

Domoticz Home Automation Workbook

Exploring Domoticz Home Automation System

Robert W.B. Linn

16.12.2019

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	9
Components	11
Hardware	11
Software	13
External Services	13
Setup	14
Development Device	14
Raspberry Pi	14
Installation	14
Network Static IP Address	15
Wired Network	16
WLAN Power Save Mode	17
Raspian Check Version & Update	18
Samba	19
Persistent USB Devices	21
Domoticz	23
Installation	23
Settings	24
Folders	25
Events	25
Functions	26
Introduction	26
Air Pressure (BME280)	27
Purpose	27
Circuit	27
I2C	28
Domoticz Configuration	29
Barometer Widget	30
Automation Script	30
Alert Message	31
Purpose	31
Solution	31
Automation Script	33
Alert Message Reset	34

Database Tables	34
Ambient Light (BH1750).....	36
Purpose.....	36
I2C	36
Parts Needed.....	36
Wiring	36
Device Setup Domoticz	36
Device Setup ESP Easy.....	37
Trigger Switch Light.....	38
Android App atHome	40
Purpose.....	40
Solutions.....	40
Anemometer (TFA 30.3168).....	42
Purpose.....	42
Device	42
Setup	42
Blinds (Somfy RTS Pure).....	43
Purpose.....	43
Domoticz Configuration	43
Dashboard & Switches Widget	44
MQTT Messages	44
Coffee Machine Monitor.....	45
Purpose.....	45
Solution	45
Automation Script.....	46
Contact Detection (433 Mhz).....	47
Purpose.....	47
Solution Sensor PB-62R	47
Domoticz Configuration	48
Alarm Detection	48
Automation Script.....	49
Alarm State Reset Switch.....	50
MQTT	51
Days-To-Go.....	53
Purpose.....	53
Solution	53
Atomation Script.....	54

Electric Usage	56
Purpose.....	56
Solution	56
Electric Usage House (volkszaehler).....	57
Purpose.....	57
Solution	57
Domoticz Configuration	58
Polling	59
Automation Script.....	61
MQTT	62
HTTP API Request.....	62
Log Examples.....	64
Database Table	64
volkszaehler.....	66
Electric Usage Rooms/Devices	81
Purpose.....	81
homematicIP HMIP-PSM (Plugin).....	81
Revolt SF-436.m.....	82
Email Control	86
Purpose.....	86
Solution	87
Event Monitor	94
Purpose.....	94
Domoticz Configuration	94
Automation Script.....	95
Maintenance	96
Indoor Air Quality (Plugin, Tinkerforge)	97
Purpose.....	97
Solution	97
Info Message	99
Purpose.....	99
Solution	99
Automation Script.....	99
Hardware Monitor Raspberry Pi	100
Purpose.....	100
Solution	101
Solution RPi volkszaehler	102

Solution RPi Domoticz Production	105
Philips Hue	106
Purpose	106
Hue Bridge Configuration.....	106
Hue Devices	106
Hue Light Add New.....	107
Hue Control Tests.....	108
Hue Timed Switch Lights	110
Hue Lights Control Node-RED	112
Other Information	113
Postbox Notifier (RaspberryMatic)	114
Purpose	114
Solutions.....	114
Solution Alert	116
Solution Alert + Switch	119
Solution Alert + Switch + Voltage	123
Radiator Thermostat HmIP-eTRV (Plugin).....	128
Purpose.....	128
Solution	128
Installation.....	128
Enhancement Push Off Button	131
Timers	132
Raspberry Pi Monitoring.....	133
Purpose.....	133
Solution	133
Monitoring.....	134
River Elbe Tide	136
Purpose.....	136
Solution	136
Node-RED Flow	137
Domoticz Configuration	137
RFXCOM RFXtrx433E.....	138
Purpose.....	138
Prepare RFXtrx433E.....	138
Domoticz Configuration	140
Troubleshooting.....	141
Soil Moisture (Plugin, Tinkerforge)	142

Purpose	142
Solution	142
Stock Quotes	143
Purpose.....	143
Domoticz Configuration	144
Solution Automation Script.....	145
Solution Node-RED.....	147
Notifications	151
Enhancements.....	153
Temperature & Humidity 433Mhz	154
Purpose.....	154
Device	154
Setup	154
Timers	155
Purpose.....	155
Rule.....	155
Device Timers	155
Timerplans.....	158
Automation Events	159
Waste Calendar.....	162
Purpose.....	162
Devices Configuration	163
Concept	163
CSV Input File.....	163
Automation Script.....	164
Python CSV File Generation Script.....	166
Webfrontend Customized	169
Purpose.....	169
Web Front End Sample.....	169
Concept	169
HTML & JavaScript	171
Webradio Volumio	174
Purpose.....	174
Volumio Setup.....	175
Solution Options	175
Solution Automation Script.....	177
Solution Node-RED.....	181

Favorites List.....	186
Wind TFA 30.3168.....	187
Purpose.....	187
Solution	187
Automation Script.....	188
User Variables	189
Syntax	189
List	189
SQL.....	189
Python	190
Usage in Scripts.....	191
Usage in Browser Interactive	192
Usage in Node-RED	193
Usage in Node-RED Dashboard.....	194
Example Ambient Light Threshold	194
Custom Icons.....	196
Explore.....	200
API Interaction	200
Purpose.....	200
Option HTTP.....	200
Option MQTT.....	202
Recommendation	204
Example Syntax HTTP vs MQTT	204
Events - Overview	207
Purpose.....	207
Event Execution Order.....	207
Event Scheduling & Trigger	208
Event Database Tables.....	208
Event Script Viewer.....	208
Event Development.....	209
Event Sample Basement Humidity Monitor	210
Events - dzVents.....	218
Device Properties.....	218
Script	219
Property List.....	220
External Modules.....	220
Shared Helper Functions.....	224

ESP8266.....	225
Purpose.....	225
Experiment	225
Prototype	225
Prepare Arduino IDE	225
Parts Needed.....	226
Wiring	226
Circuit	226
Source Code	227
ESP Easy	228
Purpose.....	228
Overview Experiments.....	228
Flash ESP	229
Update ESP	229
Domoticz Controller	230
Rules	231
GPIO Commands.....	232
Experiments.....	233
Domoticz switching ESP LED	233
ESP Ambient Light to Domoticz.....	235
ESP controlling Hue Brightness	241
ESP BMP280 to Domoticz.....	246
MQTT	249
Purpose.....	249
Install.....	249
Mosquitto	251
Python	253
Node-RED.....	259
Purpose.....	259
Installation.....	259
Update	260
Access Flows & Dashboard UI	261
Start, Stop, Log.....	261
Manage Node Packages	261
Folder Locations	262
Flow Domoticz MQTT Messages	262
Flow Raspberry Pi CPU Usage	264

Node-RED as MQTT Publisher	265
Dashboard UI	265
Python Plugin Development	270
Purpose.....	270
Hints.....	271
Plugin Traffic Light (Tinkerforge Building Blocks)	273
RaspberryMatic	280
Purpose.....	280
Solution	280
Installation.....	280
Addon XML-API.....	281
Addon CUxD.....	298
Plugin Considerations	303
CCU Device(s) Information	304
CCU Devices homematicIP	306
Experiments.....	307
SQL.....	311
Purpose.....	311
SQL Command Examples.....	311
SQLite3 Package	319
Appendix Domoticz Hints	323
Start, Stop, Status	323
Lost Username and Password	323
Create Backup	324
Change Loglevel.....	324
Troubleshooting.....	325
Device Widget turns yellow/red.....	325
Appendix Domoticz Build Source	326
Appendix volkszaehler Setup.....	328
Hardware.....	328
Software	328
Setup Raspberry Pi	329
Option 1 WLAN static IP address	329
Option 2 Ethernet static IP address.....	329
volkszaehler.....	331
PowerMeter Setup	331
USB Configuration	332

Define Channels	333
vzlogger Setup	334
Appendix Tools	341
Domoticz Internal Script Viewer	341
Purpose	341
Domoticz MQTT Logger	342
Purpose	342
Solution	342
Alternative Logger	344

Purpose

To build a Home Automation Solution, running on a Raspberry Pi with [Domoticz](#) Home Automation System.

Objectives

- ✓ Explore & learn Domoticz & Scripting.
- ✓ Build a Home Automation Solution.
- ✓ Write up experiences during development ... and enhance further.

Notes

- ❖ This is a working document
 - ... solution changes, idea's & todo's will probably never cease to exist 😊.
- ❖ There might be better solutions for what is shared
 - ... updates or changes will happen as progressing with the learning curve.
- ❖ Source code for the scripts & flows not fully shared in this document or have changed
 - ... check the [GitHub](#) scripts folder for the latest full source code.
- ❖ Automation scripts developed as dzVents Lua script events
 - ... very few in Python (to be replaced).
- ❖ Domoticz Hardware Plugin development in Python.
- ❖ Some functions make initially use of Node-RED flows
 - ... but target is to replace by dzVents scripts if possible.
- ❖ Functions that have been replaced by another solution are kept in this document as reference – might be of use.
- ❖ Screenshots shared might not be update 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.
- ❖ Abbreviations: GUI = Domoticz UI in web browser, HMIP = homematicIP, TF = TinkerForge

Functions

- Display temperature & humidity measured in rooms.
- Charts for selective weather items, room temperature & humidity.
- Control Somfy roller shutters with RTS motors in rooms.
- Philips Hue Lighting System control via Hue Bridge for ZigBee devices.

