the domoticz database
4. Run sql statement to update the devicerowid from 120 to 300
5. Run sql query to check if no records for devicerowid 120
6. Start Domoticz
7. Open the GUI and check the log of device with idx 120

## Replace Character in sValue

Example of replacing the comma (“,”) character by a semicolon (“;”) in a table field.

```
UPDATE LightingLog SET sValue = replace( sValue, ',', ';' );
```

For example: 20.0,23.0 gets 20.0;23.0

## SQL Statements from File

Executing SQL statements from a text file, is an option to clean database entries (DELETE) or select multiple entries (SELECT).

```
.read FILE
```

If the path is not specified, the file read is taken from the current folder.

If the file contains multiple SQL statements, ensure to close the command with “;”.

### Example

- .read test.sql  
(read in this case from folder /home/pi/domoticz as set as default using cd domoticz)
- .read /home/pi/test.sql  
(include the path)

### Define

File: /home/pi/domoticz/test.sql

Two SQL statements to select records for IDX=97 and 98.

```
select * from lightinglog where devicerowid=97;
select * from lightinglog where devicerowid=98;
```

### Note

Instead two SQL statements, one can be used. But using two statements was just to show multiple statements in a file.

```
select * from lightinglog where devicerowid in (97,98);
```

## Execute Interactive

```
cd domoticz
sqlite3
sqlite> .open domoticz.db
sqlite> .header on
sqlite> .read test.sql
97|0|0 J 97 T|2019-05-21 01:00:00|
97|0|0 J 98 T|2019-05-22 01:00:00|
98|0|31-05-2019|2019-05-21 00:30:00|
98|0|31-05-2019|2019-05-22 00:30:00|
```

## Execute Command-Line

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/test.sql"
```

### Note

See next for more info on running from the command-line.

## SQL Command Line

Using the sqlite3 tool with arguments from the command line

- Argument1: The full path to the Domoticz database
- Argument 2: The SQL dot command ".read path-to-sql-command-file"

The SQL statements are stored in files, located in folder

```
/home/pi/domoticz/scripts/dzVents/scripts/
```

and read by the SQL dot command

```
.read
```

If running multiple SQL statements, separate the SQL statements by ";".

The command-line can also be used from dzVents Lua scripts  
(see below example dzVents Lua Execute SQLite).

### Example SQLite3 Command-Line with the two arguments

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-all.sql"
```

The SQL file contains two statements (taken from the function [Time Control](#)):

```
-- Domoticz Home Automation Workbook - Function Time Control
-- SQL-Command-File: time-control-all.sql
-- Table [LightingLog], Field [DeviceRowID] = 101;
-- Calculate for all rows the time difference in hours for the subsequent rows where nValue is 1
-- (switch is on)
-- Run from the command line with:
-- sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-all.sql"

.header on
-- List the blocks
WITH rows AS
(
  SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM [lightinglog]
  WHERE [devicerowid] = 101
)
SELECT startdata.date AS Start,
       enddata.date AS End,
       ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
       -- Round(Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 As Real),2) AS Minutes
       -- Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 * 60 As Integer) AS Secs
FROM rows startdata
JOIN rows enddata
ON startdata.rownumber = enddata.rownumber - 1
WHERE startdata.nValue = 1;

-- Show the total
WITH rows AS
(
  SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM [lightinglog]
  WHERE [devicerowid] = 101
)
```

```

SELECT ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),4 ) AS
Total
FROM rows startdata
JOIN rows enddata
ON startdata.rownumber = enddata.rownumber - 1
WHERE startdata.nValue = 1;

```

## Examples Data Selection

The next examples are taken from the function [Time Control](#) and use:

- Domoticz database “domoticz.db”
- Table ”LightingLog”
- Fields ”DeviceRowID”, ”nValue” and ”Date” for queries
- Switch device with IDX=101 - stored in field ”DeviceRowID” and switch state in field ”nValue” with values 0 (=Off) and 1 (=On).

### Select All Entries Switch IDX=101 (interactive)

```

sqlite> select * from [LightingLog] where [DeviceRowID] = 101;

DeviceRowID|nValue|sValue|User|Date
101|1|0|Admin|2020-02-19 19:10:57
101|0|0|Admin|2020-02-19 19:12:05
101|1|0|Admin|2020-02-19 19:14:17
101|0|0|Admin|2020-02-19 19:18:26
101|1|0|Admin|2020-02-20 10:12:52
101|0|0|Admin|2020-02-20 10:16:16

```

### Select Today Entries Switch IDX=101 (interactive)

The entries are listed with a subsequent row number (window function ROW\_NUMBER()), which is used in next examples to calculate the time difference between switch On & Off state. The state is subsequent, which can be used to determine the time difference between on and off. To get the time difference, the subsequent state must be 1,0,1,0..., therefor ensure the switch is turned off.

```

sqlite> select ROW_NUMBER() OVER () Nr,* from [LightingLog] where [DeviceRowID] = 101 and [Date] >=
date('now');

Nr|DeviceRowID|nValue|sValue|User|Date
1|101|1|0|Admin|2020-02-20 10:12:52
2|101|0|0|Admin|2020-02-20 10:16:16
3|101|1|0|Admin|2020-02-20 10:25:55
4|101|0|0|Admin|2020-02-20 11:14:20

```

## Select All Activity Blocks

This is a more complex SQL statement to get

- the difference in time (hours, minutes, or seconds) between switch state On (nValue=1) and Off (nValue=0). The time between on & off is a block.
- the total time difference.

There are two SQL statements used for a) and b).

The SQL statement are

- separated by ";" to enable executing multiple SQL statements.

- stored in a file called time-control-all.sql  
(located in folder /home/pi/domoticz/scripts/dzVents/scripts/) and read by sql dot command .read.

## Define SQL Statements

```
-- SQL-Command-File: time-control-all.sql
.header on
WITH rows AS
(
  SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM [lightinglog]
  WHERE [devicerowid] = 101
)
SELECT startdata.date AS Start,
       enddata.date AS End,
       ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
       -- Round(Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 As Real),2) AS Minutes
       -- Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 * 60 As Integer) AS Secs
FROM rows startdata
JOIN rows enddata
ON startdata.rownumber = enddata.rownumber - 1
WHERE startdata.nValue = 1;

WITH rows AS
(
  SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM [lightinglog]
  WHERE [devicerowid] = 101
)
SELECT ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),4 ) AS Total
FROM rows startdata
JOIN rows enddata
ON startdata.rownumber = enddata.rownumber - 1
WHERE startdata.nValue = 1;
```

## Execute SQL Statements

Run from the command-line:

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-all.sql"

Start|End|Hours
2020-02-20 10:12:52|2020-02-20 10:16:16|0.057
2020-02-20 10:25:55|2020-02-20 11:14:20|0.807
2020-02-20 15:55:30|2020-02-20 16:22:14|0.446
2020-02-20 16:22:37|2020-02-20 16:45:08|0.375
2020-02-20 17:06:55|2020-02-20 17:06:57|0.001
2020-02-21 09:39:23|2020-02-21 09:50:17|0.182
Total
1.8667
```

## Select Today Activity Blocks

Basically, as previous, but with an additional WHERE clause "[date] >= date('now')".

## Define SQL Statements (truncated)

```
-- SQL-Command-File: time-control-today.sql
.header on
WITH rows AS (SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber   FROM [lightinglog] WHERE [devicerowid] = 101 AND [date] >= date('now'))
SELECT startdata.date AS Start, enddata.date AS End, ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
FROM rows startdata JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1 WHERE startdata.nValue = 1;
```

```

WITH rows AS (SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber FROM [lightinglog] WHERE
[devicerowid] = 101 AND [date] >= date('now'))
SELECT ROUND( SUM( CAST( JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),3 ) AS
Total
FROM rows startdata JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1 WHERE
startdata.nValue = 1;

```

## Execute SQLite Command Line with result

```

sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-
today.sql"

Start|End|Hours
2020-02-20 10:12:52|2020-02-20 10:16:16|0.057
2020-02-20 10:25:55|2020-02-20 11:14:20|0.807
2020-02-20 15:55:30|2020-02-20 16:22:14|0.446
2020-02-20 16:22:37|2020-02-20 16:45:08|0.375
2020-02-20 17:06:55|2020-02-20 17:06:57|0.001
Total
1.685

```

### Notes

To display the time only, use the function strftime('%H:%M',startdata.date) and strftime('%H:%M',enddata.date). See next.

## dzVents Lua Execute SQLite

An example of a dzVents Lua script executing the sql-command-file "time-control-today". The script is triggered by a switch (Switch Type Push On Button, IDX=102) and executes the sql-input-file.

The script creates some additional Domoticz log entries to explore how to use further.

The output of the sql-input-file (as shown previous) is captured and used to update a text device (IDX=106) by switching the push-on-button.

The format of the output can be set by using the sqlite dot command ".mode".

## Text Device

Some examples with various .mode commands - the first without mode. Try out which suits best.

<i>Mode command not used (default)</i>	.mode tabs	.mode columns	.mode line
2020-02-21 Start End Hours 09:39 09:50 0.182 10:44 10:48 0.06 10:56 11:00 0.069 11:00 11:02 0.033 Total 0.344	2020-02-21 Start End Hours 09:39 09:50 0.182 10:44 10:48 0.06 10:56 11:00 0.069 11:00 11:02 0.033 Total 0.344	2020-02-21 Start End Hours ----- 09:39 09:50 0.182 10:44 10:48 0.06 10:56 11:00 0.069 11:00 11:02 0.033 Total ----- 0.344	2020-02-21 Start = 09:39 End = 09:50 Hours = 0.182  Start = 10:44 End = 10:48 Hours = 0.06  Start = 10:56 End = 11:00 Hours = 0.069  Start = 11:00 End = 11:02 Hours = 0.033 Total = 0.344
.mode list .separator " - "			
2020-02-21			

Start - End - Hours 09:39 - 09:50 - 0.182 10:44 - 10:48 - 0.06 10:56 - 11:00 - 0.069 11:00 - 11:02 - 0.033 Total 0.344		
--	--	--

## dzVents Lua Script

Source: time\_control\_today.lua

```
-- Script triggered by switch (Switch Type Push On Button, idx=102) and executes sql-command-file.
-- Output of sql-command-file is captured and used to update a text device (idx=106).
-- Devices
-- Light/Switch,Switch,On/Off,Time Control Today,Idx: 102
-- General,Text,Time Control Status,Idx: 106.

IDXSWITCH = 102
IDXSTATUS = 106
SQLCMD = "sqlite3 /home/pi/domoticz/domoticz.db '.read /home/pi/domoticz/scripts/dzVents/scripts/time-control-today.sql'

-- split a string by delimiter
-- return array, i.e. array[1], array[2]
function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

local function osExecute(cmd)
    local fileHandle      = assert(io.popen(cmd, 'r'))
    local commandOutput   = assert(fileHandle:read('*a'))
    local returnTable     = {fileHandle:close()}
    -- rc[3] contains resultCode
    return commandOutput,returnTable[3]
end

return {
    on = {
        devices = { IDXSWITCH },
    },
    execute = function(domoticz, item)
        if (item.isDevice) then
            if (domoticz.devices(IDXSWITCH).state == "On") then
                output,rt = osExecute(SQLCMD)
                -- local handle = os.execute(sqlcmd)
                rawdate = domoticz.time.rawDate .. " "
                -- add the date to the output
                output = string.format("%s\n%s",rawdate,output)
                domoticz.log(output, domoticz.LOG_INFO)
                -- update the status device
                domoticz.devices(IDXSTATUS).updateText(output)
                -- Returncode: 0=OK
                domoticz.log(tostring(rt), domoticz.LOG_INFO)
                -- convert the output to a table. Use #outputtable to get the number of entries.
                outputtable = splitstring(output, "\n")
                domoticz.log(outputtable, domoticz.LOG_INFO)
            end
        end
    end
}
```

## Domoticz Log

2020-02-21 11:43:40.343 (VirtualDevices) Light/Switch (Time Control Today) 2020-02-21 11:43:40.327 Status: User: Admin initiated a switch command (102/Time Control Today/On) 2020-02-21 11:43:40.481 Status: dzVents: Info: Handling events for: "Time Control Today", value: "On" 2020-02-21 11:43:40.481 Status: dzVents: Info: ----- Start internal script: time_control_today: Device: "Time Control Today (VirtualDevices)", Index: 102
---

```
2020-02-21 11:43:40.507 Status: dzVents: Info: 2020-02-21
2020-02-21 11:43:40.507 Start|End|Hours
2020-02-21 11:43:40.507 09:39|09:50|0.182
2020-02-21 11:43:40.507 10:44|10:48|0.06
2020-02-21 11:43:40.507 10:56|11:00|0.069
2020-02-21 11:43:40.507 11:00|11:02|0.033
2020-02-21 11:43:40.507 Total
2020-02-21 11:43:40.507 0.344
2020-02-21 11:43:40.507
2020-02-21 11:43:40.508 Status: dzVents: Info: 0
2020-02-21 11:43:40.508 Status: dzVents: Info: {"2020-02-21 ", "Start|End|Hours", "09:39|09:50|0.182",
"10:44|10:48|0.06", "10:56|11:00|0.069", "11:00|11:02|0.033", "Total", "0.344", ""}
2020-02-21 11:43:40.509 Status: dzVents: Info: ----- Finished time_control_today
```

## Python Scripts

Various examples of Python scripts to perform SQL operations.

The lastest version of the scripts can be found in the scripts folder (file names sql-\* .py), plus some additional scripts not documented here.

### Select Device Status

Select the device status for all devices from the Domoticz database table [DeviceStatus].

Scrip name: sql-select-devicestatus.py

```
...
#!/usr/bin/env python
## File: sql-select-devicestatus.py
## Project: Domoticz-Homeautomation-Workbook
## Purpose: Test select device status for all devices; select temperatures.
## Author: Robert W.B. Linn
## Version: 20191218

## Table(s) used
## CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT NULL, [DeviceID]
VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(100) DEFAULT Unknown, [Used] INTEGER
DEFAULT 0, [Type] INTEGER NOT NULL, [SubType] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0,
[Favorite] INTEGER DEFAULT 0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0,
[nValue] INTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFAULT
(datetime('now','localtime')), [Order] INTEGER BIGINT(10) default 0, [AddjValue] FLOAT DEFAULT 0,
[AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0, [AddjMulti2] FLOAT DEFAULT 1, [StrParam1]
VARCHAR(200) DEFAULT '', [StrParam2] VARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0,
[Protected] INTEGER DEFAULT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '',
[Options] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL);

import sqlite3
from sqlite3 import Error
import time
import os

# Define globals
version="SQL Test v20180828"
# Domoticz database path
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

#####
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_device_status(conn):
    """
    Query all rows in the tasks DeviceStatus
    :param conn: the Connection object
    :return:
    """
    print("Select Device Status")
```

```
cur = conn.cursor()
cur.execute("SELECT * FROM DeviceStatus")

rows = cur.fetchall()

for row in rows:
    print(row)

def main():
    print(version)