- HomeMatic (RaspberryMatic) CCU3 integration with variety of devices.
- Security door & window wireless contact detectors.
- Information on key dates (calendar type information).
- MQTT subscribe & publish messages to trigger actions or information.
- Raspberry Pi system information with charts and threshold email notification.
- Electricity power & energy for the house from “volkszaehler” with charts.
- Electricity power & energy for selected devices / rooms.
- Coffee machine monitor – start and end time, info message.
- Control Volumio music player whilst listening to web radio.
- Monitor stock quotes.
- Ambient light (from ESP8266 running ESP Easy) with threshold.
- Hue Light controlled via ESP8266 with slide potentiometer & 4-digit-7-segment-display.
- Event monitor for selected devices.
- Hardware Monitor for the Raspberry Pi's used.
- Custom icons.
- Soil Moisture monitor for plants (plugin).
- Air Quality Monitor (plugin).

Explore How to Use/Develop

- Domoticz running on a Raspberry Pi (setup, configure).
- Scripting Python, Lua, dzVents, JavaScript.
- Python Plugin Development (mainly with Tinkerforge Building Blocks).
- Domoticz JSON/API interaction.
- MQTT messaging.
- Node-RED as an alternative script engine and User Interface.
- SQLite to read the Domoticz Database.
- ESP8266, ESP Easy.
- 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 CCU3 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.

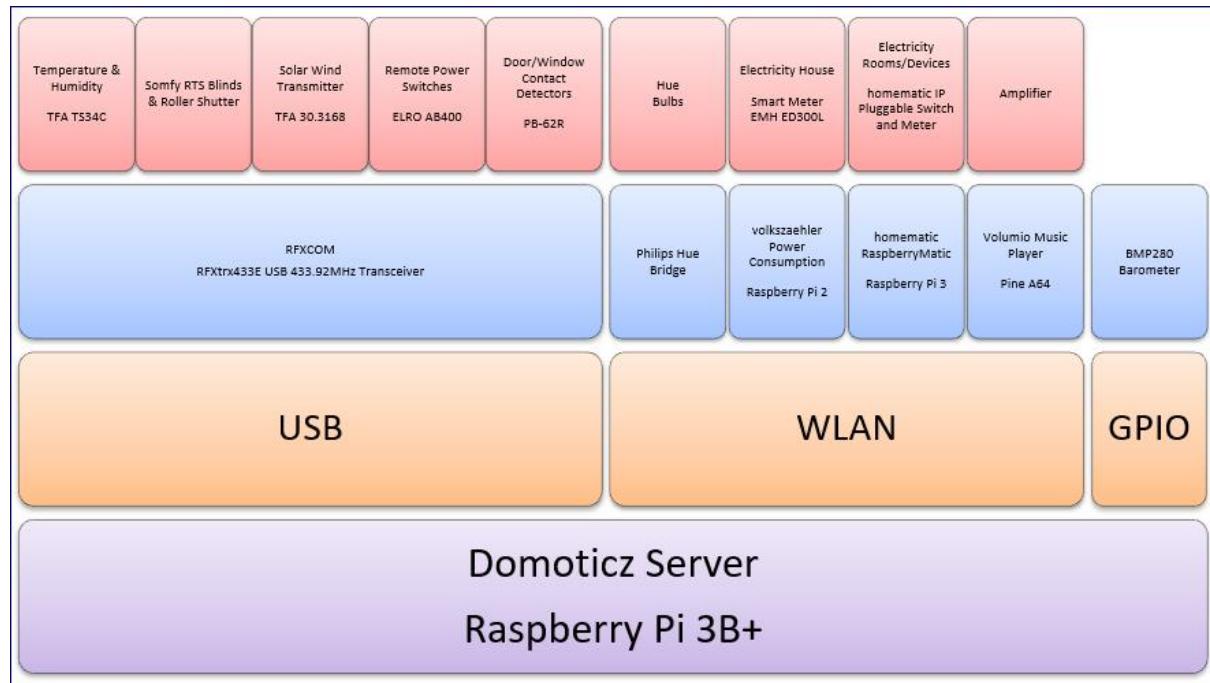
Drawings are created with [Fritzing](#).

Components

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

Hardware

Overview of the various hardware used.



- Raspberry Pi 3B+ v1.2 (Domoticz server)
- 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 & rollershutter)
 - 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
- SmartMeter EMH ED300L connected to a Raspberry Pi 2 running volkszaehler
- RaspberryMatic CCU3 (Raspberry Pi 3B+) with variety of connected devices like HMIP-PSM, HMIP-eTRV-2, HMIP-SDWO
- Volumio Music Player (Pine A64 1GB)
- Raspberry Pi GPIO for BPM280 Barometer (I2C)
- Raspberry Pi 3B+ for various testing
 - Domoticz V4.1xxxx BETA to ensure include latest fixes
 - Node-RED

Software

- Raspbian GNU/Linux 9 (Stretch) based upon NOOBS Version 2.8.2, 2018-06-27
- Domoticz V4.10417 Stable
- 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 3.45.7.20190504 (full compatibility HomeMatic CCU3 3.45.7 firmware)
- WinSCP - to exchange files between the Development Device and the Raspberry Pi's
- PuTTY - to run terminal commands
- Idle - Python Script development
- The software versions mentioned have been used during the initial setup of the project. 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)

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 Home Automation server, 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://rpi-domoticz-ip:8080>) and for a Samba shared folder used f.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
...
...
```

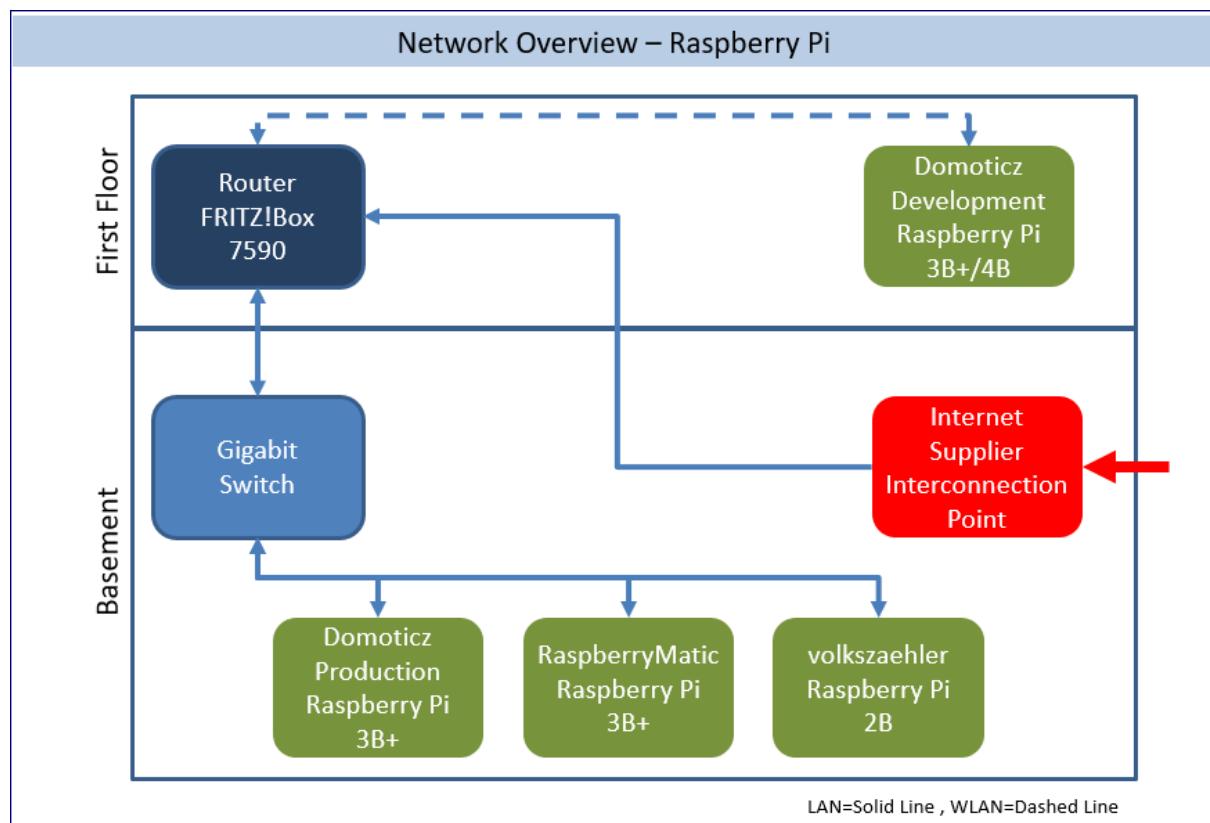
Notes

In this documentation the IP address of the Raspberry Pi Domoticz Server is referred to as:
<http://rpi-domoticz-ip:8080>

Wired Network

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

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



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.

```
cat /etc/os-release
PRETTY_NAME=" "
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
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"
```

Raspian Update to the latest version.

Perform regular updates (doing this 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
```

Samba

Start by reading [here](#).