    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection
        conn = create_connection(database)
        # select data
        with conn:
            select_all_device_status(conn)

        conn.close()

if __name__ == '__main__':
    main()
```

```
python3 sql-select-devicestatus.py

SQL Test v20180828
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Device Status
(1, 1, '0000044C', 1, 'RPi RAM Usage', 1, 243, 6, 0, 1, 12, 255, 0, '32.35', '2019-12-18 09:21:47',
104, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, 'RPi Memory Usage increases over time and needs to be
watched.', None, None)
(2, 1, '0000044E', 1, 'RPi HDD /boot', 0, 243, 6, 0, 0, 12, 255, 0, '1.50', '2019-12-18 09:19:16', 1,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(3, 1, '0000044F', 1, 'RPi Disc Usage', 1, 243, 6, 0, 1, 12, 255, 0, '32.86', '2019-12-18 09:19:16',
105, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(4, 1, '1', 1, 'RPi CPU Temp', 1, 80, 5, 0, 0, 12, 255, 0, '41.9', '2019-12-18 09:21:47', 117, 0.0,
1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(5, 1, '0000044D', 1, 'RPi CPU Usage', 1, 243, 6, 0, 0, 12, 255, 0, '3.55', '2019-12-18 09:21:36', 11,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
...
and many more
```

## Select Device Status IDX=201

Script name: sql-select-idx201.py

```

#!/usr/bin/env python
## File: sql-select-idx201.py
## Project: Domoticz-Homeautomation-Workbook
## Purpose: Test selecting device status for device with idx=201
## Author: Robert W.B. Linn
## Version: 20191218
import sqlite3
from sqlite3 import Error
import time
import os

version="SQL Test v20191218"
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: Database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_device_status(conn,idx):
    """
    Query rows table DeviceStatus for a specific device ID (=Idx)
    :param conn: connection object
    :param idx: idx of the device
    :return:
    """
    print("Select Device Status for Idx=%s " % (idx))
    cur = conn.cursor()
    table = "DeviceStatus"

    sql = "SELECT group_concat(name, ' , ') FROM pragma_table_info(?)"
    args = (table,)
    cur.execute(sql, args)
    colnames = cur.fetchone()
    print(colnames[0])

    sql = "SELECT * FROM {tn} WHERE ID={idx}"
    cur.execute(sql.format(tn=table, idx=str(idx)))
    devicedata = cur.fetchall()
    print(devicedata)
    print("****")
    for row in devicedata:
        print(row)                  #all fields as tupels
        print("----")
        print("ID (=Idx)=%s" % row[0]) #id with value 201
        print("----")
        print(row[0],row[1],row[2])    #more fields
        print("****")

def main():
    print(version)
    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection

```

```
conn = create_connection(database)
with conn:
    select_device_status(conn, 201)

conn.close()

if __name__ == '__main__':
    main()
```

```
python3 sql-select-idx201.py

SQL Test v20191217
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Device Status for Idx=201
ID, HardwareID, DeviceID, Unit, Name, Used, Type, SubType, SwitchType, Favorite, SignalLevel,
BatteryLevel, nValue, sValue, LastUpdate, Order, AdjValue, AdjMulti, AdjValue2, AdjMulti2,
StrParam1, StrParam2, LastLevel, Protected, CustomImage, Description, Options, Color
[(201, 20, '00140001', 1, 'Hzg WZ Sollwert', 1, 242, 1, 0, 0, 12, 255, 1, '21.0', '2019-12-17
06:15:02', 181, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, '')]
 ***
201
201 20 00140001
***
```

## Delete Temperature Rows

<TODO> Add function, i.e. delete\_temperature\_by\_date(conn, date).

Script name: sql-delete-temperature-threshold.py

```
#!/usr/bin/env python
## File: sql-delete-temperature-threshold.py
## Project: Domoticz-Homeautomation-Workbook
## Purpose: Test deleting records from the SQLite database domoticz.db table Temperature
## Author: Robert W.B. Linn
## Version: 20180828

## Table(s) used
## CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill] FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT 0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));

import sqlite3
from sqlite3 import Error
import time
import os

# Define globals
version="SQL Test v20180828"
# Domoticz database path
# Tests
database="/home/pi/python/sql/domoticz.db"
# Production
#database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

####
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_temperatures(conn):
    """
    Query all rows in the tasks Temperature
    :param conn: the Connection object
    :return:
    """
    print("Select all Temperatures")
    cur = conn.cursor()
    cur.execute("SELECT * FROM Temperature")

    rows = cur.fetchall()

    for row in rows:
        print(row)

def select_temperature_by_threshold(conn, threshold):
    """
    Query temperatures by threshold
    :param conn: the Connection object
    :param threshold:
```

```
:return:  
"""  
print("Select Temperature by Threshold %s" % (threshold))  
cur = conn.cursor()  
cur.execute("SELECT * FROM Temperature WHERE Temperature>?", (threshold,))  
  
rows = cur.fetchall()  
  
for row in rows:  
    print(row)  
  
def delete_temperature_by_threshold(conn, threshold):  
    """  
    Delete all temperatures by threshold  
    :param conn: the Connection object  
    :param threshold:  
    :return:  
    """  
    print("Delete Temperature by Threshold %s" % (threshold))  
    cur = conn.cursor()  
    cur.execute("DELETE FROM Temperature WHERE Temperature>?", (threshold,))  
    conn.commit()  
  
def main():  
    print(version)  
  
    if DatabaseExists(database):  
        print("Connecting to the database...",database)  
        # create a database connection  
        conn = create_connection(database)  
        with conn:  
            threshold = 40.0  
  
            select_temperature_by_threshold(conn, threshold)  
  
            delete_temperature_by_threshold(conn, threshold)  
  
            # Check if zero  
            select_temperature_by_threshold(conn, threshold)  
  
        conn.close()  
  
if __name__ == '__main__':  
    main()
```

```
python3 sql-delete-test.py  
  
SQL Test v20180828  
Database /home/pi/domoticz/domoticz.db found.  
Connecting to the database... /home/pi/domoticz/domoticz.db  
Select Temperature by Threshold 40.0  
(2, 40.8, 0.0, 0, 0, 0.0, 0.0, '2018-08-28 19:20:00')  
Delete Temperature by Threshold 40.0  
Select Temperature by Threshold 40.0
```

# Tinkerforge

## Purpose

- To explore how to interact between Domoticz and the [Tinkerforge Building Blocks](#).
- To explore how to write Python scripts with the Tinkerforge [Python API bindings](#) triggered by dzVents.
- To learn how to write Domoticz Hardware Plugins with the Domoticz [Python Framework](#) using the Tinkerforge Python API bindings.

**TinkerForge** is an open source hardware platform of stackable microcontroller building blocks (Bricks) that can control different modules (Bricklets).

There is wide range of bricklets enabling to build all kind of solutions.

The building blocks can be controlled using a variety of high-level programming languages through API bindings. The focus is on using the Python API bindings - unfortunate Lua is not supported (author's favorite Domoticz scripting language through dzVents).

Recommend visiting the Tinkerforge website to explore more indepth whats possible with the Building Blocks.

The setup to explore Tinkerforge uses following hard- and software enabling to test the various concepts.

## Hardware

- Tinkerforge Master Brick 2.1 connected, via USB, to the Raspberry Pi 3B+ running the Domoticz Development System v4.x
- Ambient Light Bricklet 2.0
- IO4 Bricklet 2.0
- LCD 20x4 Bricklet 1.2
- RGB LED Bricklet 2.0

## Software

- Raspberry Pi Raspian Debian Linux Buster 4.19.93-v7+ #1290
- Domoticz Home Automation System 4.11696 (BETA)
- Tinkerforge Python API-Binding 2.1.24
- Python 3.7.3, GCC 8.2.0
- *Note:* The versions are subject to change.

Installed the Tinkerforge Brick Daemon (brickd) and the Brick Viewer (brickv) on the Domoticz Development System.

The Brick Daemon acts as a bridge between the Bricks/Bricklets and the application.

The Brick Viewer provides a graphical interface for testing Bricks and Bricklets.

The Python API binding is used to control (via TCP/IP connection) the Bricks and Bricklets from the application.

# Concepts

Several concepts explored to interact between Domoticz and Tinkerforge.

Interaction means:

- **Get** data from a Tinkerforge Bricklet which is taken by Domoticz to action accordingly
- **Set** data from Domoticz to the Tinkerforge Bricklet
- **Listen via Callbacks** to Tinkerforge Bricklet changes which is taken by Domoticz to action accordingly

## Tinkerforge Script

To interact via dzVents by executing Python scripts using the Tinkerforge Python API binding for a bricklet (getter, setter).

## Domoticz Plugin

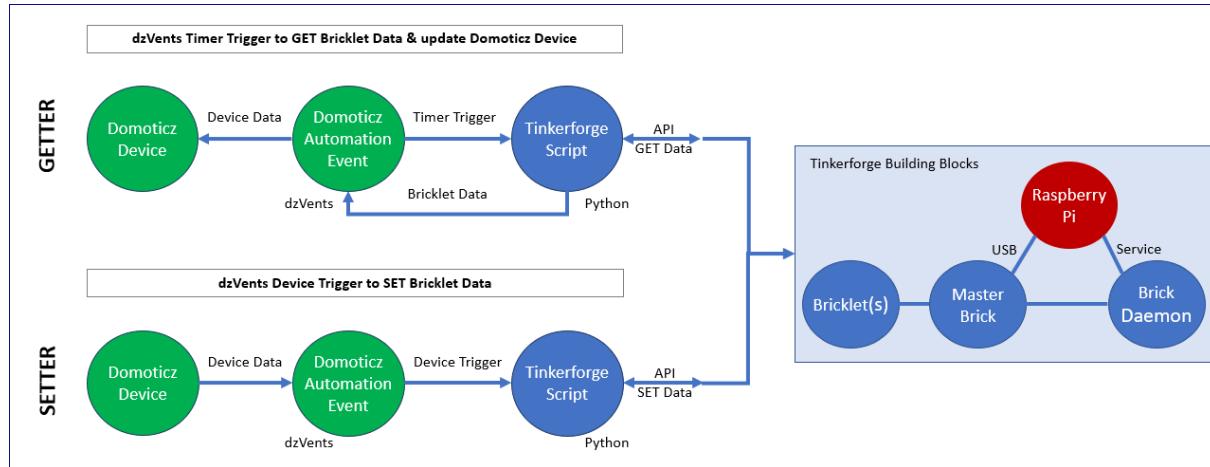
To interact using the Domoticz Python Framework with the Tinkerforge Python API Binding (getter, setter, callbacks).

## Tinkerforge Service

To interact between Domoticz and Tinkerforge v.v. using a service running a Tinkerforge Python Script for one or more bricklets depending functionality (getter, setter, callbacks).

## Tinkerforge Script

To interact between Domoticz & Tinkerforge using dzVents by executing Python scripts with the Tinkerforge Python API binding for the selected bricklet.



This concept is basically a setter or getter with following sequence:

1. dzVents calls the Tinkerforge Python script with parameter JSON format
2. Tinkerforge Python script
  - a. makes a Tinkerforge IP connection
  - b. sets or gets data, i.e. bricklet configuration or value
  - c. disconnects the Tinkerforge IP connection
  - d. returns data and status in JSON format
3. dzVents handles returned data & status, i.e. update Domoticz devices or control action

There is no ongoing connection between Domoticz and Tinkerforge nor is callback used. It is not required to create or use dedicated Domoticz device(s) – depends on the actions. To use callback, it is recommended to create a Domoticz Python Plugin or a Service. As an extension it is also possible to create a combined Tinkerforge Python script for multiple bricklets, with logic integrated or controlled via arguments.

Decided this concept, i.e. running external Python scripts doing the Tinkerforge work, as it is easier to test & maintain (API changes, updates).

### Example GET Illuminance (Lux) from Ambient Light Bricklet 2.0 (snippet)

dzVents script “tfalv2\_get\_lux.lua” executes, triggered by a Timer (every minute), the Tinkerforge Python script “tfalv2\_get\_lux.py” with the Ambient Light Bricklet 2.0 UID (obtained using the Tinkerforge Brick Viewer):

```
python3 tfalv2_get_lux.py '{"UID": "yyc"}'
```

Tinkerforge Python Script returns if ok, the Lux value:

```
{"status": "OK", "title": "", "lux": "146.21"}
```

OR in case of an error

```
{"status": "ERROR", "title": "Error message"}
```

dzVents to decode returned JSON text and update the Domoticz Device with the Lux value:

```
local IDXLUX = 107
...
local result = JSON:decode(output)
domoticz.devices(IDXLUX).updateLux(result.lux)
```

### Example SET Color of RGB LED Bricklet 2.0 (snippet)

dzVents script “tfrgbledv2\_set\_color.lua” executes, triggered by a change of the Domoticz Device Selector Switch value (0=OFF,10=RED,20=YELLOW,30=GREEN), the Tinkerforge Python script “tfrgbledv2\_set\_color.py” with the RGB LED bricklet 2.0 UID (obtained using the Tinkerforge Brick Viewer) and the R,G,B color values – example color yellow 255-255-0.

```
python3 tf_rgbledv2_set_color.py '{"UID": "Jng", "R": 255, "G": 255, "B": 0}'
```

Tinkerforge Python Script returns if ok:

```
{"status": "OK", "title": "255,255,0"}
```

OR in case of an error

```
{"status": "ERROR", "title": "Error message"}
```

## Examples (with reference to the dzVents & Python scripts)

Tinkerforge Bricklet	Example	dzVents Script (.lua)	Trigger	Tinkerforge Script (.py)	Domoticz Device (idx, hardware, name, type, subtype, switchtype, data)	
Ambient Light 2.0 [yyc]	Get illuminance and update Domoticz device	tfalv2_get_lux	Timer Every Minute	tfalv2_get_lux	107, VirtualDevices, MakeLab Lux, Lux, Lux, n/a 2974 Lux	
RGB LED 2.0 [Jng]	Set RGB color triggered by Domoticz selector switch	tfrgbledv2_set_color	Device	tfrgbledv2_set_color	12, VirtualDevices, RGBLED - Traffic Light Switch, Light/Switch, Switch Selector Set Level: 10 %	
LCD 20x4 2.0 [BHN]	Set LCD text HH:mm triggered by dzVents every minute	tflcd20x4_set_clock	Timer Every Minute	tflcd20x4_set_text	Not used. Tinkerforge script writes direct to the LCD Bricklet.	
LCD 20x4 2.0 [BHN]	Set LCD text Raspberry Pi info triggered by dzVents every minute	tflcd20x4_set_rpi_monitor	Timer Every Minute	tflcd20x4_set_text	112, VirtualDevices, LCD Text General, Text Note: Text and display taken different times.	

The latest version of the scripts can be found in the file tf<bricklet>\_set|get\_action.py.

To get started, two templates are available for a dzVents & Python script.

Source: tfbricklet\_set\_get.lua

```
-- Domoticz Home Automation Workbook - Function Tinkerforge
-- dzVents Automation Script: tfbricklet_set_get
-- Template for setting or getting bricklet values.
-- The dzVents Lua script controls the functionality and run a Python script executing Tinkerforge API functions for the bricklet.
-- The logic is handled by the Domoticz script whereas the Python script sets or gets bricklet value(s) only.
-- Python Script
-- The script is triggered every minute or by a device (i.e. switch) and executes python script.
-- The python script expects a string argument (enclosed in '') with key:value pairs:
-- '{"HOST":"IP-Address","PORT":4223,"UID":"yyc"}'
-- The UID argument is mandatory, HOST and PORT are optional. The keys must in UPPERCASE!
-- In addition, key:value pairs can be defined, depending on the functionality and if getter or setter.
-- Examples Bricklets:
-- RGB LED 2.0: set color = '{"HOST":"IP-Address","PORT":4223,"UID":"Jng","R":255,"G":255,"B":0}'
-- LCD 20x4: set text 4 lines =
'{"UID":"BHN","LCDLINES":[{"line":0,"position":0,"clear":1,"text":"R=255"}, {"line":1,"position":5,"clear":1,"text":"CPU 1%"}, {"line":2,"position":0,"clear":1,"text":"RAM 45%"}, {"line":3,"position":15,"clear":1,"text":"DISC 69%"}]'
```

-- Result

-- The result of the command is a JSON formatted string containing the key:value pairs:

```
-- {"status": "OK", "title": ""} or {"status": "ERROR", "title": "Error message"}
```

-- For a getter, the requested values are added to the result.

-- Examples Bricklets:

```
-- Ambient Light 2.0: get lux = {"status": "OK", "title": "", "lux": "146.21"}
```

-- 20200226 by rwbl

-- json library

```
local JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")() -- For Linux
-- define the path to the python script
local CMD = "python3 /home/pi/domoticz/scripts/python/tfbricklet_set_get.py"
```

-- Idx of the devices used, i.e. a switch (set) or a device (get) to be updated

```
local IDXDEVICE = NNN
-- TF Bricklet parameter
local TFUID = "UID"
```

-- run external command

```
local function osExecute(cmd)
```

```

local fileHandle      = assert(io.popen(cmd, 'r'))
local commandOutput  = assert(fileHandle:read('*a'))
local returnTable    = {fileHandle:close()}
-- rc[3] contains returnCode
return commandOutput,returnTable[3]
end

return {
on = {
-- select the function
timer = { 'every minute' },
devices = { IDXDEVICE },
},
-- select the function
execute = function(domoticz, timer)
domoticz.log('Timer event was triggered by ' .. timer.trigger, domoticz.LOG_INFO)

execute = function(domoticz, device)
domoticz.log(string.format('Device changed:%s to %s.',device.name, device.state),
domoticz.LOG_INFO)

-- define the python command with json argument
-- example: python3 tfbricklet_set_get.py {"UID":"yyc"}
local pycmd = string.format('%s \'{"UID":"%s"}\'',CMD,TFUID)
domoticz.log(pycmd, domoticz.LOG_INFO)

-- run the external python scripts ad handle return code
local output,rt = osExecute(pycmd)
-- domoticz.log(string.format('%d:%s',rt,output), domoticz.LOG_INFO)
-- decode the result string, which is json format, i.e.
-- {"status": "OK", "title": "", "lux": "146.21"}
-- {"status": "ERROR", "title": "Error message"}
local result = JSON:decode(output);
-- domoticz.log(string.format('TEST Status:%s, Title:%s,
Lux:%s',result.status,result.title,result.lux), domoticz.LOG_INFO)
-- handle the result of the python command: 0=OK,>0=ERROR
if rt == 0 then
-- handle the result of the tinkerforge scripts
if result.status == "OK" then
-- setter uses status and result only
domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_INFO)

-- getter to get the value and update domoticz deice
domoticz.log(string.format('Status:%s, Title:%s,
Value:%s',result.status,result.title,result.value), domoticz.LOG_INFO)
domoticz.devices(IDXDEVICE).updateLux(result.value)
-- tinkerforge script returns an error
else
domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_ERROR)
end
-- handle the result of the python command
else
domoticz.log(string.format('Python command: RT:%d, Output:%s',rt,output),
domoticz.LOG_ERROR)
end
end
}
}

```

### Source: tfbricklet\_set\_get.py

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
File:      tfbricklet_get_set.py
Project:   Domoticz-Home-Automation-Workbook
Purpose:   Get or set the value of a bricklet.
Depends on: Python API Binding
Description:JSON formatted string as argument with key:value pairs enclosed in ''.
IMPORTANT: all keys must be in uppercase.
Key:Value
"HOST":String - IP address of the host running the brick deamon. Default: "localhost" (optional)
"PORT":integer - Port number. Default: 4223 (optional)

```

```

"UID": "String" - UID of the bricklet. Get from the Brick Viewer, i.e. "yyc" (mandatory)
The UID argument is mandatory, HOST and PORT are optional. The keys must be in UPPERCASE!

Define additional key:value pairs depending bricklet:
Examples Bricklets:
RGB LED 2.0: set color = '{"HOST": "IP-Address", "PORT": 4223, "UID": "Jng", "R": 255, "G": 255, "B": 0}'
LCD 20x4: set text 4 lines =
'{"UID": "BHN", "LCDLINES": [{"line": 0, "position": 0, "clear": 1, "text": "R=255"}, {"line": 1, "position": 5, "clear": 1, "text": "G=0"}, {"line": 2, "position": 10, "clear": 1, "text": "B=0"}, {"line": 3, "position": 15, "clear": 1, "text": "Color"}]}'
Result
The result of the command is a JSON formatted string containing the key:value pairs:
{"status": "OK", "title": ""} or {"status": "ERROR", "title": "Error message"}
For a getter, the requested values are added to the result.
Examples Bricklets:
Ambient Light 2.0: get lux = {"status": "OK", "title": "", "lux": "146.21"}

Tests:
Recommend during script development, to run (iterate) the script for a terminal command line without
using Domoticz (dzVents, Lua).
Example: Run to get the lux value of the bricklet connected to the master brick which is connected via
USB with Raspberry Pi:
cd domoticz/scripts/python
python3 /home/pi/domoticz/scripts/python/tfalv2_get_lux.py '{"UID": "yyc"}'
Output:
{"status": "OK", "title": "", "lux": "177.92"}
```

Version: 20200226 by rwbl

```

"""
import sys
import json

# Tinkerforge
from tinkerforge.ip_connection import IPConnection
# Bricklet api - see Tinkerforge documentation
from tinkerforge.bricklet_<BRICKLETAPINAME> import Bricklet<BRICKLETAPINAME>
## Set defaults for connection
HOST = "localhost"
PORT = 4223
UID = "<UID>"

# Arguments Length
ARGVLEN = 2

if __name__ == "__main__":
    status = 1
    # Get the command line arguments
    # print("Arguments: {}".format(str(sys.argv)))
    if len(sys.argv) == ARGVLEN:
        # Parse the command line argument 1 from string to json
        args = json.loads(sys.argv[1])
        if "HOST" in args:
            HOST = args["HOST"]
        if "PORT" in args:
            PORT = int(args["PORT"])
        if "UID" in args:
            UID = args["UID"]
        # Add additional arguments depending bricklet
        if "VALUE" in args:
            VALUE = args["VALUE"]
        # Examples:
        # RGB LED 2.0 set RGB color: '{"UID": "Jng", "R": 255, "G": 255, "B": 0}'
        # LCD 20x4 set text lines: '{"UID": "BHN",
        "LCDLINES": [{"line": 1, "position": 1, "clear": 1, "text": "Text"}, {"line": 2, "position": 1, "clear": 1, "text": "Text2"}]}'
        #
        # Check arguments
        # print("JSON:H={},P={},U={}".format(HOST,PORT,UID))
    else:
        status = 0
        result = { "status" : "ERROR", "title" : "Wrong number of arguments." }

    if status == 1:
        try:
            # Create IP connection

```

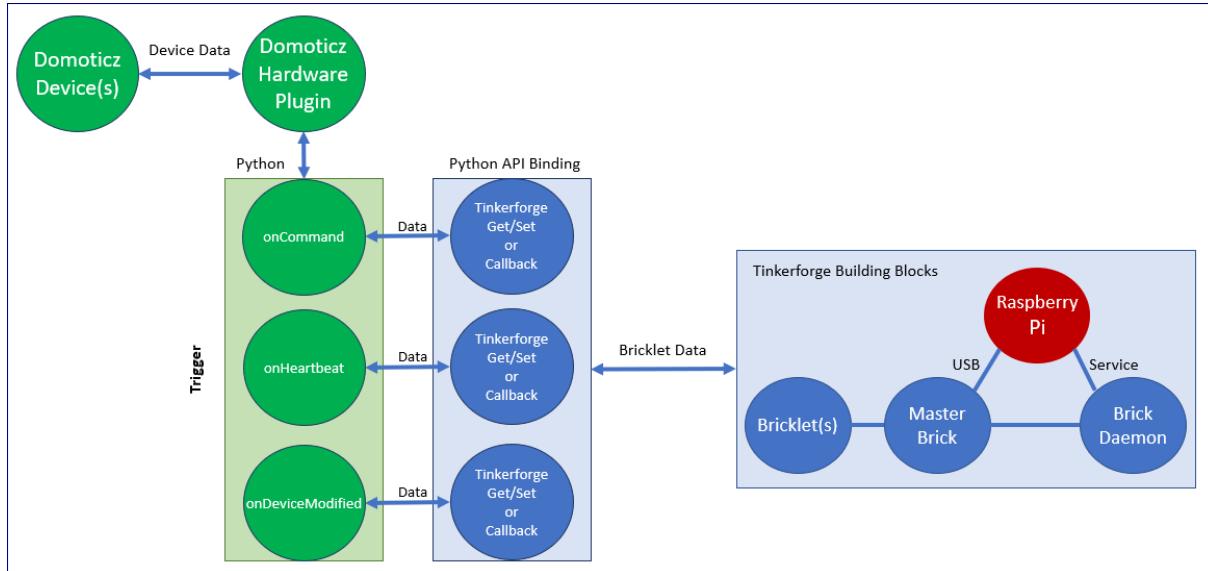
```
ipcon = IPConnection()
# Create device object
devobj = BrickletBRICKLETAPINAME(UID, ipcon)
# Connect to brickd
ipcon.connect(HOST, PORT)
# Check if the bricklet is reachable - else error is triggered
# Depending the version of the bricklet the UID uses different function:
# Older bricklets .get_identity() and newer .read_uid()
# In case the bricklet is not reachable, it takes 1-2 seconds before a response is given
devobj.get_identity()
devobj.read_uid()
# GET
value = devobj.get_<value>()
# define the result containing the mandatory status, title, and the value
result = { "status" : "OK", "title" : "", "value" : "{}".format(value) }

# SET
devobj.set_value(<VALUE>)
result = { "status" : "OK", "title" : "{}".format(<VALUE>) }

# Disconnect
ipcon.disconnect()
except:
    result = { "status" : "ERROR", "title" : "get_value failed. Check master & bricklet." }
# Print the result which is used by the dzVents Lua script
print(json.dumps(result))
```

## Domoticz Plugin

To interact using the Domoticz Python Framework with the Tinkerforge Python API Binding. This solution can be applied to dedicated bricklets or solutions containing multiple bricklets with logic control.



More to be found in section [Plugin Development](#).

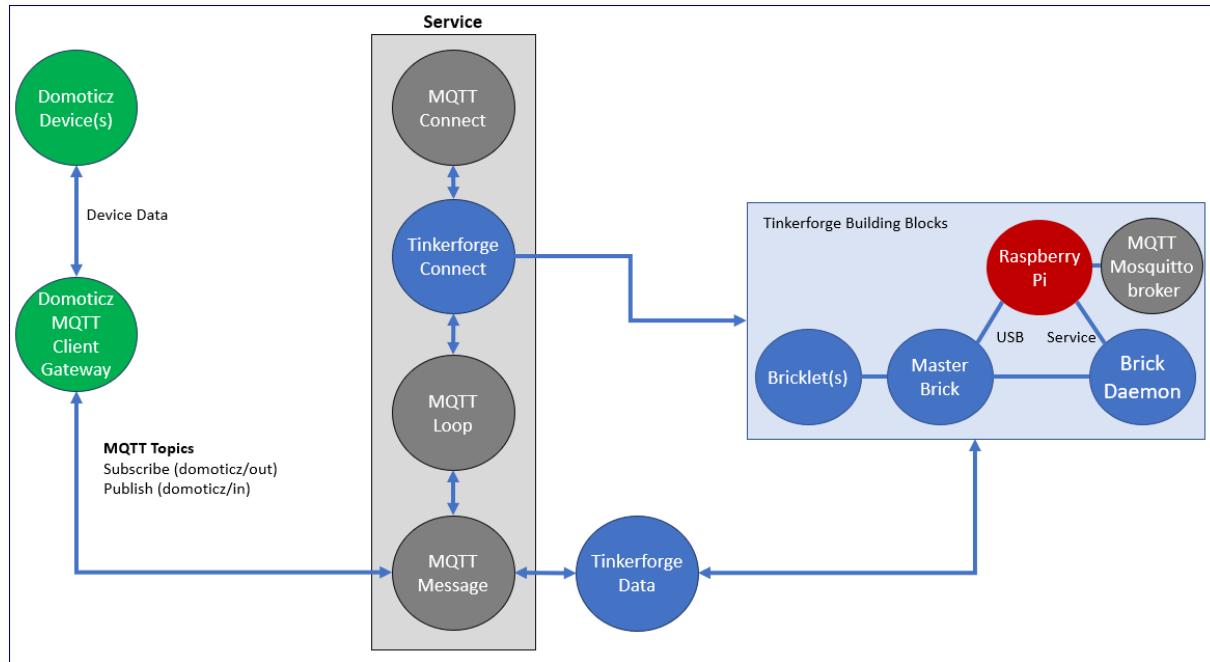
A selection of Domoticz Tinkerforge Plugins can be found in the repository folder `plugins` or on [GitHub](#):

- Air Quality Solution  
(`domoticz-plugin-indoor-air-quality-monitor`)
- Ambient Light Bricklet 2.0  
(`domoticz-plugin-tinkerforge-ambientlightv2`)
- IO4 Bricklet 2.0 with input callback  
(`domoticz-plugin-tinkerforge-io4v2`)
- LCD 20x4 Bricklet Full of buttons callback & configuration options  
(`domoticz-plugin-tinkerforge-lcd20x4`)
- LCD 20x4 Bricklet Lite display text only  
(`domoticz-plugin-tinkerforge-lcd20x4-lite`)
- RGB LED Bricklet 2.0  
(`domoticz-plugin-tinkerforge-rgbledv2`)
- Soil Moisture Solution  
(`domoticz-plugin-soil-moisture-monitor`)
- Tinkerforge Traffic Light with RGB LED Bricklet 2.0  
(`domoticz-plugin-tinkerforge-trafficlight`)

## Tinkerforge Service

To interact between Domoticz and Tinkerforge v.v. using a service running a Tinkerforge Python Script for one or more bricklets depending functionality.

The communication between Domoticz and the Service is via MQTT (using the mosquitto broker and the Domoticz MQTT Client Gateway Hardware).



### Example IO4 Bricklet 2.0

To interact between Domoticz and Tinkerforge v.v. using a service "tfio4v2-mqtt.service" running the Python script "tfio4v2-mqtt.py".

The example shows how to use a Tinkerforge callback by subscribing to a MQTT topic and how to publish MQTT messages to a MQTT topic.

The Python script uses the paho mqtt client library and the Tinkerforge Python API bindings.

The IO4 Bricklet 2.0 configuration (channel, sensor/actor connected, direction, initial value):

- IO4 Channel 0: LED connected. Output, Low.
- IO4 Channel 1: Push-button connected. Input, Low.

There are 2 Domoticz devices used (Idx,Name,Type,SubType,SwitchType):

- 108,TFIO4 Status,General,Text
- 109,TFIO4 LED Switch,Light/Switch,Switch,On/Off

### Outline

By using dzVents, Domoticz can control & take actions. Just some simple examples:

- if the push-button gets pushed, a light can be switched on by dzVents

- if the Raspberry Pi Domoticz system CPU temperature gets too high, switch the RED LED connected to IO4 channel 0 to On

## MQTT Communication

The Python script:

**Subscribes** to topic "domoticz/out" and listens to messages published by Domoticz having IDX=109 (switch).

If the Domoticz switch is set to On/Off, the LED connected to IO4 channel 0 is turned On/Off.