Hints: 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 Server

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
Writeable = 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 /etc/init.d/samba restart
```

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 \\ rpi-domoticz-ip\DoProDomoticz
(DoProDomoticz is previous defined in the samba configuration)

To connect from the Windows explorer:

Network drives:

Add new drive with url \\rpi-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: \\rpi-domoticz-ip\MyMusic

Username: Pi

Password: *****

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:

Services:

http, Port: 8080.

Domoticz Folder:

/home/pi/domoticz

Domoticz URL:

http://rpi-domoticz-ip:8080

or

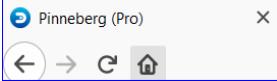
http://localhost:8080 (when using the Monitor connected to Raspberry Pi)

If everything went ok, enter the Domoticz URL in the Browser and ... the Domoticz UI is shown. The Domoticz version is displayed as V4.9700.

Now READY to configure the Domoticz Home Automation 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
Log History	Set to 1 days
Notifications	See under Functions where applicable
Email	
Email enabled	username <email-address>
Send Email notification alerts	Enabled
Meters / Counters	
Wind Meter	Display:Beaufort
Floorplan	See under Functions where applicable
Other	
Event System	Enabled
dzVents	Enabled <i>Note</i> dzVents is the main Event scripting used.

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

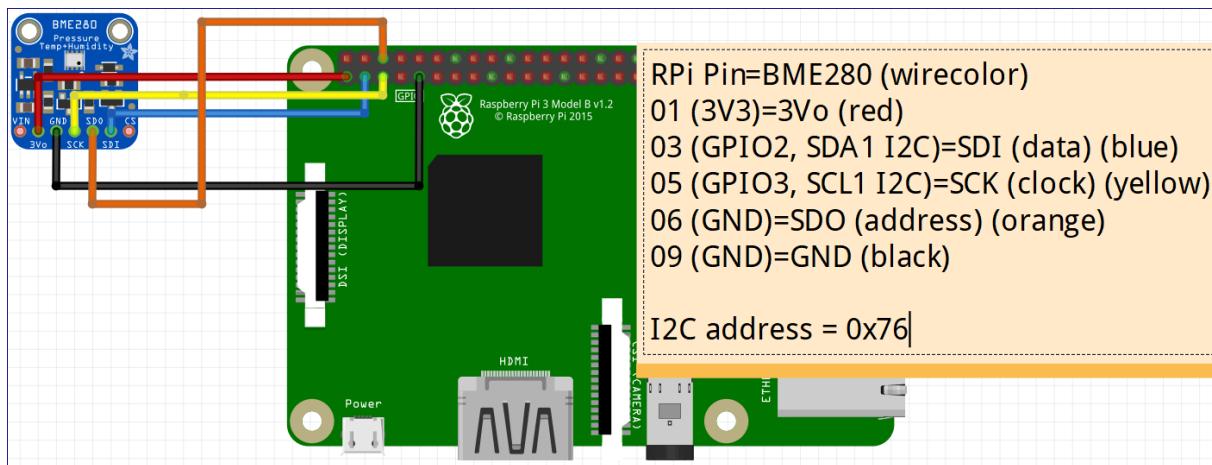
Air Pressure (BME280)

Purpose

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

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 command `i2cdetect -y 1` to check
- Domoticz uses the default address 0x76
- Adding multiple I2C sensors, requires a unique address for each sensor and two 1.k resistors between Vcc and the data & clock lines

Raspberry Pi B+ GPIO Header



I2C

Check if the I2C module (`i2c_bcm2835`) is loaded.

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

Below list is only an extract...

```
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                  I2C adapter
i2cdetect -y 1
  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 entry 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.

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

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).

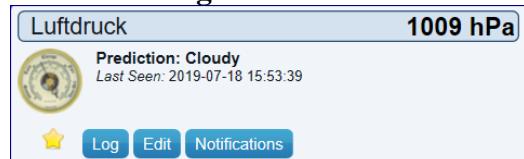
In the Dashboard would like to have a Barometer Widget with value taken from the BME280.

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 Lua script event to update the pressure and pressure forecast from the BME280 data. This is done using the Domoticz Event Editor.

Idx	Hardware	ID	Unit	Name	Type	Sub-Type	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: luftdruck_update

```
-- [[
luftdruck_update.lua
If the a value of the BME280 device (idx=115) has changed,
update the value of the Virtual Sensor Barometer named Luftdruck (idx=116)
Use print(device.dump()) ONLY ONCE to get the properties.
Project: atHome
Interpreter: dzVents, Device
See: athome.pdf
Author: Robert W.B. Linn
Version: 20180911
]]-- 

-- Idx of the devices
local IDX_BME280 = 115;
local IDX_LUFTDRUCK = 116;

-- Event handling changes of the BME280 device
return {
  on = {
    devices = {
      IDX_BME280
    }
  },
  execute = function(domoticz, device)
    -- print(device.dump())
    -- domoticz.log('Device '.. device.name .. ' changed ', domoticz.LOG_INFO)
    domoticz.log('Barometer: '.. device.barometer .. '/' .. device.forecast .. '/' ..
device.forecastString, domoticz.LOG_INFO)

    -- Round the pressure
  end
}
```

```

pressure = math.floor(device.barometer)

-- Forecast from the BME280 device to the Barometer device
-- See source code RFXNames.cpp, BMP_Forecast_Desc
-- Map the BMP_Forecast_Desc string to dzEvents Forecast value:
-- domoticz.BARO_STABLE, BARO_SUNNY, BARO_CLOUDY, BARO_UNSTABLE, BARO_THUNDERSTORM

forecast = -1
if (device.forecastString == 'Stable') then forecast = domoticz.BARO_STABLE end
if (device.forecastString == 'Sunny') then forecast = domoticz.BARO_SUNNY end
if (device.forecastString == 'Cloudy') then forecast = domoticz.BARO_CLOUDY end
if (device.forecastString == 'Unstable') then forecast = domoticz.BARO_UNSTABLE end
if (device.forecastString == 'Thunderstorm') then forecast = domoticz.BARO_THUNDERSTORM end
if (device.forecastString == 'Unknown') then forecast = domoticz.BARO_STABLE end
if (device.forecastString == 'Cloudy/Rain') then forecast = domoticz.BARO_CLOUDY end

-- Update the virtual sensor Luftdruck
domoticz.devices(IDX_LUFTDRUCK).updateBarometer(pressure, forecast)
end
}

```

Domoticz Log Entry

```

2019-07-18 15:57:39.571 Status: dzVents: Info: Handling events for: "BME280", value:
"22.5;53;1;1009.9;4"
2019-07-18 15:57:39.572 Status: dzVents: Info: ----- Start internal script: luftdruck_update: Device:
"BME280 (BMP280)", Index: 115
2019-07-18 15:57:39.572 Status: dzVents: Info: Barometer: B=1009.9000244141/F=4/FS=Cloudy
2019-07-18 15:57:39.573 Status: dzVents: Info: ----- Finished luftdruck_update

```

Alert Message

Purpose

To display an Alert Message on the GUI tabs Dashboard and Utility.
If the Alert Level is 4 (ALERTLEVEL_RED), then send 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 lamps living room timed switch (Script: hue_wz_timer.lua)
- More see the dedicated functions

Solution

Add a Virtual Device named “Alert Messages”, Type General, SubType Alert, Idx=55.

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
55	VirtualSensors	82054	1	Alert Messages	General	Alert	08.07.2019 14:41 - Postbox opened!

Add a User Variable IDX_ALERTMSG, Integer, Value=55 (see previous).

6	IDX_ALERTMSG	Integer	55
---	--------------	---------	----

Dashboard

Utility Sensors

Alert Messages

08.07.2019 14:41 - Postbox opened! ⚠

Last Seen: 2019-07-08 14:41:39

Type: General, Alert

Log Edit Notifications

Widget

Alert Messages

08.07.2019 14:41 - Postbox opened!

Last Seen: 2019-07-08 14:41:39

Type: General, Alert

Log Edit Notifications

Log Entries

Alert Messages	
Show 25 ▾ entries	
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 level (nValue) > 4 (or any other value set by uservar TH_ALERTTOEMAIL) then 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

```
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

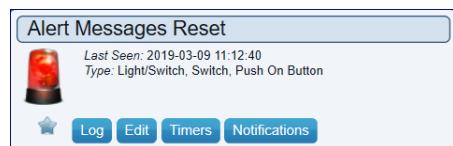
Create a virtual device type: Light/Switch, SubType: Switch.

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.

Device List

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
119	VirtualSensors	000140C7	1	Alert Messages Reset	Light/Switch	Switch	Off

Widget



User Variable

Idx	Variable name	Variable type	Current value
7	DEF_ALERTMSG	String	Keine

dzVents Script

```
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.db, table DevicesStatus.

SELECT * FROM DeviceStatus WHERE ID=55;																										
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	0	0	0	0	0	Opt	

The log entries are stored in table LightingLog.

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	

In case entries should be deleted, stop domoticz, delete the rows using an SQLite tool and start domoticz.

Ambient Light (BH1750)

Purpose

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

I2C

The BH1750 sensor communicates with the ESP8266 microcontroller via I2C.
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

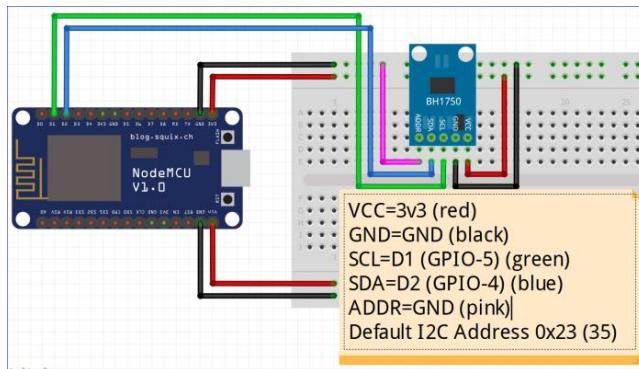
This device uses the default address 0x23 (ADDR connected to GND).

Parts Needed

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

Wiring

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



Device Setup Domoticz

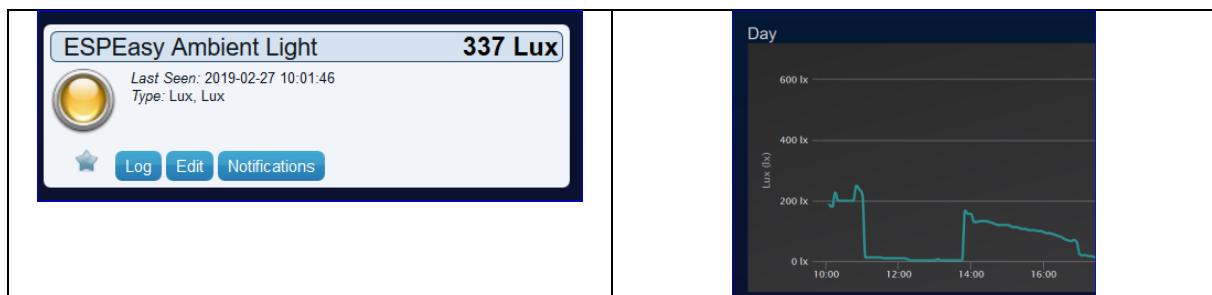
Create a virtual sensor: Sensor Type Lux with Name “ESPEasy Ambient Light”.
After adding the new device, the idx is required for the ESP device settings: idx=46.

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 web browser by opening:

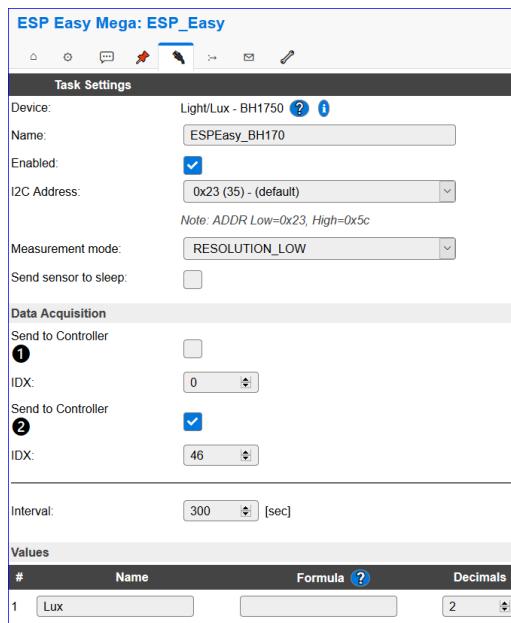
<http://esp-device-ip-address> > goto Devices > select Task Edit and configure.

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 Entry Sample

```
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 Entry

The log entry shows a Lux value which 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

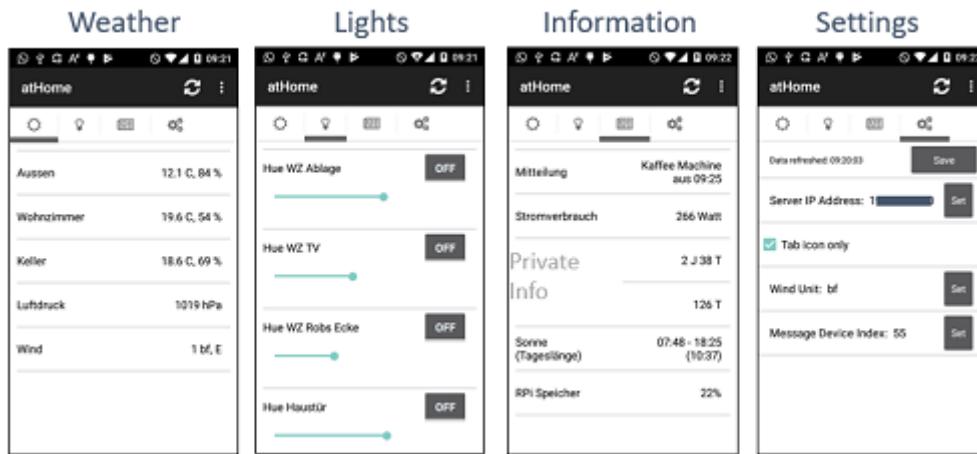
Purpose

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

Solutions

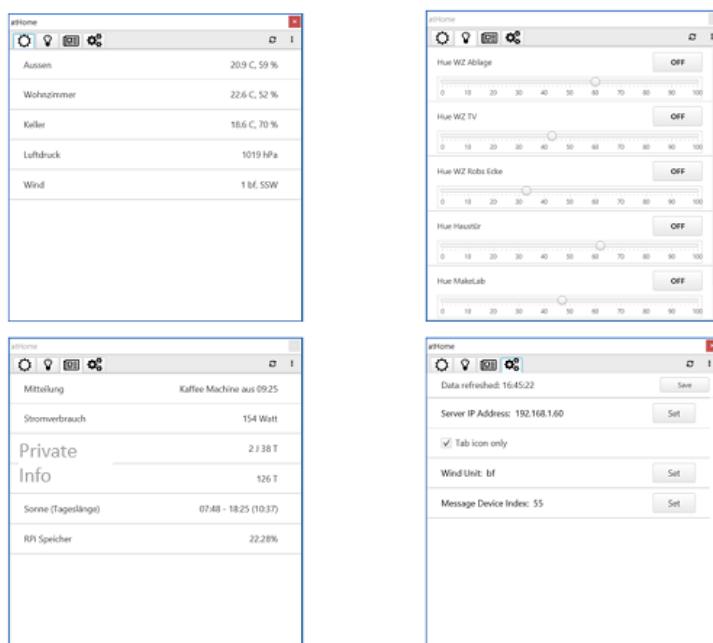
Android app developed with [B4A](#) v8.5.

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



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

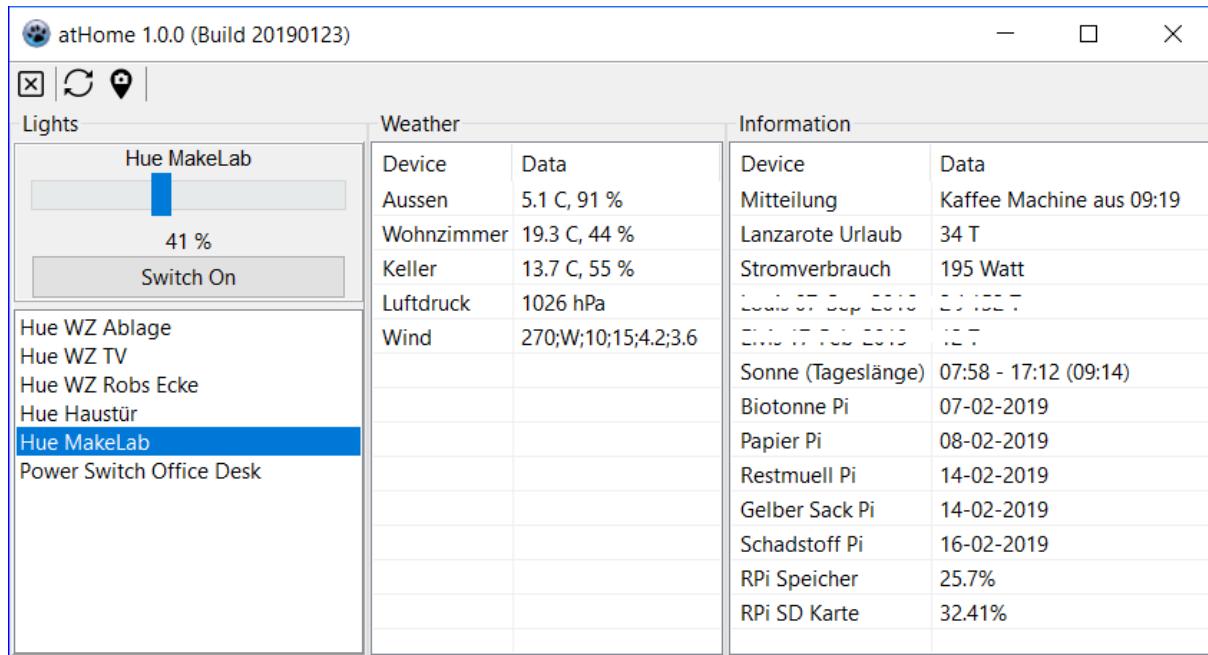
Read the B4J Forum *Share My Creations* post [here](#).



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)



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://rpi-domoticz-ip:8080)
2. Domoticz GUI Tab Switches > Select Manual Switches
3. Select Manual/Light Switch
4. Define the properties (see right)

Do NOT press Add Device

5. On the Remote RTS Pure Telis 1:
Press the Prog Button for 3 seconds.
The Blind will move short (=react).
6. Click Add Device
7. 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

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

The screenshot shows the Domoticz software interface divided into three main sections:

- Add Manual Light/Switch Device:** A configuration dialog box where "Hardware" is set to "RFXtrx433e", "Device name" is "Blind Bed Room", "Switch Type" is "Blinds", "Type" is "RFY", "ID" is "1E 1E 1E", "Unit Code" is "2", and "As:" is set to "Main Device". Buttons for "Test", "Add Device", and "Cancel" are visible at the bottom.
- Blind Bed Room:** A device card for the newly added blind. It shows the device name, last seen time (2015-06-19 15:06:34), type (RFY, RFY, Blinds), and status (Stopped). It includes icons for a star, Log, Edit, Timers, and Notifications.
- Devices List:** A table view of all configured devices. The table has columns for Idx, Hardware, ID, Unit, Name, Type, SubType, Data, and Last Seen. The row for the newly added blind (Idx 13) is highlighted with a red border. The table shows the following data:

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Last Seen
14	RFXCom	450E	2	Temp Outside	Temp + Humidity	Cresta, TFA TS34C	19.0 C, 55 %	2015-06-19 15:55:10
11	RFXCom	2A0E	1	Temp Living Room	Temp + Humidity	Cresta, TFA TS34C	17.6 C, 59 %	2015-06-19 15:54:06
6	GPIO Port	18	1	LED GPIO18	Lighting_1	Impuls	Off	2015-06-19 15:20:16
12	RFXCom	1F1F1F	1	Blind Living Room	RFY	RFY	Stopped	2015-06-19 15:18:55
13	RFXCom	1E1E1E	2	Blind Bed Room	RFY	RFY	Stopped	2015-06-19 15:06:34

Dashboard & Switches Widget

Dashboard



Domoticz GUI Tab Switches Widgets Blind Living & Bed Room



MQTT Messages

Example MQTT Messages for Topic="Domoticz/out", idx=13, name="Blind Bed Room".

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

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"> Left = Magnet Right = Detector with <ul style="list-style-type: none"> top left red LED (lights up if the contact is “broken”), Middle right reset button used for Domoticz to learn the device. 	<p>(Status 20190521) Tested but not supported by RFXCOM RFxTrx433E. The RFXCOM Manager recognized the switch as: Packettype = Lighting5 ERROR: Unknown Sub type for Packet type=14: 11 Signal level = 7</p>
<p>Ways-of-Working 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><TODO> 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: 160
- HW: RFXtrx433e
- Name: Unknown
- Type: Security
- SubType: X10 security
- Data: Alarm + Tamper
- Battery: full green

Add and name the device: Security Main Door.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Full	Battery
(Device icon)	160	RFXtrx433e	5C0700	0	Security Main Door	Security	X10 security	Alarm + Tamper	7	(Battery icon)

The device widget is listed in the tab Switches.

Alarm Detection

 <p>Security Main Door Alarm + Tamper</p> <p>Last Seen: 2019-05-20 09:43:46 Type: Security, X10 security, Security</p> <p>   </p>	<p>If door is opened, the alarm state is set to Alarm + Tamper. 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 Normal</p> <p>Last Seen: 2019-05-20 09:36:00 Type: Security, X10 security, Security</p> <p>   </p>	<p>State reset to Normal via HTTP API Request: <code>http://rpi-domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=160&switchcmd=Normal</code></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

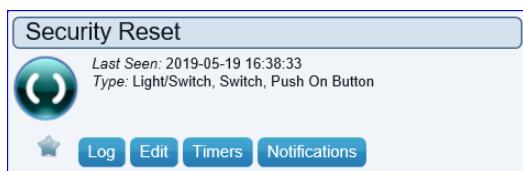
Idx	Variable name	Variable type	Current value
13	DEF_SECURITY_ALARM	Integer	0

Alarm State Reset Switch

Solution to reset all security devices (mainly X10) to state Normal by using a switch:

- Idx: 162
- Name: Security Reset
- Type: Light/Switch
- SubType: Switch
- Switch Type: Push On Button

which triggers a dzVents Lua script event “security_reset”.



	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
⑥	162	VirtualSensors	000140F2	1	Security Reset	Light/Switch	Switch	On

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://rpi-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://rpi-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.

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 Entry

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 now 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
```