### Domoticz Log

Sample switching On/Off.

```
2020-03-01 11:37:51.740 (VirtualDevices) Light/Switch (TF IO4 LED Switch)
2020-03-01 11:37:51.732 Status: User: Admin initiated a switch command (109/TF IO4 LED Switch/On)
2020-03-01 11:38:03.716 (VirtualDevices) Light/Switch (TF IO4 LED Switch)
2020-03-01 11:38:03.708 Status: User: Admin initiated a switch command (109/TF IO4 LED Switch/Off)
```

**Publishes**, using topic "domoticz/in", a text to Domoticz which updates the text device with IDX=108.

If the push-button connected to IO4 channel 0 is pressed, a message is published to Domoticz. Domoticz updates the text of the device.

Sample text displayed on the text device:

```
IO4 changed:True, value:True, time:2020-03-01 11:38:58
IO4 changed:True, value:False, time:2020-03-01 11:39:05
```

### Service File (tfio4v2-mqtt.service)

Login as user Pi and create the service file in folder /home/pi/domoticz/scripts/python:

```
[Unit]
Description=Tinkerforge Domoticz Service IO4V2
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u tfio4v2-mqtt.py
WorkingDirectory=/home/pi/domoticz/scripts/python
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

Copy the service file to the systemd folder

```
sudo cp tfio4v2-mqtt.service /etc/systemd/system/tfio4v2-mqtt.service
```

### Start the service

```
sudo systemctl start tfio4v2-mqtt.service
```

Check if the service is running

```
ps ax | grep tfio4v2-mqtt
4745 ? Ssl 0:00 /usr/bin/python3 -u tfio4v2-mqtt.py
```

### Stop the service

```
sudo systemctl stop tfio4v2-mqtt.service
```

and check again if the service has stopped = nothing to be returned from the command.

### Start Service at Boot

Start automatically on reboot:

```
sudo systemctl enable tfio4v2-mqtt.service
```

The systemctl command can also be used to restart the service or to disable.

Example with output:

```
sudo systemctl enable tfio4v2-mqtt.service
Created symlink /etc/systemd/system/multi-user.target.wants/tfio4v2-mqtt.service →
/etc/systemd/system/tfio4v2-mqtt.service.
```

### Bash: tfio4v2-mqtt-start.sh

```
#!/bin/bash
## File:tfio4v2-mqtt-start.sh
## Attributes: make the script executable: sudo chmod +x tfio4v2-mqtt-start.sh
## 20200301 rwbl

cd /home/pi/domoticz/scripts/python
# start the service
echo Starting...
sudo systemctl start tfio4v2-mqtt.service
# check if running
echo check if running...
echo entry should be listed line: pi 14621 1 0 19:16 ? 00:00:00 /usr/bin/python3 -u tfio4v2-mqtt
ps -ef | grep tfio4v2-mqtt
echo done
```

### Bash: tfio4v2-mqtt-stop.sh

```
#!/bin/bash
## File:tfio4v2-mqtt-stop.sh
## Attributes: make the script executable: sudo chmod +x tfio4v2-mqtt-stop.sh
## 20200301 rwbl

cd /home/pi/domoticz/scripts/python
# stop the service
echo service stopping...
sudo systemctl stop tfio4v2-mqtt.service
# check if stopped
echo check if stopped...
echo no entry should be listed line: pi 14621 1 0 19:16 ? 00:00:00 /usr/bin/python3 -u tfio4v2-mqtt
ps -ef | grep tfio4v2-mqtt
echo done
```

### Service: tfio4v2-mqtt-stop.service

```
[Unit]
Description=Tinkerforge Domoticz Service IO4V2
After=network.target

[Service]
```

```
ExecStart=/usr/bin/python3 -u tfio4v2-mqtt.py
WorkingDirectory=/home/pi/domoticz/scripts/python
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

## Python: tfio4v2-mqtt.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

File:      tfio4v2-mqtt.py
Project:   Domoticz-Home-Automation-Workbook
Purpose:   To test mqtt communication between Tinkerforge IO4 Bricklet 2.0 and Domoticz devices or
scripts.

The IO4 Bricklet 2.0 configuration:
IO4 Channel 0: LED connected. Output, High.
IO4 Channel 1: Push-button connected. Input, Low.
There are 2 Domoticz devices used (Idx,Name,Type,SubType,SwitchType):
108,TFI04 Status,General,Text
109,TFI04 LED Switch,Light/Switch,Switch,On/Off

To configure the IO4 bricklet, a JSON formatted string with key:value pairs enclosed in '' is required.
* IMPORTANT: all keys must be in uppercase except for the CHANNELS which must be in lowercase.

(Key:Value)
"HOST":"String" - IP address of the host running the brick deamon. Default: "localhost" (optional)
"PORT":integer - Port number. Default: 4223 (optional)
"UID":"String" - UID of the bricklet. Get from the Brick Viewer, i.e. "yyc" (mandatory)
STATUSLED:integer - Set the status LED configuration (see Bricklet documentation)
The UID argument is mandatory, HOST and PORT are optional. The keys must in UPPERCASE!
Additional key:value pairs for the io4v2 bricklet:
"CHANNELS":json array - Channels 0 to 3 with value& direction, ie.
[{"channel":0,"value":1,"direction":"o"}, {"channel":1,"value":0,"direction":"i"}]''
"channel":integer - Channel number 0-3 (mandatory)
"value":integer - State of the channel 0=low (off) 1=high (on)
"direction":String - Direction output "o" or input "i"

Tests
1: Switch the IO4 Channel 0 LED on/off triggered by the Domoticz Switch device (idx=109) (MQTT
subscribe)
2: Set the text of the Domoticz Text device (idx=108) when pushing or release the IO4 Channel 1 push-
button (MQTT publish)
Example running python script from folder /home/pi/domoticz/scripts/python
# IO4 Channel 0: LED connected. Output, High.
# IO4 Channel 1: Push-button connected. Input, Low.
python3 tfio4v2-mqtt.py

The config is set below in the global CONFIG:
'{"UID":"G4d","STATUSLED":2,"CHANNELS":[{"channel":0,"value":0,"direction":"o"}, {"channel":1,"value":0,
"direction":"i"}]}''

Recommend during script development, to run (iterate) the script for a terminal command line without
using Domoticz (dzVents, Lua).
Depends on: Tinkerforge Python API Binding, MQTT Python paho client
Version:    20200229 rwbl
"""

# Imports
import sys
import time
from datetime import datetime
## json to handle incoming mqtt messages from domoticz
import json
## mqtt client library for subscribe & publish mqtt messages
import paho.mqtt.client as mqtt
# Tinkerforge
from tinkerforge.ip_connection import IPConnection
## Bricklet api - see Tinkerforge documentation
from tinkerforge.bricklet_io4_v2 import BrickletIO4V2
```

```

# Info
VERSION = "DoTfI04V2"

# Debug
# DEBUG = True
DEBUG = False

# Configuration
CONFIG =
'{"UID":"G4d","STATUSLED":2,"CHANNELS":[{"channel":0,"value":0,"direction":"o"}, {"channel":1,"value":0,"direction":"i"}]}'

## MQTT connection parameter defaults and domoticz topics publish & subscribe
MQTT_BROKER = "localhost"
MQTT_CLIENT = "tfdomoticzio4"
MQTT_TOPIC_SUB = "domoticz/out"
MQTT_TOPIC_PUB = "domoticz/in"

# Tinkerforge
## Connection parameter defaults
TF_HOST = "localhost"
TF_PORT = 4223
## IO4
IO4_UID = "G4d"
## Set the input channel callback period to 0.25s (250ms) or 500 ...
IO4_CALLBACK_PERIOD = 500
## Set the status LED to heartbeat (see bricklet api doc for the values to be set)
IO4_STATUS_LED = 2
## Channels
IO4_CHANNEL_0 = 0
IO4_CHANNEL_1 = 1

# Domoticz
## (mqtt subscribe) Domoticz switch to turn the io4 led on&off (connected to channel 0)
IDX_SWITCH = 109
## (mqtt publish) Domoticz text device gets updated when io4 button (connected to channel 1) is pressed.
IDX_STATUS = 108

##
## INIT with globals having initial values
##

## MQTT functions
## 

# mqtt subscribe to topic(s) after successful connection to the mqtt broker
# for this solution only the domoticz mqtt topic "domoticz/out" is subscribed to
def on_connect(client, userdata, flags, rc):
    mqttConnected = True
    if DEBUG == True:
        print("MQTT on_connect: client={},userdata={},flags={},rc={},subscribe topic={}".format(client, userdata, flags, rc, mqttTopicSub))
        client.subscribe(mqttTopicSub)
        print("MQTT Connected. Waiting for messages...".format())

# mqtt handle incoming messages from domoticz topic "domoticz/in"
# for this topic, all domoticz mqtt messages are received (json formatted string)
# the mqtt message is json parsed to get the device idx and act.
# for this example, the io4 channel 0 is set to high or low, i.e. led on / off
def on_message(client, userdata, message):
    if DEBUG == True:
        print("MQTT on_message: topic={}".format(message.topic))
        print("MQTT on_message: payload={}".format(message.payload.decode("utf-8")))
        print("MQTT on_message: qos={},retain={}".format(message.qos,message.retain))
        # decode the message payload to be able to parse
        json_string = str(message.payload.decode("utf-8"))
        # Handle changes of the domoticz switch to turn io4 led channel 0 on or off
        set_io4_channel_value(json_string, IDX_SWITCH, IO4_CHANNEL_0)

# mqtt log any messages
def on_log(client, userdata, level, buf):
    if DEBUG == True:
        print("MQTT on_log:client={},userdata={},level={},buf={}".format(client, userdata, level, buf))

```

```

# mqtt create a new instance and start listening.
# prior string to listen to mqtt messages, connect to tinkerforge brick deamon with callback on io4
input channels
def create_mqtt_instance():
    global mqttBroker,mqttClient,mqttTopicSub,mqttTopicPub,mqttConnected

    mqttBroker = MQTT_BROKER
    mqttClient = MQTT_CLIENT
    mqttTopicSub = MQTT_TOPIC_SUB
    mqttTopicPub = MQTT_TOPIC_PUB
    mqttConnected = False

    print("Connecting to Tinkerforge & MQTT...")

    if DEBUG == True:
        print("MQTT create new instance:broker={}.format(mqttBroker,mqttClient))")
    mqttClient = mqtt.Client(mqttClient)
    mqttClient.on_connect = on_connect
    mqttClient.on_message = on_message
    mqttClient.on_log = on_log
    if DEBUG == True:
        print("Connecting to the broker", mqttBroker)
    mqttClient.connect(mqttBroker, 1883, 60)
    time.sleep(4)
    # Connect to the tinkerforge brick daemon with json string holding the configuration
    # Instead of constant CONFIG, an argument sys.argv can be used
    if connect_tinkerforge(CONFIG) == True:
        # Loop the matt client
        mqttClient.loop_forever()
    else:
        print("[ERROR] Cannot connect to Tinkerforge. Check configuration.")
    return

# mqtt publish new message to domoticz
## Examples payload:
## Text device: '{"command": "udevice", "idx": "%s", "svalue": "Hello World at %s"}' %
(IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
def publish_mqtt_message(payload):
    if DEBUG == True:
        print("MQTT publish_mqtt_message:topic={},payload={}".format(mqttTopicSub,payload))
    mqttClient.publish(mqttTopicPub,payload)

def disconnect_mqtt_broker():
    mqttClient.loop_stop()
    if DEBUG == True:
        print("MQTT disconnected mqtt broker={}".format(mqttBroker))

##
## JSON Parsing
##
def get_domoticz_device_property(json_string, idx, property):
    result = ""
    if DEBUG == True:
        print("get_domoticz_device_property: json={},idx={},property={}".format(json_string, idx, property))
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        name = parsed_json['name']
        value = parsed_json[property]
        result = value
        if DEBUG == True:
            print("get_domoticz_device_property: idx={},name={},property={},value={}".format(idx, name, property, value))
    return result

## Text device: '{"command": "udevice", "idx": "%s", "svalue": "Hello World at %s"}' %
(IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
def set_domoticz_text_device(idx, svalue):
    if DEBUG == True:
        print("set_domoticz_text_device: idx={},svalue={}".format(idx, svalue))
    payload = '{{"command": "udevice", "idx": {}, "svalue": "{}"}}'.format(idx,svalue)
    # payload = '{{"command": "udevice", "idx": "%d", "svalue": "%s"}' % (idx, svalue)
    print("set_domoticz_text_device: payload={}".format(payload))

```

```

publish_mqtt_message(payload)
if DEBUG == True:
    print("set_domoticz_text_device: idx={},svalue={}".format(idx, svalue))

# set the value of an io4 input channel depending domoticz witch on/off device nvalue
def set_io4_channel_value(json_string, idx, channel):
    # get all idx properties
    # if DEBUG == True:
    #     print("set_io4_channel_value: json={},idx={}".format(json_string, idx))
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        # get the properties of the selected idx
        if DEBUG == True:
            print("set_io4_channel_value: json={},idx={}".format(json_string, idx))
        name = parsed_json["name"]
        value = int(parsed_json["nvalue"])
        io4Dev.set_selected_value(channel,value)
        print("set_io4_channel_value: channel={},idx={},name={},nvalue={}".format(channel,idx, name, value))

##
## Tinkerforge functions
##

# Connect to the tinkerforge brick deamon, create io4 bricklet device object and set bricklet parameter
# Parameter:
# config - json string with settings
def connect_tinkerforge(config):
    result = False
    # Define the globals for the tinkerforge connection and io4 bricklet
    global tfHost,tfPort
    global io4Uid,io4CallbackPeriod,io4StatusLED,io4Dev

    # Get the config parameter
    if DEBUG == True:
        print("Config:{}".format(config))

    # Parse the config string
    args = json.loads(config)
    # Assign to the vars
    tfHost = args["HOST"] if "HOST" in args else TF_HOST
    tfPort = int(args["PORT"]) if "PORT" in args else TF_PORT
    io4Uid = args["UID"] if "UID" in args else IO4_UID
    io4StatusLED = int(args["STATUSLED"]) if "STATUSLED" in args else IO4_STATUS_LED
    io4CallbackPeriod = IO4_CALLBACK_PERIOD
    io4Channels = args["CHANNELS"] if "CHANNELS" in args else ""

    # Check arguments
    if DEBUG == True:
        print("connect tinkerforge:")
host={},port={},uid={},callbackperiod={},channels={}".format(tfHost,tfPort,io4Uid,io4CallbackPeriod,io4Channels))

    try:
        # Create IP connection
        ipCon = IPConnection()
        # Create device object
        io4Dev = BrickletIO4V2(io4Uid, ipCon)
        # Connect to brickd
        ipCon.connect(tfHost,tfPort)
        # Check if the bricklet is reachable - else error is triggered
        # In case the bricklet is not reachable, it takes 1-2 seconds befor a response is given
        io4Dev.read_uid()
        # Set the status led
        io4Dev.set_status_led_config(io4StatusLED)
        # Set one or more channels
        if io4Channels != None:
            title = ""
            for item in io4Channels:
                channel = int(item["channel"])
                value = int(item["value"])
                direction = item["direction"]
                # set configuration is used to enable the of getting the value using get_configuration
                io4Dev.set_configuration(channel, direction, value)

```

```
# Register input value callback to function cb_input_value
if direction == "i":
    io4Dev.register_callback(io4Dev.CALLBACK_INPUT_VALUE, on_io4_callback)
    # Set period for input value for the channel with callback
    io4Dev.set_input_value_callback_configuration(channel, io4CallbackPeriod, False)
    title = "channel:{} value:{} direction:{}".format(channel,value,direction)
    result = True
    # result = { "status" : "OK", "title" : "{}".format(title) }
#
print("Tinkerforge Connected: host={},port={},uid={}".format(tfHost,tfPort,io4Uid))
# Disconnect
# ipCon.disconnect()
except:
    result = False
    # result = { "status" : "ERROR", "title" : "io4 failed. Check master & bricklet." }
# Print the result which is used by the dzVents Lua script
if DEBUG == True:
    print(json.dumps(result))

return result

# input("Press key to exit\n") # Use raw_input() in Python 2
# ipCon.disconnect()

# Callback function for the io4 channel(s) configured as input
# Updates the domoticz text device:
# Button pushed (example log entry & device text):
# 2020-02-29 12:09:08.468 MQTT: Topic: domoticz/in, Message:
#"command":"udevice","idx":108,"svalue":"IO4 changed:True, value:False, time:2020-02-29
# TF IO4 Status IO4 changed:True, value:True, time:2020-02-29 12:13:13
def on_io4_callback(channel, changed, value):
    if DEBUG == True:
        print("on_io4_callback:channel:{},changed:{},value:{}".format(channel,changed,value))
    # Select the channel
    if (channel == IO4_CHANNEL_1) and (changed == True):
        svalue = "IO4 changed:{}, value:{}, time:{}".format(changed,value,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
        set_domoticz_text_device(IDX_STATUS, svalue)

# Main
if __name__ == "__main__":
    print("{} starting...".format(VERSION))
    create_mqtt_instance()
```

# Appendix Domoticz Hints

## Start, Stop, Status

### Terminal commands

```
// Start, Stop, Status
sudo service domoticz.sh start
sudo service domoticz.sh stop
sudo service domoticz.sh status

// Enable Domoticz service at boot
sudo systemctl daemon-reload
sudo systemctl enable domoticz.service

// Disable Domoticz Service at boot
sudo systemctl disable domoticz.service

// Start, Stop Domoticz service using systemctl
sudo systemctl start domoticz.service
sudo systemctl stop domoticz.service
```

### Output Status Sample

```
domoticz.service - LSB: Home Automation System
  Loaded: loaded (/etc/init.d/domoticz.sh)
  Active: active (running) since Mon 2018-08-27 09:32:39 CEST; 16min ago
    Process: 31803 ExecStop=/etc/init.d/domoticz.sh stop (code=exited, status=0/SUCCESS)
    Process: 32093 ExecStart=/etc/init.d/domoticz.sh start (code=exited, status=0/SUCCESS)
      CGroupl: /system.slice/domoticz.service
              └─32100 /home/pi/domoticz/domoticz -daemon -www 8080 -sslww 443

Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.694 Status: Dom...z
Aug 27 09:32:39 139 domoticz[32098]: Domoticz is starting up....
Aug 27 09:32:39 139 domoticz[32100]: Domoticz running...
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Bui...6
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Sys...i
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Sta.../
Aug 27 09:32:39 139 domoticz.sh[32093]: domoticz: Domoticz is starting up....
Aug 27 09:32:39 139 systemd[1]: Started LSB: Home Automation System.
Hint: Some lines were ellipsized, use -l to show in full.
```

## Lost Username and Password

Open terminal session and login as User Pi

Stop Domoticz and wait 15 seconds to complete

```
sudo service domoticz.sh stop
```

Check Domoticz Status

```
sudo service domoticz.sh stop
```

Change to Domoticz Folder

```
cd domoticz
```

*Note*

The full path is /home/pi/domoticz

Start Domoticz with parameter no www password

```
sudo ./domoticz -nowwwpwd
```

# Create Backup

Ensure to make regular backups.  
Several options used.

## Option - Domoticz database to downloads folder

This option is the default provided by Domoticz.

Select GUI > Setup > Settings > Backup/Restore > Backup Database

This result is a database file in the downloads folder of the triggered device (device accessing the Domoticz Web UI).

## Option - Copy Domoticz folder to local device using WinSCP

This option uses [WinSCP](#).

Create a new site (if not done already) or select the Domoticz system.

Login and copy the Domoticz folder

```
/home/pi/domoticz
```

to local device folder, i.e. backups/domoticz.

## Option - Copy Domoticz folder to local device using scp

The scp (secure copy) command enables to copy files remote to the local device.

Example copying the Domoticz folder (on the Raspberry Pi) to the backup folder (on a Unix system running Ubuntu).

```
scp -r pi@domoticz-ip:/home/pi/domoticz /home/username/backup
```

The folder created is:

```
/home/username/backup/domoticz.
```

### Note

to get the real path of the backup folder, use:

```
realpath backup
```

issued from folder /home/username, resulting in output

```
/home/username/backup
```

# Change Log Level

```
Using username "pi".  
Linux DoPro 4.19.42-v7+ #1219 SMP Tue May 14 21:20:58 BST 2019 armv7l
```

```
Edit  
sudo nano /etc/init.d/domoticz.sh
```

```
Add the line:  
DAEMON_ARGS="$DAEMON_ARGS -loglevel normal,status,error,debug"
```

```
Save, followed by:  
sudo systemctl daemon-reload  
sudo service domoticz.sh restart
```

# Troubleshooting

## Device Widget turns yellow/red

Check the device status using URL with parameter type=devices and rid=idx.

### Example

Checking device named Windmesser (TFA Dostmann 30.3168 Windmeter) because the weather widget turned yellow.



From a browser run:

```
http://NNN.NNN.1.60:8080/json.htm?type=devices&rid=28
```

### Result JSON string (snippet)

```
{  
    "app_version" : "4.9999",  
    "result" : [  
        {  
            "BatteryLevel" : 0,  
            "Data" : "22.00;NNE;5;0;26.5;26.5",  
            "HardwareName" : "RFXtrx433e",  
            "HardwareType" : "RFXCOM - RFXtrx433 USB 433.92MHz Transceiver",  
            "SignalLevel" : 6,  
            "Used" : 1,  
            "idx" : "28"  
        }  
    ],  
    "status" : "OK",  
    "title" : "Devices"  
}
```

From the returned result, the **BatteryLevel** is 0.

It seems that the rechargeable battery is defect for this device.

The battery is recharged by a small solar panel on top of the device.

## Check Domoticz Status

### Command

```
sudo service domoticz.sh status
```

### Output Example

The log shows Domoticz is running OK.

```
domoticz.service - LSB: Home Automation System
  Loaded: loaded (/etc/init.d/domoticz.sh; generated)
  Active: active (running) since Fri 2020-01-31 13:04:15 CET; 3h 7min ago
    Docs: man:systemd-sysv-generator(8)
   Tasks: 19 (limit: 2077)
  Memory: 43.7M
  CGroup: /system.slice/domoticz.service
          └─503 /home/pi/domoticz/domoticz -daemon -www 8080 -sslwww 443

Jan 31 13:04:13 dodev systemd[1]: Starting LSB: Home Automation System...
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.463 Status: Domoticz V4.11666 (c)2012-2020
GizMoCuz
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.463 Status: Build Hash: 3630122d7, Date:
2020-01-31 08:45:14
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.464 Status: Startup Path:
/home/pi/domoticz/
Jan 31 13:04:15 dodev domoticz.sh[431]: domoticz: Domoticz is starting up....
Jan 31 13:04:15 dodev domoticz[457]: Domoticz is starting up....
Jan 31 13:04:15 dodev domoticz[503]: Domoticz running...
Jan 31 13:04:15 dodev systemd[1]: Started LSB: Home Automation System.
```

## GLIBC Version not found

After updating a beta version on Raspberry Pi running Raspbian Debian Strech, the Domoticz server was not reachable via webbrowser domoticz-ip:port.

A possible cause could be the GLIBC version is not right.

### How to find out?

1. Stop Domoticz
2. cd /home/pi/domoticz
3. sudo chmod +x ./domoticz.sh
4. sudo ./domoticz.sh start
5. If a message like below occurs then Domoticz requires newer GLIBC version.

```
/home/pi/domoticz/domoticz: /lib/arm-linux-gnueabihf/libm.so.6: version `GLIBC_2.27' not found
(required by /home/pi/domoticz/domoticz)
/home/pi/domoticz/domoticz: /lib/arm-linux-gnueabihf/libc.so.6: version `GLIBC_2.28' not found
(required by /home/pi/domoticz/domoticz)
```

### How to fix?

As the Raspberry Pi was not running the latest Raspbian Debian Buster version, decided to reinstall the system from scratch.

After installation, verified GLIBC version by running the command:

```
ldd --version | grep -i glibc
```

### Output

```
ldd (Debian GLIBC 2.28-10+rpi1) 2.28
```

*Note*

Installation steps are explained in chapter [Setup](#), but in a nutshell to get going:

1. Create new Raspberry Pi Image Raspbian Debian Linux Buster NOOBS v3.2.1
2. Reboot the Raspberry Pi and followed finalization steps
3. Set a fixed WLAN IP address (this is a MUST)
4. Created folders under /home/pi for later use, i.e. b4j,domoticz, java,python,tinkerforge
5. Installed Domoticz: curl -L install.domoticz.com | sudo bash
6. Installed mosquito: sudo apt-get install mosquitto mosquitto-clients
7. Reboot the Raspberry Pi
8. Access Domoticz GUI from browser

## Script Python Code 32512

Running Python scripts on a Raspberry Pi, ensure to set the line endings to LF (UNIX). Especially if developed on a Windows device using CRLF.

Check the Domoticz log after running the script.

This is an error message example from the Domoticz log (Raspberry Pi):

```
2020-08-16 18:47:48.129 Error: Error executing script command  
(/home/pi/domoticz/scripts/python/lcdi2cccontrol.py). returned: 32512
```

Action: Check the line endings and correct to LF (UNIX).

# HTTP API Requests

Several hints using Domoticz HTTP API requests.

## Rename Device

The name of the device is changed, but the device value is not changed.

In the request, ensure the special characters in the name are escaped.

In dzVents Lua use the function: domoticz.utils.urlEncode(string).

### Example: Rename text device with IDX=135

#### HTTP API Request

```
http://domoticz-ip:8080/json.htm?type=setused&idx=135&used=true&name=This+is+a+new+message
or
http://domoticz-ip:8080/json.htm?type=setused&idx=135&used=true&name=There%20is%20a%20new%20message
```

#### HTTP Response

```
{"status": "OK", "title": "SetUsed"}
```

### Example: Rename text device via switch and automation event.

If the switch state changes, a new name for the text device is set.

In the HTTP API request, the parameter “name” is encoded (using domoticz.utils.urlEncode(string)).

```
local IDX_SWITCH = 134
local IDX_MESSAGE = 135
local STATEON = "On"
local STATEOFF = "Off"

return {
    on = { devices = { IDX_SWITCH } },
    execute = function(domoticz, device)
        domoticz.log('Device ' .. device.name .. ' was changed', domoticz.LOG_INFO)
        local state = domoticz.devices(IDX_SWITCH).state
        -- Set a new name for the message device
        local newname = domoticz.utils.urlEncode(string.format("Switch state changed to %s", state))
        local urlRequest =
            string.format("http://127.0.0.1:8080/json.htm?type=setused&idx=%d&used=true&name=%s",
                IDX_MESSAGE, newname)
        domoticz.openURL({url = urlRequest})
    end
}
```

# Various

## CLI Commands

Some useful command line interface (CLI) commands.

### Check Ports in Use

```
sudo netstat -tulpn | grep LISTEN
tcp      0      0 0.0.0.0:4240          0.0.0.0:*          LISTEN
8175/brickd
tcp      0      0 0.0.0.0:22          0.0.0.0:*          LISTEN
465/sshd
tcp      0      0 0.0.0.0:1880          0.0.0.0:*          LISTEN
345/node-red
tcp      0      0 0.0.0.0:1883          0.0.0.0:*          LISTEN
428/mosquitto
tcp      0      0 0.0.0.0:4223          0.0.0.0:*          LISTEN
8175/brickd
tcp6     0      0 :::8080            ::::*              LISTEN
503/domoticz
tcp6     0      0 :::22              ::::*              LISTEN
465/sshd
tcp6     0      0 :::443             ::::*              LISTEN
503/domoticz
tcp6     0      0 :::1883             ::::*              LISTEN
428/mosquitto
tcp6     0      0 :::6144             ::::*              LISTEN
503/domoticz
```

*Note*

The list shows the Domoticz ports 8080 & 443 are in use.

### Remove ambitious CR characters from bash scripts.

To avoid the message:

```
/bin/bash^M: bad interpreter: No such file or directory...
```

The ^M has probably been created by editing the bash script under Wondows. Linux does not like ^M. It cannot find /bin/bash^M as it expects /bin/bash.

### Remove CR Chracters

```
sed -i -e 's/\r$//' *.sh
```

## String Length

The maximum length of strings is 200 bytes.

This applies to items like text sensors, user variables.

The size can be checked from the Domoticz database table Create Statements, i.e.  
VARCHAR(200).

### Examples

```
CREATE TABLE [LightingLog] ([DeviceRowID] BIGINT(10) NOT NULL, [nValue]  
INTEGER DEFAULT 0, [sValue] VARCHAR(200), [Date] DATETIME DEFAULT  
(datetime('now','localtime')), [User] VARCHAR(100) DEFAULT (''));
```

```
CREATE TABLE [UserVariables] ([ID] INTEGER PRIMARY KEY, [Name]  
VARCHAR(200), [ValueType] INT NOT NULL, [Value] VARCHAR(200), [LastUpdate]  
DATETIME DEFAULT(datetime('now', 'localtime')));
```

#### *Note*

The string is often used by the field sValue in conjunction with the nValue>

- nValue = Numeric - single value, INTEGER
- sValue = String – single or multiple values, VARCHAR(200)

## Hide Devices

To use but hide device from the GUI, place a dollar sign (“\$”) at the beginning of the device name. These hidden devices can still be used in scripts and for historical data.

# Appendix Domoticz Build Source

Sample commands for building Domoticz on a Raspberry Pi from the Domoticz Source. Followed the instructions from [here](#).