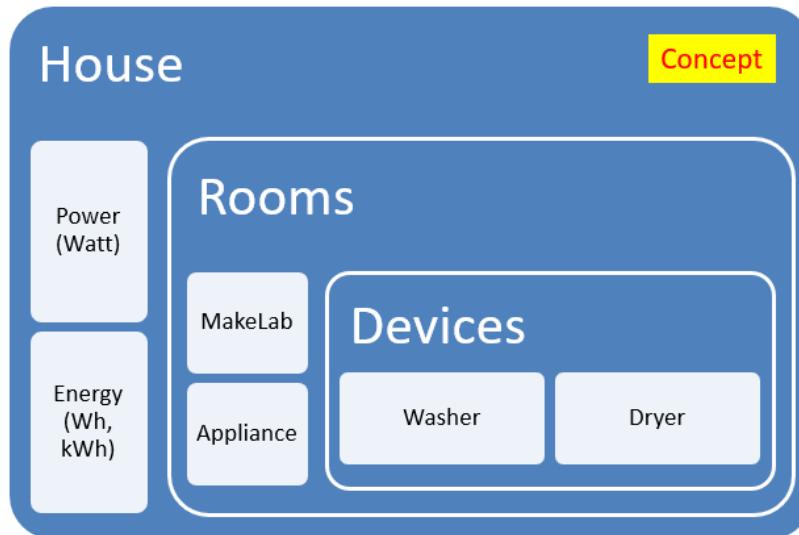
Electric Usage

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

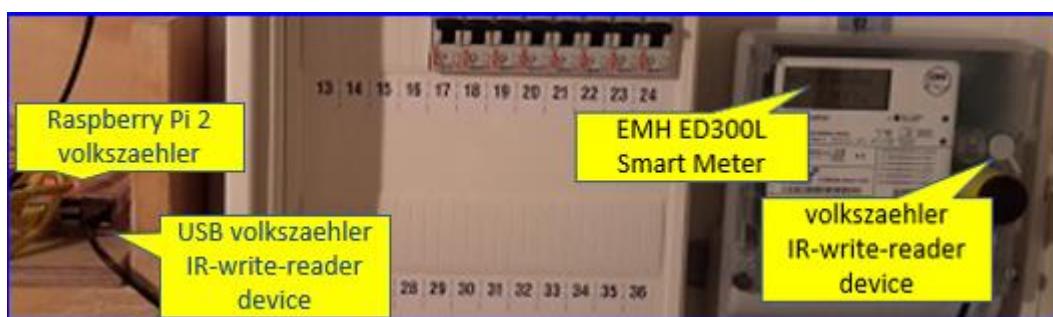
The house has a Smart Meter (Type EMH ED300L) which records consumption of electric energy used for monitoring and billing.

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 server 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

 <p>Electric Usage House 2317 Watt</p> <p>Today: 2.060 kWh Last Seen: 2019-07-29 09:35:00 Type: General, kWh</p> <p> Log Edit Notifications</p>	Edit Device <hr/> <p>Name: <input type="text" value="Electric Usage House"/></p> <p>Description: Energy (Wh) & Power (Watt) provided, every minute, by the volkszaehler Raspberry Pi. The Energy is computed from the Power.</p> <p>Type: <input type="button" value="Usage"/></p> <p>Energy read: <input checked="" type="radio"/> Computed <input type="radio"/> From device</p>
--	--

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://rpi-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-ADDRESS:8080'
local URL_VZREQUEST = 'http://VZ-IP-ADDRESS/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 Entry

```
#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: "electricusagelhouse"
2019-07-29 10:03:00.396 Status: dzVents: Info: ----- Start internal script: electric_usage_house: HTTPResponse: "electricusagelhouse"
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://rpi-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

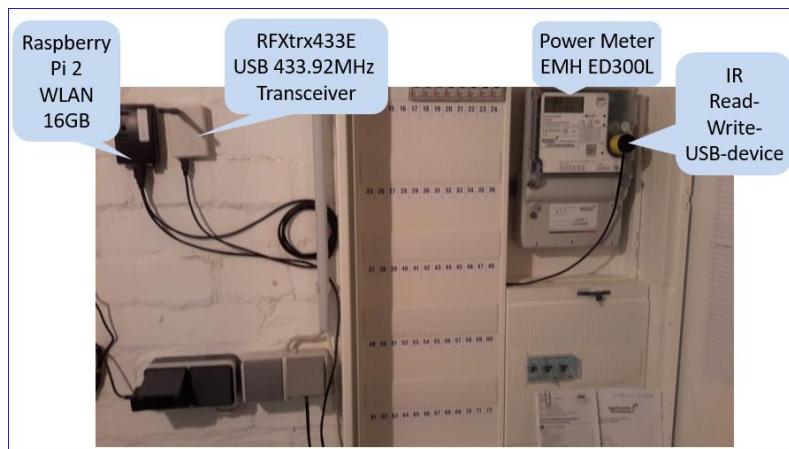
Example SQL SELECT on table Meter

```
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 Server and the vz Server via WLAN. The vz Server has a fixed IP address NNN.NNN.1.NNN, which is defined in files:

/etc/dhcpcd.conf

```
# wlan interface
interface wlan0
static ip_address=rpi-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 **rpi-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.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://rpi-vz-ip`) is crucial.

Must read: 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.

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 Kanal erstellen</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th><th>Wert</th></tr> </thead> <tbody> <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> </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>Erstellen Cookie: <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	 	Stil	steps	Füllgrad		Achse	auto	Kosten		Initialverbrauch		<p>Kanal hinzufügen</p> <p>Öffentliche Kanäle Private Kanäle Kanal erstellen</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th><th>Wert</th></tr> </thead> <tbody> <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> </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>Erstellen Cookie: <input checked="" type="checkbox"/></p>	Eigenschaft	Wert	Middleware:	Local (default)	Typ:	El. Energie (Leistungswerte)	Titel	Energie Leistung	Öffentlich	<input checked="" type="checkbox"/>	Farbe	 	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	 																																																
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	 																																																
Stil	lines																																																
Füllgrad																																																	
Achse	auto																																																
Auflösung																																																	
Kosten																																																	
Initialverbrauch																																																	
UUID 5b8ea2a0-b342-11e8-bec6-15c040e6d041	UUID 958bce60-b342-11e8-b54f-bbec0573e1f4																																																
// /etc/vzlogger.conf <pre>{ // Zählerstand "uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "middleware" : "http://localhost/middleware.php", "identifier" : "1-0:1.8.0" }</pre>	/etc/vzlogger.conf <pre>{ // 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>

Output

```
{"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, 182.37082168777], "max": [1559311320072, 209.18070811092], "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://rpi-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.
 Browser URL = <http://rpi-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	Browse Structure Search Insert Empty Drop	~2,612,489	InnoDB	latin1_swedish_ci	190.3 Mib	-
data	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 Kib	-
entities	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8_unicode_ci	32 Kib	-
entities_in_aggregator	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 Kib	-
properties	Browse Structure Search Insert Empty Drop	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://rpi-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 commands.