```
Swap File 100 to 500MB
sudo service dphys-swapfile status
sudo swapoff -a
free
sudo nano /etc/dphys-swapfile
    change 100 to 500
sudo swapon -a
free

Domoticz Build (took about 2 hours)
sudo apt-get install cmake make gcc g++ libssl-dev git libcurl4-gnutls-dev libusb-dev python3-dev
zlib1g-dev

mkdir boost

cd boost

wget https://dl.bintray.com/boostorg/release/1.68.0/source/boost_1_68_0.tar.gz
tar xfvz boost_1_68_0.tar.gz
cd boost_1_68_0/
./bootstrap.sh
./b2 stage threading=multi link=static --with-thread --with-system
sudo ./b2 install threading=multi link=static --with-thread --with-system
cd ../../
rm -Rf boost/
cd /home/pi
free

git clone https://github.com/domoticz/domoticz.git dev-domoticz
Cloning into 'dev-domoticz'...
remote: Counting objects: 59677, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 59677 (delta 9), reused 13 (delta 6), pack-reused 59655
Receiving objects: 100% (59677/59677), 206.16 MiB | 3.82 MiB/s, done.
Resolving deltas: 100% (43178/43178), done.
Checking out files: 100% (2784/2784), done.

cd dev-domoticz

git pull
Already up-to-date.

cmake -DCMAKE_BUILD_TYPE=Release CMakeLists.txt
-- The C compiler identification is GNU 6.3.0
-- The CXX compiler identification is GNU 6.3.0
...
-- Compiling Revision #9700
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/dev-domoticz

make
Scanning dependencies of target minizip
...
[100%] Linking CXX executable domoticz
```

```
[100%] Built target domoticz

ls
...
domoticz.sh
...

sudo ./domoticz
2018-09-12 16:18:10.882 Status: Domoticz V4.9999 (c)2012-2018 GizMoCuz
2018-09-12 16:18:10.882 Status: Build Hash: 9e8ea729, Date: 2018-09-09 18:27:57
2018-09-12 16:18:10.883 Status: Startup Path: /home/pi/dev-domoticz/
2018-09-12 16:18:11.292 Status: PluginSystem: Started, Python version '3.5.3'.
2018-09-12 16:18:11.299 Active notification Subsystems: gcm, http (2/13)
2018-09-12 16:18:11.304 Status: WebServer(HTTP) started on address: :: with port 8080
2018-09-12 16:18:11.318 Status: WebServer(SSL) started on address: :: with port 443
2018-09-12 16:18:11.319 Status: Proxymanager started.
2018-09-12 16:18:11.321 Starting shared server on: :::6144
2018-09-12 16:18:11.321 Status: TCPServer: shared server started...
2018-09-12 16:18:11.322 Status: RxQueue: queue worker started...
2018-09-12 16:18:13.323 Status: EventSystem: reset all events...
2018-09-12 16:18:13.323 Status: EventSystem: reset all device statuses...
2018-09-12 16:18:13.472 Status: PluginSystem: Entering work loop.
2018-09-12 16:18:13.480 Status: Python EventSystem: Initializing event module.
2018-09-12 16:18:13.481 Status: EventSystem: Started
2018-09-12 16:18:13.481 Status: EventSystem: Queue thread started...
2018-09-12 16:18:42.447 Status: Incoming connection from: NNN.NNN.1.3

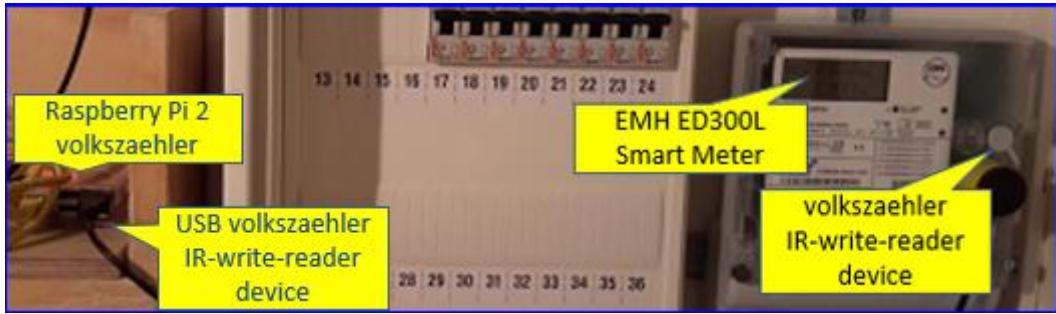
Updating Domoticz
Binary install update:
Use the Web Interface to update Domoticz (Setup->Check for Updates)

Manual update:
Stable: cd domoticz, ./updaterelease
Beta: cd domoticz, ./updatebeta
Source Update: cd domoticz, git pull, make -j 4 OR make
```

# Appendix volkszaehler Setup

Description of the **volkszaehler** with an **IR device**, connected to a Raspberry Pi. [volkszaehler.org](http://volkszaehler.org) is an Open Source (GPL license) Smart Metering Hard- and Software solution.

Ensure to read the "[howto get started](#)" first to get an understanding of the concept.



## Hardware

- Raspberry Pi Model 2 with WLAN and a 16GB SD Card.
- Power Meter EMH ED300L.

## Software

- Raspberry Pi Linux vz 4.14.62-v7+ #1134 (armv7l)
- volkszaehler Raspberry Pi image

# Setup Raspberry Pi

Download the volkszaehler image to a Windows PC, unpack the image, write the image using win32diskimager to a 16GB SD card. Stick the SD card in the Raspberry Pi and power on.

*Note*

This setup was done 2015 – things might have changed – checkout the volkszaehler website.

## Option 1 WLAN static IP address

The communication uses WLAN with a static network address.

Set a static IP address by changing /etc/network/interfaces and /etc/wpa\_supplicant/wpa\_supplicant.conf.

File /etc/network/interfaces:

Editing via **sudo nano /etc/network/interfaces**

```
auto lo
iface lo inet loopback
auto wlan0
iface wlan0 inet manual
    wireless-power off
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
    iface wlan0 inet static
        address domoticz-ip
        netmask 255.255.255.0
        broadcast NNN.NNN.0.255
        gateway domoticz-ip
```

**Edit sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf**

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="YOURSSID"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP
    group=CCMP
    psk="YOURPSK"
}
```

## Option 2 Ethernet static IP address

The Raspberry Pi volkszahler and the Raspberry Pi Domoticz production system are direct connected via an Ethernet cable. A standard cable is used, no twisted wires required.

The communication uses ETH0 with static network addresses on both sides.

Important: The ETH0 network address must be different from the WLAN network address.

Raspberry Pi volkszahler [Raspbian GNU/Linux 8 (jessie)] Linux 39 4.1.19-v7+ #858	Raspberry Pi Domoticz production [Raspbian GNU/Linux 8 (jessie)] Linux openHABianPi 4.9.35-v7+ #1014
eth0 inet addr:169.254.87.85 Bcast:169.254.255.255 Mask:255.255.0.0	eth0 inet addr:169.254.87.84 Bcast:169.254.255.255 Mask:255.255.0.0
Set the static Ethernet network Address sudo nano /etc/dhcpcd.conf	sudo nano /etc/dhcpcd.conf
Add the lines	
interface eth0 static ip_address=169.254.87.85	interface eth0 static ip_address=169.254.87.84

static routers=169.254.0.1	static routers=169.254.0.1
----------------------------	----------------------------

*Note*

- Determine the Linux version: uname -a to
- Determine the Raspberry Pi release: cat /etc/os-release to

Used option direct communication with ethernet cable.

# volkszaehler

## PowerMeter Setup

Powermeter: [EMH ED300L](#)

The powerpuls is captured using an [IR-write-reader device](#).

This device is connected via USB to the Raspberry Pi 2 Port /dev/ttyUSB22.

*Note*

- Check the connected USB devices with command **ls /dev/tty\***
- Ensure when changing the Raspberry Pi or adding USB devices, to use the same port for the IR-write-reader device ([read](#)).

Persistent USB devices are defined using:

```
sudo nano /etc/udev/rules.d/99-usb-serial.rules
```

### Add Line

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="00F2E620",
SYMLINK+="ttyUSB-VZ", MODE="0666"
```

### Check USB device information

```
sudo lsusb -v
```

To check detailed USB port information, use the command:

```
udevadm info --query=all --name=/dev/ttyUSB-VZ
```

### Output Example

```
P: /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
N: ttyUSB0
S: serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0
S: serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0
S: ttyUSB-VZ
E: DEVLINKS=/dev/serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0 /dev/serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0 /dev/ttyUSB-VZ
E: DEVNAME=/dev/ttyUSB0
E: DEVPATH=/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
E: ID_BUS=usb
E: ID_MODEL=CP2104_USB_to_UART_Bridge_Controller
E: ID_MODEL_ENC=CP2104\x20USB\x20to\x20UART\x20Bridge\x20Controller
E: ID_MODEL_FROM_DATABASE=CP210x UART Bridge / myAVR mySmartUSB light
E: ID_MODEL_ID=ea60
E: ID_PATH=platform-3f980000.usb-usb-0:1.5:1.0
E: ID_PATH_TAG=platform-3f980000_usb-usb-0_1_5_1_0
E: ID_REVISION=0100
E: ID_SERIAL=Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620
E: ID_SERIAL_SHORT=00F2E620
E: ID_TYPE=generic
E: ID_USB_DRIVER=cp210x
E: ID_USB_INTERFACES=:ff0000:
E: ID_USB_INTERFACE_NUM=00
E: ID_VENDOR=Silicon_Labs
E: ID_VENDOR_ENC=Silicon\x20Labs
E: ID_VENDOR_FROM_DATABASE=Cygnal Integrated Products, Inc.
E: ID_VENDOR_ID=10c4
E: MAJOR=188
E: MINOR=0
E: SUBSYSTEM=tty
E: TAGS=:systemd:
E: USEC_INITIALIZED=5005222
```

## USB Configuration

The USB port settings are 9600 Baud 8 n.

Set via command stty:

Minicom

The setup can also be done via [minicom](#)

```
sudo apt-get install minicom  
sudo minicom -s
```

Ensure port is set during boot

Add this line into /etc/rc.local (by sudo nano /etc/rc.local)

Check Data coming from the USB device (option 1):

```
cat /dev/ttyUSB-VZ | od -tx1
```

## Output Example:

```
00000000 2f 38 01 01 63 e4 c7 00 76 07 00 0a 00 07 db 75  
00000020 62 00 62 00 72 63 07 01 77 01 0b 09 01 4d 48 00  
...
```

Check Data coming from the USB device (option 2):

```
xxd </dev/ttyUSB-VZ
```

## Define Channels

To define the channels, reboot the RPi and access via a browser the volkszaehler Frontend using its ip address: <http://vz-ip>

Define Channels using the volkszaehler FrontEnd

```
Channel 1 =
{"version":"0.3","entity": {"uuid": "d14e9a80-8894-11e5-9dc8-298a9a1001ac", "type": "current", "active": true, "color": "aqua", "description": "Strom Direktverbrauch", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Direktverbrauch", "yaxis": "auto"}}

Channel 2 =
{"version": "0.3", "entity": {"uuid": "e460bec0-8894-11e5-b0d8-3f141d19092c", "type": "current", "active": true, "color": "aqua", "description": "Strom Gesamtverbrauch", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Gesamtverbrauch", "yaxis": "auto"}}

Channel 3 = 1.8.0
{"version": "0.3", "entity": {"uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "type": "electric meter", "active": true, "color": "#0033ff", "cost": 0.1992, "description": "Stromverbrauch Fuchsbau", "fillstyle": 0.2, "initialconsumption": 20, "public": true, "resolution": 1000, "style": "steps", "title": "Stromverbrauch", "yaxis": "auto"}}

Channel 4 = 16.7.0
{"version": "0.3", "entity": {"uuid": "a3c76600-8895-11e5-9432-333f81e04451", "type": "current", "active": true, "color": "aqua", "description": "Wirkleistung", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Wirkleistung", "yaxis": "auto"}}
```

## vzlogger Setup

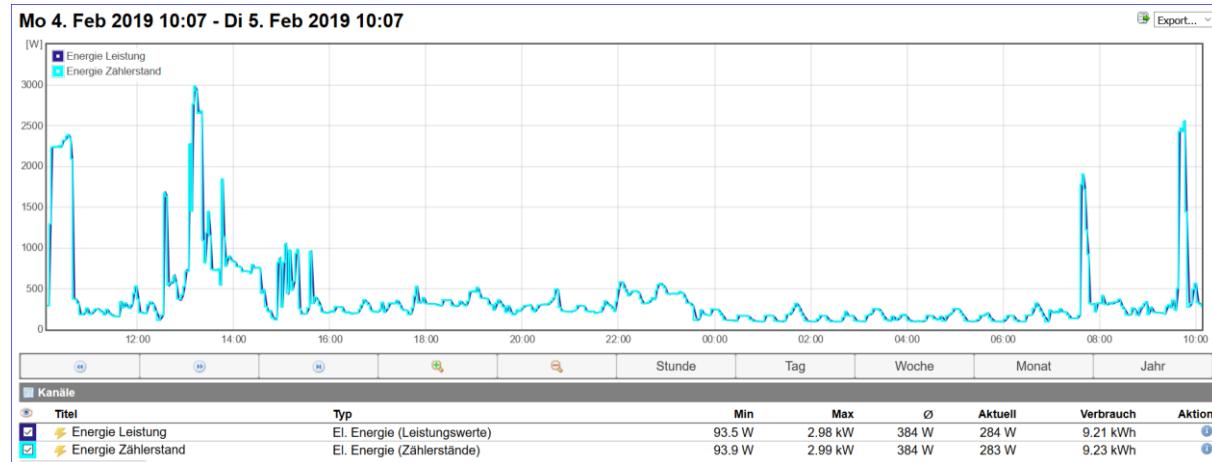
The defined channels must be defined in the file /etc/vzlogger.conf :

```
sudo nano /etc/vzlogger.conf
```

Two channels are defined "**Energie Leistung**" and "**Energie Zaehlerstand**":

```
/**  
 * vzlogger configuration  
 * Use properly encoded JSON with javascript comments  
 * http://wiki.volkszaehler.org/software/controller/vzlogger#configuration  
 * Online configuration editor: http://volkszaehler.github.io/vzlogger/  
 */  
  
{  
// General settings  
"daemon": true, // run periodically  
"verbosity": 0, // log (0=log_alert,1=log_error,3=log_warning,5=log_info,10=log_debug,15=log_fine)  
"log": "/var/log/vzlogger.log", // log file, optional  
"retry": 30, // http retry delay in seconds  
  
// Build-in HTTP server  
"local": {  
"enabled": false, // enable local HTTPd for serving live readings  
"port": 8080, // TCP port for local HTTPd  
"index": true, // provide index listing of available channels if no UUID was requested  
"timeout": 30, // timeout for long polling comet requests in seconds (0 disables comet)  
"buffer": 600 // HTTPd buffer configuration for serving readings, default -1  
// >0: number of seconds of readings to serve  
// <0: number of tuples to server per channel (e.g. -3 will serve 3 tuples)  
},  
  
"push": [],  
"meters" : [  
{  
"enabled" : true,  
"protocol" : "sml",  
"device" : "/dev/ttyUSB-VZ",  
"baudrate": 9600,  
"parity": "8n1",  
"channels":  
[  
{  
// Leistungswert  
"uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4",  
"middleware" : "http://localhost/middleware.php",  
"identifier" : "1-0:16.7.0"  
},  
{  
// Zaehlerstand  
"uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041",  
"middleware" : "http://localhost/middleware.php",  
"identifier" : "1-0:1.8.0"  
}  
]  
}  
]
```

volkszaehler Frontend <http://vz-ip> showing the two channels defined.



Clicking on the Aktion (i) icon provides the details for the two channels defined:

<p>"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4" "identifier": "1-0:16.7.0" Watt (Aktuelle Gesamtwirkleistung (P+ - P-))</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p><b>Details für Energie Leistung</b></p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>UUID</td> <td>958bce60-b342-11e8-b54f-bbec0573e1f4</td> </tr> <tr> <td>Middleware</td> <td>middleware.php</td> </tr> <tr> <td>Typ</td> <td>⚡ El. Energie (Leistungswerte)</td> </tr> <tr> <td>Cookie</td> <td></td> </tr> <tr> <td>Titel</td> <td>Energie Leistung</td> </tr> <tr> <td>Öffentlich</td> <td></td> </tr> <tr> <td>Farbe</td> <td>#311b92</td> </tr> <tr> <td>Stil</td> <td>Linien</td> </tr> <tr> <td>Füllgrad</td> <td>0</td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Aktiv</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;"><a href="#">Daten</a> <a href="#">Löschen</a> <a href="#">Bearbeiten</a> <a href="#">Schließen</a></p> </div>	Eigenschaft	Wert	UUID	958bce60-b342-11e8-b54f-bbec0573e1f4	Middleware	middleware.php	Typ	⚡ El. Energie (Leistungswerte)	Cookie		Titel	Energie Leistung	Öffentlich		Farbe	#311b92	Stil	Linien	Füllgrad	0	Achse	auto	Aktiv		<p>"uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041" "identifier": "1-0:1.8.0" Wh (Positive Gesamtwirkenergie (A+))</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p><b>Details für Energie Zählerstand</b></p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>UUID</td> <td>5b8ea2a0-b342-11e8-bec6-15c040e6d041</td> </tr> <tr> <td>Middleware</td> <td>middleware.php</td> </tr> <tr> <td>Typ</td> <td>⚡ El. Energie (Zählerstände)</td> </tr> <tr> <td>Cookie</td> <td></td> </tr> <tr> <td>Titel</td> <td>Energie Zählerstand</td> </tr> <tr> <td>Auflösung</td> <td>1000/kWh</td> </tr> <tr> <td>Öffentlich</td> <td></td> </tr> <tr> <td>Farbe</td> <td>aqua</td> </tr> <tr> <td>Stil</td> <td>Stufen</td> </tr> <tr> <td>Füllgrad</td> <td>0</td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Aktiv</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;"><a href="#">Daten</a> <a href="#">Löschen</a> <a href="#">Bearbeiten</a> <a href="#">Schließen</a></p> </div>	Eigenschaft	Wert	UUID	5b8ea2a0-b342-11e8-bec6-15c040e6d041	Middleware	middleware.php	Typ	⚡ El. Energie (Zählerstände)	Cookie		Titel	Energie Zählerstand	Auflösung	1000/kWh	Öffentlich		Farbe	aqua	Stil	Stufen	Füllgrad	0	Achse	auto	Aktiv	
Eigenschaft	Wert																																																		
UUID	958bce60-b342-11e8-b54f-bbec0573e1f4																																																		
Middleware	middleware.php																																																		
Typ	⚡ El. Energie (Leistungswerte)																																																		
Cookie																																																			
Titel	Energie Leistung																																																		
Öffentlich																																																			
Farbe	#311b92																																																		
Stil	Linien																																																		
Füllgrad	0																																																		
Achse	auto																																																		
Aktiv																																																			
Eigenschaft	Wert																																																		
UUID	5b8ea2a0-b342-11e8-bec6-15c040e6d041																																																		
Middleware	middleware.php																																																		
Typ	⚡ El. Energie (Zählerstände)																																																		
Cookie																																																			
Titel	Energie Zählerstand																																																		
Auflösung	1000/kWh																																																		
Öffentlich																																																			
Farbe	aqua																																																		
Stil	Stufen																																																		
Füllgrad	0																																																		
Achse	auto																																																		
Aktiv																																																			
<p><b>Bearbeiten von Energie Leistung</b></p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Titel</td> <td>Energie Leistung</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td></td> </tr> <tr> <td>Stil</td> <td>lines</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Auflösung</td> <td></td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;"><a href="#">Speichern</a> <a href="#">Abbrechen</a></p>	Eigenschaft	Wert	Titel	Energie Leistung	Öffentlich	<input checked="" type="checkbox"/>	Farbe		Stil	lines	Füllgrad		Achse	auto	Auflösung		Kosten		Initialverbrauch		<p><b>Bearbeiten von Energie Zählerstand</b></p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Titel</td> <td>Energie Zählerstand</td> </tr> <tr> <td>Auflösung</td> <td>1000</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td></td> </tr> <tr> <td>Stil</td> <td>steps</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </tbody> </table> <p style="text-align: center;"><a href="#">Speichern</a> <a href="#">Abbrechen</a></p>	Eigenschaft	Wert	Titel	Energie Zählerstand	Auflösung	1000	Öffentlich	<input checked="" type="checkbox"/>	Farbe		Stil	steps	Füllgrad		Achse	auto	Kosten		Initialverbrauch											
Eigenschaft	Wert																																																		
Titel	Energie Leistung																																																		
Öffentlich	<input checked="" type="checkbox"/>																																																		
Farbe																																																			
Stil	lines																																																		
Füllgrad																																																			
Achse	auto																																																		
Auflösung																																																			
Kosten																																																			
Initialverbrauch																																																			
Eigenschaft	Wert																																																		
Titel	Energie Zählerstand																																																		
Auflösung	1000																																																		
Öffentlich	<input checked="" type="checkbox"/>																																																		
Farbe																																																			
Stil	steps																																																		
Füllgrad																																																			
Achse	auto																																																		
Kosten																																																			
Initialverbrauch																																																			

## API Channel Properties

### Energie Leistung (Channel 1)

```
http://vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

```
{"version":"0.3","entity":{"uuid":"958bce60-b342-11e8-b54f-bbec0573e1f4","type":"powersensor","color":"#311b92","fillstyle":0,"public":true,"style":"lines","title":"Energie Leistung","yaxis":"auto"}}
```

### Energie Zaehlerstand (Channel 2)

```
http://vz-ip/middleware.php/channel/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json
```

```
{"version":"0.3","entity":{"uuid":"5b8ea2a0-b342-11e8-bec6-15c040e6d041","type":"electric meter","color":"aqua","fillstyle":0,"public":true,"resolution":1000,"style":"steps","title":"Energie Zaehlerstand","yaxis":"auto"}}
```

## vzlogger Service

Ensure the vzlogger service is correctly defined.

*Note*

Ensure the vzlogger command is using the parameter -c (see ExecStart).

```
cd /etc/systemd/system  
sudo nano vzlogger.service
```

Content:

```
[Unit]  
Description=vzlogger  
After=syslog.target network.target  
Requires=mysql.service  
[Service]  
ExecStart=/usr/local/bin/vzlogger -c /etc/vzlogger.conf  
ExecReload=/bin/kill -HUP $MAINPID  
StandardOutput=null  
[Install]  
WantedBy=multi-user.target
```

## vzlogger Commands

vzlogger Enable as a Service	systemctl enable vzlogger
vzlogger Start	systemctl start vzlogger OR <u>service vzlogger start</u>
vzlogger Stop	systemctl stop vzlogger OR <u>service vzlogger stop</u>
vzlogger Status	systemctl status vzlogger Example output:
vzlogger Daemon reload	systemctl daemon-reload
vzlogger <u>Debug</u>	vzlogger -f

## Test

For test purposes a `vzlogger.test` can be used to check data flow ([vzlogger configuration](#)).

```
{
  "retry" : 30, /* how long to sleep between failed requests, in seconds */
  "daemon": false,           /* run periodically */
  "verbosity" : 15,          /* between 0 and 15 */
  "log" : "/var/log/vzlogger.log",/* path to logfile, optional */
  "local" :
  {
    "enabled" : false, /* start the local HTTPD for serving live readings? */
    "port" : 80,      /* the TCP port for the local HTTPD */
    "index" : true,   /* provide an index listing of available channels? */
    "timeout" : 30,   /* timeout for long polling comet requests, 0 disables comet,sec*/
    "buffer" : 600   /* how long to buffer readings for the local interface, secs */
  },
  "meters" :
  [
    {
      "enabled" : true,      /* disabled meters will be ignored */
      "protocol" : "sml",    /* use 'vzlogger -h' for list of available protocols */
      "device" : "/dev/usb-ir-lesekopf0",
    },
    {
      "enabled" : true,      /* disabled meters will be ignored */
      "protocol" : "sml",    /* use 'vzlogger -h' for list of available protocols */
      "device" : "/dev/usb-ir-lesekopf1",
    }
  ]
}
```

### vzlogger Start:

```
vzlogger -c /etc/vzlogger.conf
```

### vzlogger Check status:

```
systemctl status vzlogger
```

### Output Example:

```
? vzlogger.service - vzlogger
   Loaded: loaded (/etc/systemd/system/vzlogger.service; enabled)
   Active: active (running) since Fri 2015-11-13 11:55:53 CET; 13min ago
     Main PID: 670 (vzlogger)
        CGroup: /system.slice/vzlogger.service
                  +-670 /usr/local/bin/vzlogger -c /etc/vzlogger.conf
Nov 13 11:55:53 raspberrypi systemd[1]: Started vzlogger.
```

## API volkszaehler Hints

API Reference: <http://wiki.volkszaehler.org/development/api/reference>

### API Obtain Configuration Request

`http://vz-ip/middleware.php/entity/958bce60-b342-11e8-b54f-bbec0573e1f4.json`

Result

```
{"version": "0.3", "entity": {"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4", "type": "powersensor", "color": "#311b92", "fillstyle": 0, "public": true, "style": "lines", "title": "Energie Leistung", "yaxis": "auto"}}
```

### API Channel Properties

`http://vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json`

Result

```
{"version": "0.3", "entity": {"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4", "type": "powersensor", "color": "#311b92", "fillstyle": 0, "public": true, "style": "lines", "title": "Energie Leistung", "yaxis": "auto"}}
```

### API Entities

`http://vz-ip/middleware.php/capabilities/definitions/entities.json`

### API Read Data Requests

*Note*

Relative Dateformats = <http://de.php.net/manual/en/datetime.formats.relative.php>

### Get All Data

`http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json`

### Get One Record

`http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?tuples=1`

Result

```
{"version": "0.3", "data": {"tuples": [[1447316802613, 246.491, 19046]], "uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "from": 144726000365, "to": 1447316802613, "min": [1447316802613, 246.49134831111], "max": [1447316802613, 246.49134831111], "average": 246.491, "consumption": 3752.3, "rows": 1}}
```

### Get the last value

`http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?from=now&to=now`

Result

```
{"version": "0.3", "data": {"tuples": [[1447317119885, 201.681, 1], [1447317121700, 198.347, 1]], "uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "from": 1447317118100, "to": 1447317121700, "min": [1447317121700, 198.34710743513], "max": [1447317119885, 201.68067227331], "average": 200, "consumption": 0.2, "rows": 3}}
```

**API Logging**

```
http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?ts=1284677961150&value=12
```

Using the volkzaehler Frontend to display data

<http://vz-ip/frontend/?uuid=23844c00-8895-11e5-a642-6d8ce36c9a44>

# Appendix Tools

In progress developing some tools for managing Domoticz.

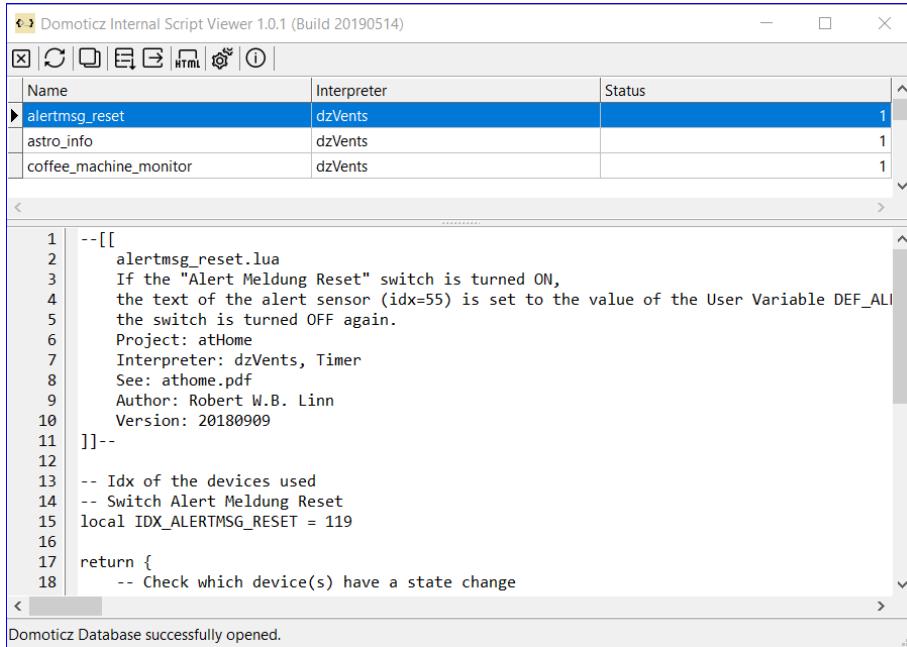
## Domoticz Internal Script Viewer

### Purpose

- To view offline the Domoticz internal scripts (focus on dzVents scripts) stored in the Domoticz database (domoticz.db), which are created using the Domoticz GUI Event editor (Setup > More Options > Events – My automation scripts)
- To export the scripts to the clipboard, text file or HTML file

[Download](#) the application from GitHub.

### Screenshot



# Domoticz MQTT Logger

## Purpose

To view Domoticz MQTT messages published, by subscribing to the topic “domoticz/out”.

The MQTT logger

- logs (in reverse order) the last 5 MQTT message payloads from the Domoticz Production or Development server.
- displays in a separate group the last entry Idx, Name and Payload.
- enables to analyze MQTT message in detail:
  - check the values from properties nvalue, svalue, svalue1 etc.
  - explore the properties of a new device, to use in scripts or apps.

More information about [MQTT](#).

## Solution

The solution is based on a Node-RED flow, subscribing to Domoticz MQTT messages published by topic “domoticz/out” using the Domoticz Hardware “MQTT Client Gateway with LAN interface” (see GUI > Settings > Hardware).

The screenshot shows the Domoticz hardware configuration page for an MQTT Client. The table at the top lists one entry:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
10	MQTT Client	Yes	MQTT Client Gateway with LAN interface	Domoticz-ip-address	1883	Disabled

Below the table is a form for editing the MQTT Client settings:

- Enabled:**
- Name:** MQTT Client
- Type:** MQTT Client Gateway with LAN interface
- Data Timeout:** Disabled
- Remote Address:** Domoticz-ip-address
- Port:** 1883
- Username:** [redacted]
- Password:** [redacted]
- Publish Topic:** out
- Select the Topic(s) Domoticz will use to publish outgoing messages.  
Flat - publish outgoing messages on topic domoticz/out.  
Hierarchical - publish outgoing messages on topic domoticz/out/\${floorplan name}/\${plan name}.  
Combined - Use both Flat and Hierarchical topic schemes.  
None - disable outgoing messages.
- Note that Hierarchical only reports sensor updates for sensors that are placed on a floorplan/plan.
- CA Filename:** [redacted]

## Node-RED Dashboard UI

The dashboard has its own tab MQTT Logger with the three groups Log Entries, Last Entry and Settings.

The screenshot shows the MQTT Logger dashboard with the following sections:

- Log Entries:** Displays a list of log entries. The first entry is:
 

```
18:28:42 pidodev CPU Usage (39)
      {"Battery":255, "RSSI":12, "description":"",
      "dtype":"General", "hwid":"2",
      "id":"0000044D", "idx":39, "name":"pidodev
      CPU Usage", "nvalue":0, "stype":"Percentage",
      "svalue1":"12.05", "unit":1,
      "msgreceived":"18:28:42"}
```
- Last Entry:** Shows the details of the most recent log entry:
 

<b>Idx</b>	39
<b>Name</b>	pidodev CPU Usage
<b>Payload</b>	<pre>{"Battery":255, "RSSI":12, "description":"",       "dtype":"General", "hwid":"2",       "id":"0000044D", "idx":39, "name":"pidodev CPU Usage",       "nvalue":0, "stype":"Percentage",       "svalue1":"12.05", "unit":1,       "msgreceived":"18:28:42"}</pre>
- Settings:** Includes a Broker dropdown set to "DoDev", an "Idx Filter" input field containing "115," (with a comma), a "Use Filter" toggle switch (disabled), and a "Version" label showing "v2.20 (Build 20200313)".