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://rpi-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://rpi-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://rpi-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://rpi-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 scripts 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 befor 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
pi@169.254.87.85's password: *****
- The pi@DoPro prompt is shown

Electric Usage Rooms/Devices

Purpose

To measure the Power (Watt) and Energy Usage (kWh) for selected devices (washer, dryer etc.) or dedicated rooms (MakeLab, Library).

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

homematicIP HMIP-PSM (Plugin)

The Domoticz Python Plugin is used:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
17	MakeLab Energy	Yes	homematicIP Pluggable Switch and Meter (HMIP-PSM)	192.168.1.225		Disabled

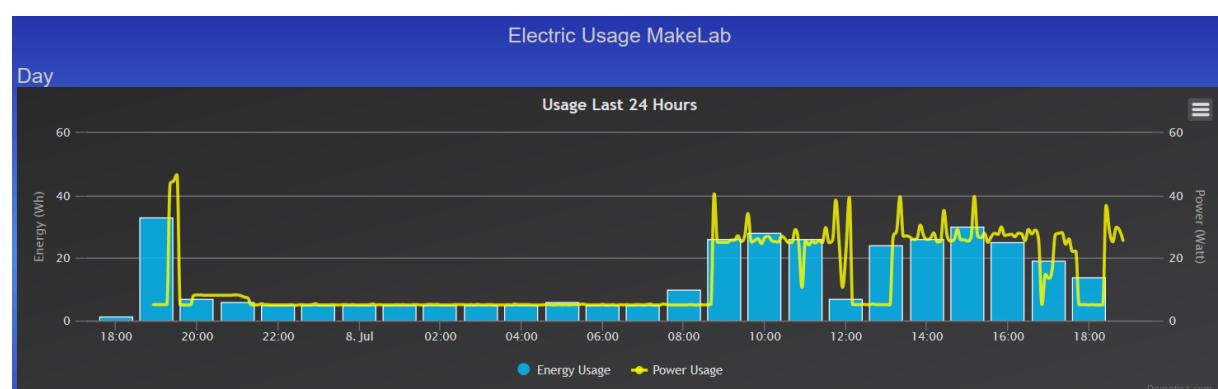
This plugin is described and can be downloaded [here](#).

Devices (created by the plugin)

Idx	Hardware	ID	Unit	Name	Type	SubType	Data			
179	MakeLab Energy	00110001	1	Electric Usage MakeLab	General	kWh	1.613 kWh	-	-	↻
180	MakeLab Energy	00110002	2	MakeLab - Voltage	General	Voltage	230.1 V	-	-	⟳
181	MakeLab Energy	00110003	3	MakeLab - Current	General	Current	0.21 A	-	-	⟳
182	MakeLab Energy	00110004	4	Powerswitch MakeLab	Light/Switch	Switch	On	-	-	↻

Note

The device idx=179, Electric Usage MakeLab is relevant.



The dashboard for room Energy shows the devices which belong to the Domoticz Plugins

- homematicIP Radiator Thermostat (HMIP-eTRV-2)
- Pluggable Switch and Meter (HMIP-PSM)

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** (~14EUR).

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:

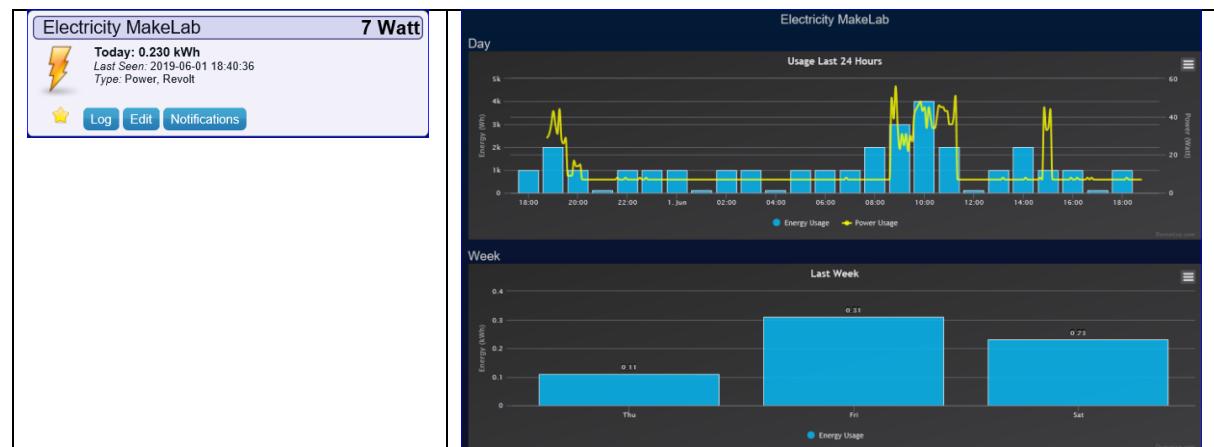
1. Type=Power, SubType=Revolt, Data=0.070kWh, Idx=167
2. Type=General, SubType=Voltage, Data=228 V, Idx=168
3. Type=General, SubType=Percentage, Data=0.73%, Idx=169 Note: power factor.
4. Type=General, SubType=Percentage, Data=50%, Idx=170 Note: frequency Hz.

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://rpi-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 then 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 server from the public internet.

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

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

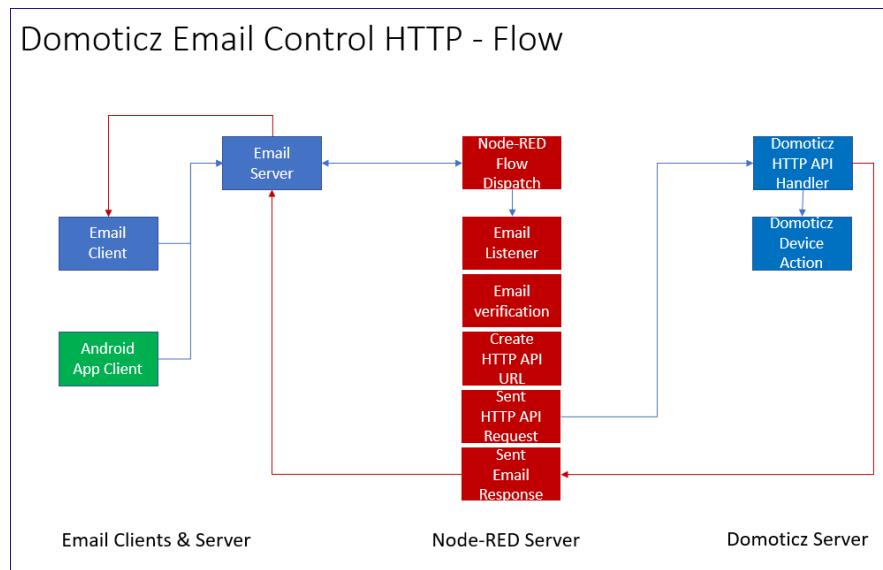
The solution uses a Node-RED flow (named Email Control), acting as kind of middleware, to handle Domoticz commands being sent via email (i.e. email client or App).

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 futher.



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-ADDRESS: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
```

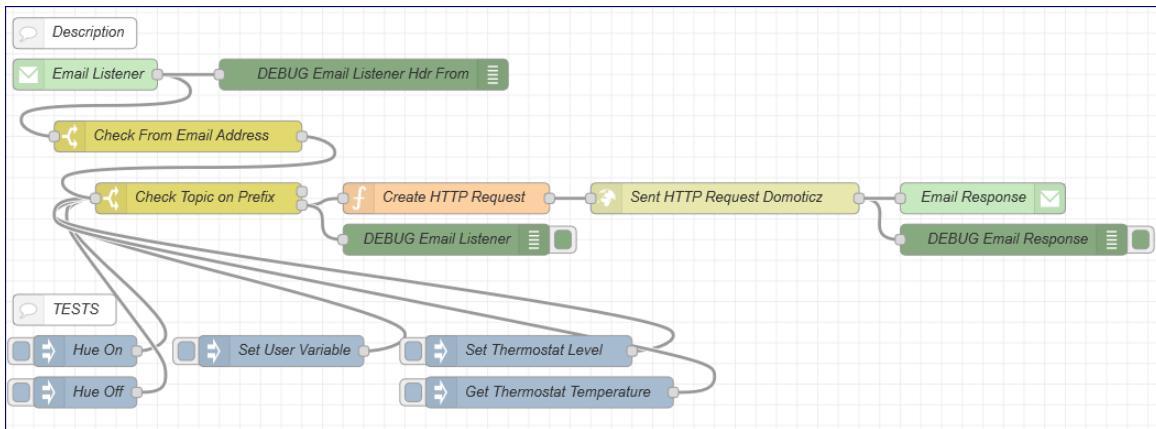
```
DOMOTICZ:Set User Variable AlertToEmail to 1
type=command&param=updateuservariable&vname=TH_ALERTTOEMAIL&vtype=0&vvalue=1
```

```
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 Server using 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 server. The message header is JSON parsed.

Example:

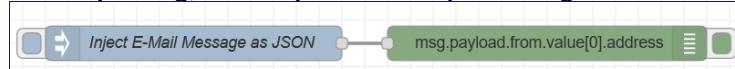
```
msg.header.from.value[0].address
```

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 to check 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-address:8080/json.htm?" + msg.payload;  
// Return the message to be published by the mqtt out node  
return msg;
```

Note:

if the Node-RED is running on the same server 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:

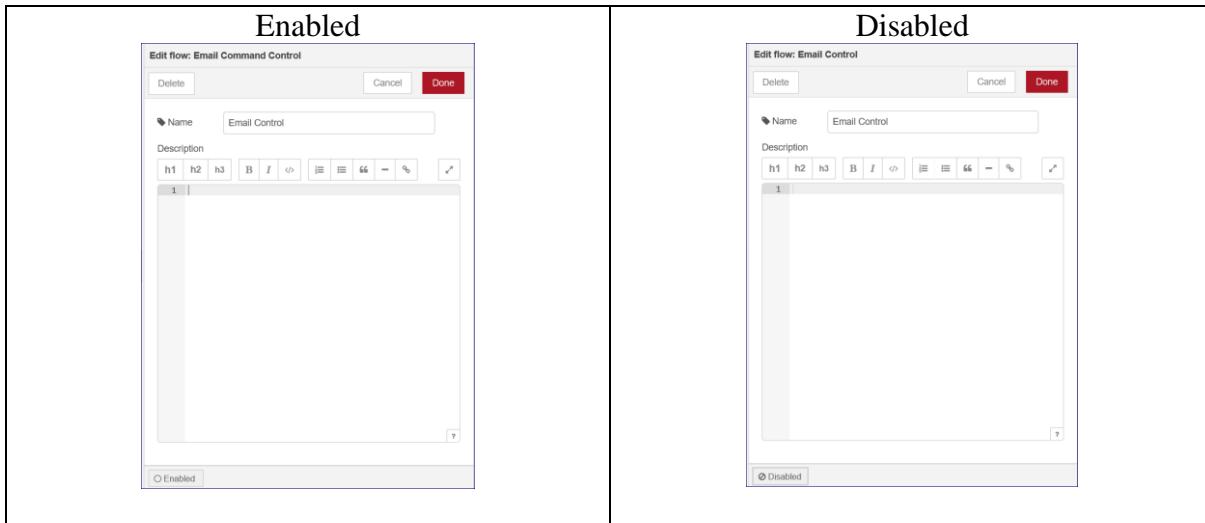
```
{"status" : "OK", "title" : "SwitchLight"}
```

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¶m=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: (wem)
"DOMOTICZ:Thermostat MakeLab Off"
07/11/2019, 09:59:01 node: Create HTTP Request
function: (wem)
"RESPONSE:Thermostat MakeLab Off"
07/11/2019, 09:59:01 node: Create HTTP Request
function: (wem)
"http://<ip-address>/0/json.htm?type=command&param=switchlight&idx=183&switchcmd=Set%20Level&level=0"
07/11/2019, 09:59:01 node: DEBUG 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: DEBUG Email Response
RESPONSE:Thermostat MakeLab Off: msg.payload: string[5]
>"{"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¶m=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:// ip-address 8/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[240]
{"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"
}

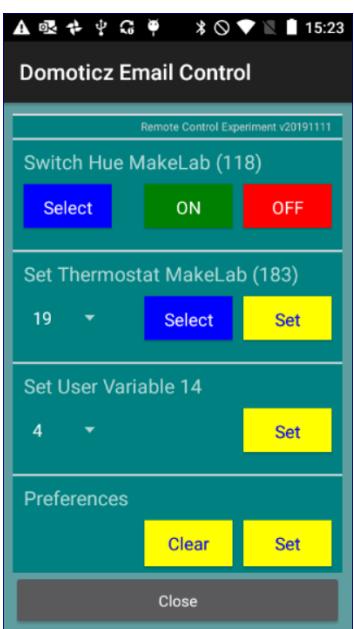
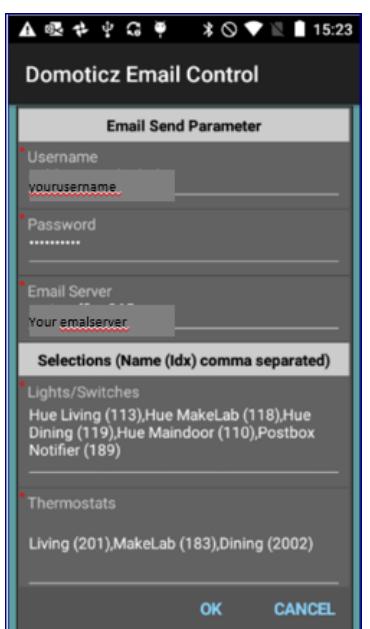
Android Client App

An example of a simple app to test sending emails with commands (subject & body) to the Domoticz server 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"> 1. Switch Light / Switch Select a device and switch ON or OFF Example Hue MakeLab 2. Set Thermostat Select a device and set the temperature from dropdown list between 0 (Off), 18, 19, 20 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) 	<p>The preferences dialog uses the B4XPreferences library with a json configuration file (Files Folder > 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

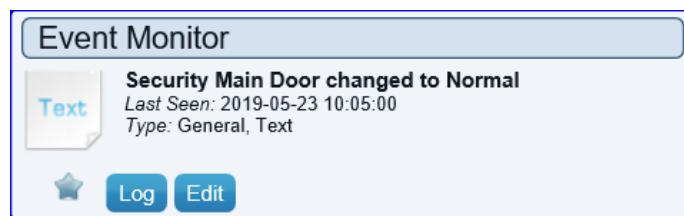
Name: Event Monitor

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 section Helpers)
IDX_EVENTMONITORMSG = 161,

(in section 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
```

Indoor Air Quality (Plugin, Tinkerforge)

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: 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):

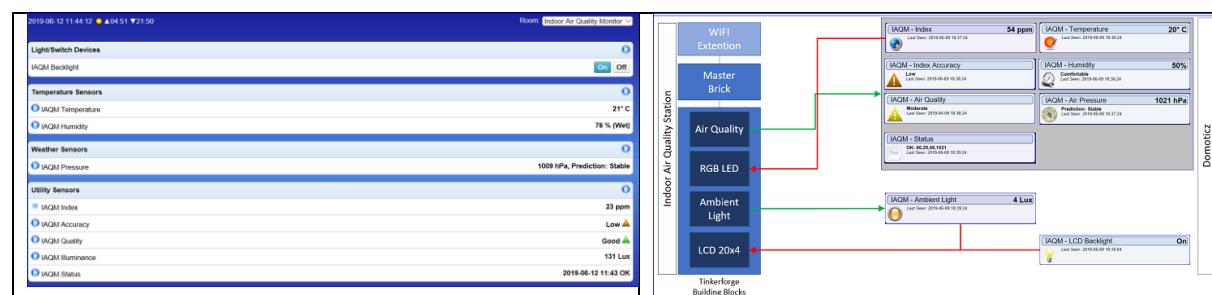
- Index (General, Custom Sensor), Index Accuracy (General, Alert), Air Quality (General, Alert [Text=Level]).
- Temperature (Temp, LaCrosse TX3), Humidity (Humidity, LaCrosse TX3), Air Pressure (General, Barometer), Ambient Light (Lux, Lux).
- LCD Backlight (Light/Switch, Switch), Status (General, Text).

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	Last Seen: 2018-09-05 17:18:05	
Type: General, Text		
	Log	Edit

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
}
```

Hardware Monitor Raspberry Pi

Purpose

To monitor hardware data for the Raspberry Pi (RPi) devices used in this Home Automation 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 Server.

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 Device:

- 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

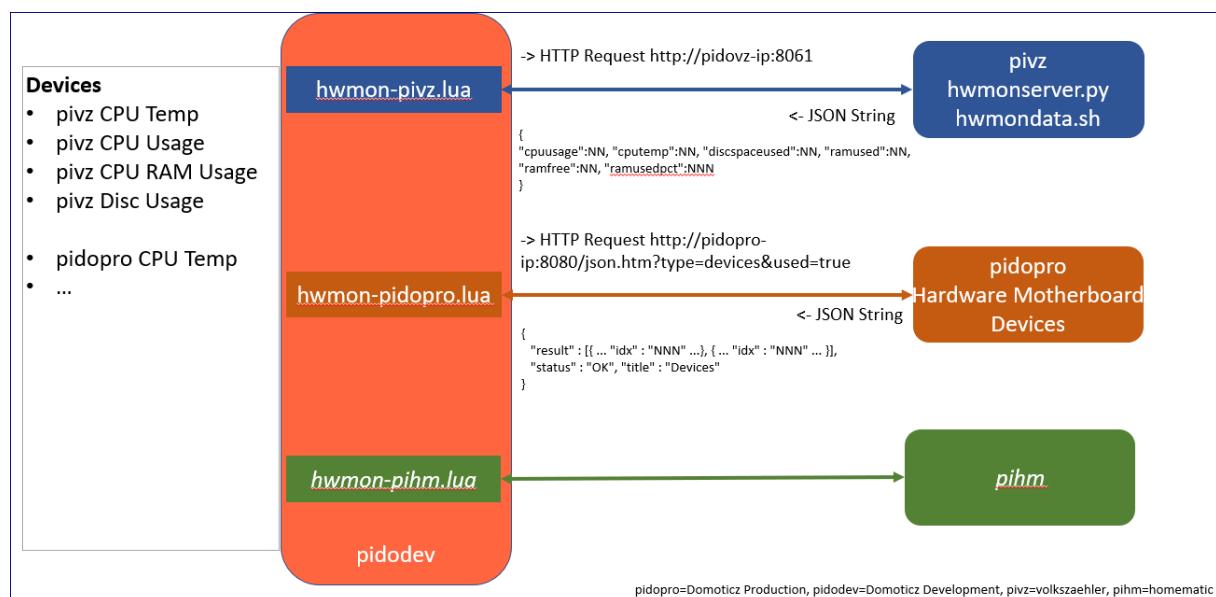
Tested various Node-RED flows for pivz data.

- 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



Solution 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 located 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 Example