Recommend setting the flow disabled when not using, as rather high message traffic could take place.

### Notes

It is easy to build flows in Node-RED for these kind of utility functions. Scribbled first the concept on paper, some trial-and-error flows on payload handling and lastly use the Dashboard UI nodes to build the UI.

### Description

#### Group Log Entries

Display the last 5 messages logged - rolling entries (the number of displayed messages is defined in the function node “Add msg to flow.log”).

The flow uses data stored as flow context for the

- log = an array for the messages displayed (flow.log)
- idx filter content = comma separated string to filter device idx (flow.idxfilter)
- use the idxfilter = flag 0 or 1 (flow.usefilter)

Each entry has a timestamp - taken from the msg.payload property msgreceived which is added to the Domoticz payload after receiving the message.

The msg.payload is converted to a string to enable word wrap - after a comma, a space is added.

### *Hold Logging*

Hold the logging - useful if want to view in detail or copy the listed messages.

### *Clear Log*

Clear the listed log entries.

### **Group Last Entry**

Display the Idx, Name and Payload of the last message received.

The msg.payload is converted to a string to enable word wrap - after a comma, a space is added.

### **Button Copy Message**

Copy the content of the message field to the clipboard. Example content:

```
{"Battery":100,"RSSI":5,"description":"","dtype":"Temp + Humidity","id":"10254","idx":40,"name":"WZ Temperatur","nvalue":0,"stype":"Cresta, TFA TS34C","svalue1":"21.6","svalue2":"43","svalue3":"1","unit":1,"msgreceived":"17:56:24"}
```

### **Group Settings**

#### *Broker*

Select the MQTT broker. Two options: Domoticz Production (DoPro) or Domoticz Development (DoDev) broker.

#### *Idx Filter*

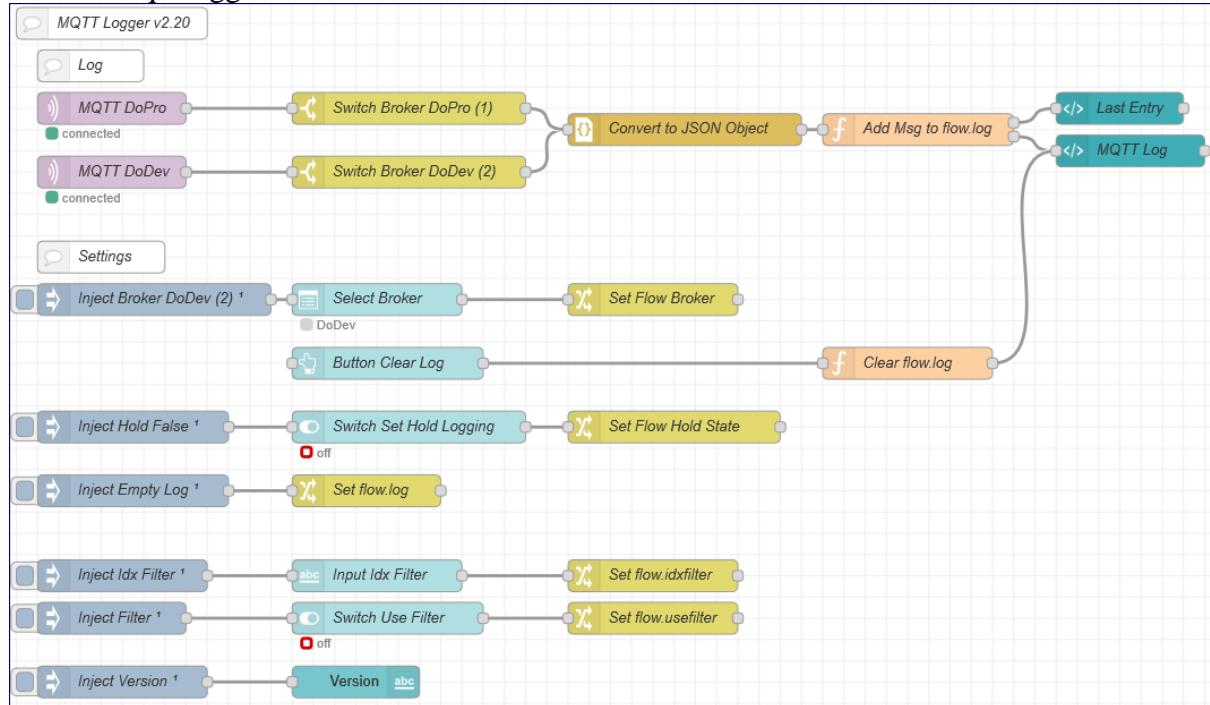
Set a filter containing 1 or more devices idx. The input must end with a comma.

#### *Use Filter*

Use the filter defined. If not switched on, then all messages are listed.

## Node-RED Flow

Source: mqtt-logger.flow



## Node-RED Flow Brief Description

The flow makes use of the [Node-RED Dashboard Module](#).

The mqtt-in node is the entry point from which the message payload is routed to Node-RED Dashboard nodes. The MQTT broker used is the Domoticz Production server. And subscribes to the topic “domoticz/out”.

The function “Add Msg to flow.log”, which is the core of this solution, manages the new MQTT incoming message entries by creating an array with max. 5 msg.payload entries. The array is used by the ui\_template node “MQTT Log” to display in reverse order the messages – using form. The function also sends as a second’s output, the last message which is displayed in a form by the ui\_template “Last Entry”. Both templates uses javaScript scope function to create the field content.

The setting options are setting flow context which are used by the previous described core function.

If want to explore more in detail, lookup at the source of the flow or import the flow.

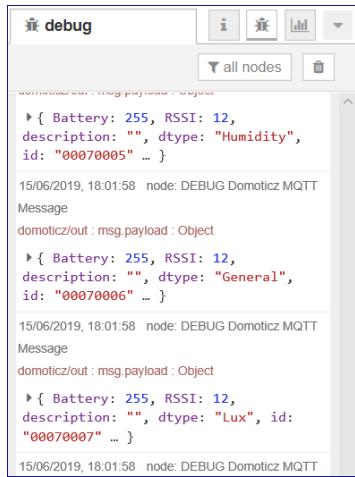
## Alternative Logger

An alternative, rather simple Nod-RED flow, to see Domoticz MQTT messages in JSON format, is to create a flow with three nodes:



Mqtt in	Json	Debug
Listen to topic “domoticz/out” from the Domoticz system	Convert the message payload to a Javascript object	Log the full message object (i.e. msg.payload) or selective property (i.e. msg.payload.idx or msg.payload.svalue1)

## Output Debug Node (as full message payload) (see Browser > Node-RED development tab)



### Example Log Entry

```
{
  "Battery": 255, "RSSI": 12, "description": "", "dtype": "General", "id": "00070006", "idx": 33, "name": "IAQM Pressure", "nvalue": 0, "stype": "Barometer", "svalue1": "1011", "svalue2": "0", "unit": 6
}
```

# RaspberryMatic Statelist

## Purpose

To obtain the list of devices and datapoints , with selected properties, from the RaspberryMatic system.

The focus is on getting the id of the devices and datapoints configured in the CCU.

The ids are used by RaspberryMatic scripts and Domoticz Automation Events (dzVents).

## Solution

The RaspberryMatic addon XML-API is used to get the data.

Send a HTTP XML-API request to the Raspberrymatic system, with the statelist script (statelist.cgi) as parameter.

Example HTTP API Request:

```
http://CCU-IP/config/xmlapi/statelist.cgi
```

The HTTP response is an XML Tree (as plain text; XML is used as information storage) with all devices and datapoints.

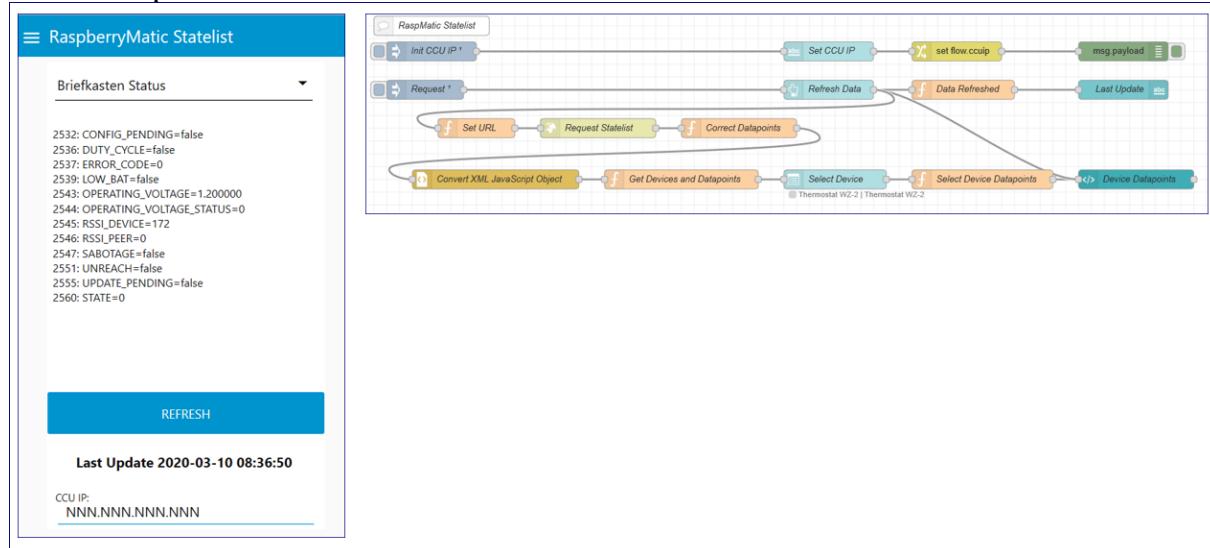
The XML Tree is parsed to get the device name(s) & id plus the according datapoint(s) id, type, and value.

## Node-RED

Versions: Node-RED v1.0.4 with the add-on node-red-dashboard 2.19.4.

## Dashboard UI and Flow

Source: raspmatic\_statelist.flow



### In Brief

The dashboard ui has a device dropdown list which populates a html list with selected device datapoints. The button refresh obtains the latest statelist. The input field CCU IP sets the ip address of the CCU (running on the RaspberryMatic system).

The http request node sends an HTTP XML-API request to the CCU to obtain the statelist in XML format.

The XML tree is converted to a JavaScript object which is parsed to arrays holding JSON formated entries for the devices and datapoints.

These arrays are used for the UI.

The UI is based on the Node-RED Dashboard [Info](<https://github.com/node-red/node-red-dashboard>).

The devices are added, from the devices array, to a ui\_dropdown node.

Selecting a device, lists the assigned datapoints, from the datapoints array, in a ui\_template node as list.

A button Refresh requests a new statelist from the CCU.

## HTTP XML-API Response

The response contains the statelist with devices, channels and datapoints.

Example with data extract.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device config_pending="false" unreach="false" ise_id="2530" name="Briefkasten Status">
  <channel ise_id="2531" name="Briefkasten Status:0" operate="true" visible="true" index="0">
    <datapoint ise_id="2532" name="HmIP-RF.0.000DA498D5859:0.CONFIG_PENDING" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
    <datapoint ise_id="2536" name="HmIP-RF.0.000DA498D5859:0.DUTY_CYCLE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
    <datapoint ise_id="2537" name="HmIP-RF.0.000DA498D5859:0.ERROR_CODE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="8" value="0" type="ERROR_CODE"/>
    <datapoint ise_id="2539" name="HmIP-RF.0.000DA498D5859:0.LOW_BAT" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="LOW_BAT"/>
    <datapoint ise_id="2543" name="HmIP-RF.0.000DA498D5859:0.OPERATING_VOLTAGE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="4" value="1.20000" type="OPERATING_VOLTAGE"/>
    <datapoint ise_id="2544" name="HmIP-RF.0.000DA498D5859:0.OPERATING_VOLTAGE_STATUS" operations="5"
      timestamp="1583826706" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
    <datapoint ise_id="2545" name="HmIP-RF.0.000DA498D5859:0.RSSI_DEVICE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="8" value="180" type="RSSI_DEVICE"/>
    <datapoint ise_id="2546" name="HmIP-RF.0.000DA498D5859:0.RSSI_PEER" operations="5" timestamp="0"
      valueunit="" valuetype="8" value="0" type="RSSI_PEER"/>
    <datapoint ise_id="2547" name="HmIP-RF.0.000DA498D5859:0.SABOTAGE" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="SABOTAGE"/>
    <datapoint ise_id="2551" name="HmIP-RF.0.000DA498D5859:0.UNREACH" operations="5"
      timestamp="1583826706" valueunit="" valuetype="2" value="false" type="UNREACH"/>
    <datapoint ise_id="2555" name="HmIP-RF.0.000DA498D5859:0.UPDATE_PENDING" operations="5"
      timestamp="1582992737" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
  </channel>
  ... more channels
</device>
<device config_pending="false" unreach="true" ise_id="3597" name="Eingang Monitor">
  ... channels & datapoints
</device>
... more devices
</stateList>
```

## Function Node Get Devices and Datapoints

The purpose of this function is to

- return a json array with devices used for the ui\_dropdown
- set flow context "datapoints" for the ui\_template to display the datapoints for the selected device id

*Notes*

- Example device entry: deviceobj["Briefkasten Status"] = 2530;
- Example datapoint entry:  
device:"2530",datapoint:{ "id":"2532","type":"CONFIG\_PENDING","value":false,"name":"HmIP-RF.0000DA498D5859:0.CONFIG\_PENDING"}
- the second array below is not used as replaced by the flow context "datapoints" - but is kept in case any enhancements planned

```

statelist = msg.payload.stateList;
devices = statelist.device
arrdevices = [];
arrdatapoints = [];
devices.forEach(function(device) {
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename]=deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            channeldatapoints.forEach(function(datapoint){
                obj = datapoint.$;
                if (obj !== undefined) {
                    dpobj = {
                        "device":deviceise_id,
                        "datapoint": {
                            "id":obj.ise_id,"type":obj.type,"value":obj.value,"name":obj.name
                        }
                    };
                    arrdatapoints.push(dpobj);
                }
            });
        }
    });
});
flow.set("datapoints",arrdatapoints);

msgdevices = {};
msgdevices.options = arrdevices;

// not used
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices,msgdatapoints];

```

## Function Node Select Device Data

The purpose of this function is to

- get all datapoints for the selected device using the device id (ise\_id)
- return an array with datapoint entries: id: type=value

### Notes

- The type is the short name of the datapoint
- The array is used for the ui\_template html list

```
const DEBUG = false;
// get the device id, i.e. 2530 for the selected device
var deviceid = msg.payload;
if (DEBUG) node.warn("Selected: " + deviceid);

if (deviceid === null) return [];

// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");

devicedatapoints = [];
datapoints.forEach(function(datapoint){
  if (datapoint.device == deviceid) {
    dp = datapoint.datapoint;
    var entry = dp.id + ":" + dp.type + "=" + dp.value;
    devicedatapoints.push(entry);
    if (DEBUG) node.warn("Added: " + entry);
  }
});

msg.payload = devicedatapoints;
return msg;
```

Last page initially left blank