```
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
```

Solution RPi Domoticz Production

Device name: pidopro

A dzvents Lua script event is running on the Domoticz Development server, gathers all used devices data from the Domoticz Production server.,

The hardware monitor devices (#4) are selected ad used to update the devices on the Domoticz Development server.

Automation Script

Event name: hwmon-pidopro (embedded in the domoticz database).

Pseudo Code

Place a HTTP request to the Domoticz Production (pidopro) server 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 server

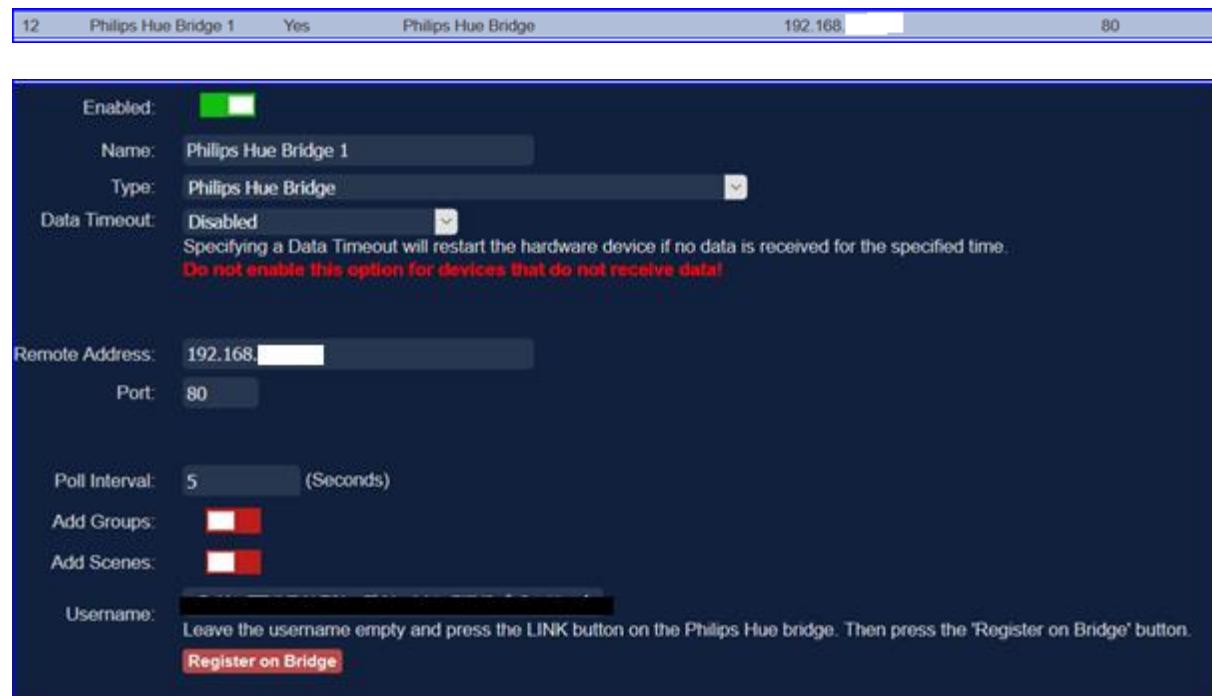
Philips Hue

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 (Android).

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).

Note

This might take few minutes.

Examples

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	00000007	1	Hue Bloom Schrankwand	Color Switch	RGBW	Set Color

Domoticz Log Entry – 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)

Add the device with name Hue MakeLab, set the level to 100%.



118	Philips Hue Bridge 1	0000005	1	Hue MakeLab	Lighting 2	AC	Set Level: 100 %
-----	----------------------	---------	---	-------------	------------	----	------------------

Domoticz Log Entry – Hue Lamp Level Change

If the level of the Hue Lamp is changed, the Domoticz Log Entry:

2018-09-05 13:37:55.133 Status: User: Admin initiated a switch command (118/Hue MakeLab/Set Level)
--

Hue Control Tests

Lets start 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 1 – Device Trigger

If the Hue Lamp is switched ON, then switch it 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 2 – 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.convertimehhmm(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 Entry

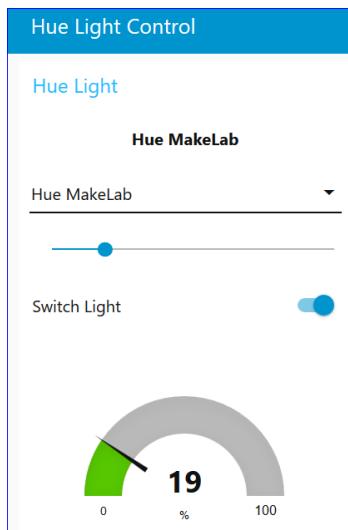
```
2018-09-05 20:02:00.394 (Philips Hue Bridge 1) Color Switch (Hue WZ Ablage)
2018-09-05 20:02:00.417 (Philips Hue Bridge 1) Lighting 2 (Hue WZ TV)
2018-09-05 20:02:00.259 Status: dzVents: Info: ----- Start internal script: hue_wz_timer:, trigger: at sunset
2018-09-05 20:02:00.276 Status: dzVents: 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 webbrowser.
Select a Hue and switch the light ON|Off or set Brightness Level.

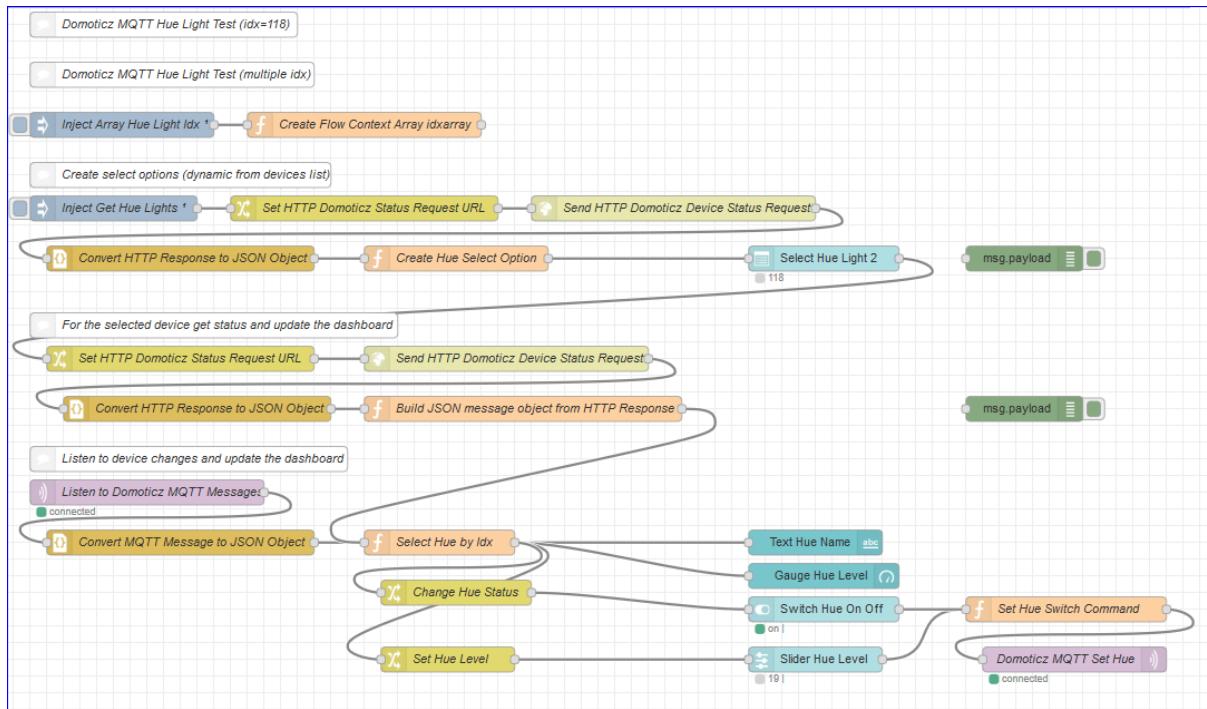
Read more about Node-RED [Appendix Node-RED](#).

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 web 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 send 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 web 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.

Other Information

Learn more in depth about Hue at [meethue](#).

Postbox Notifier (RaspberryMatic)

Purpose

To watch the postbox for post and notify via Email Notification and Alert Message.

Solutions

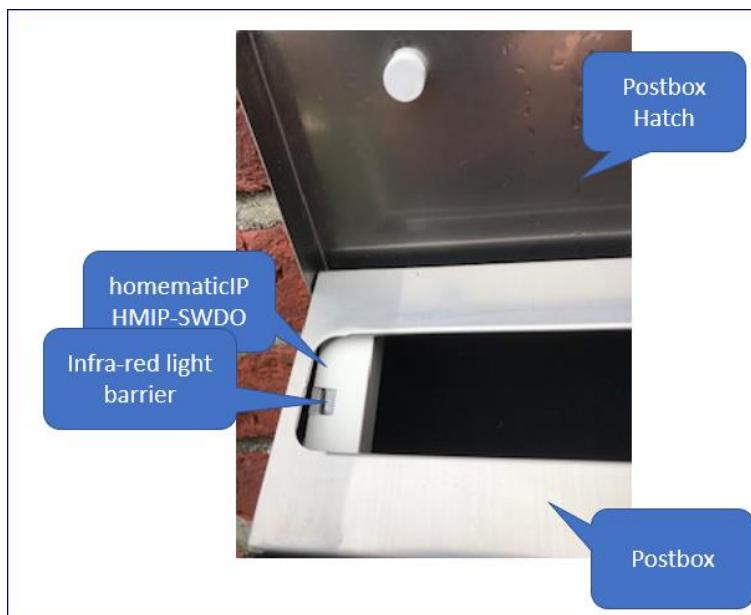
A detector (with an integrated infra-red light barrier) detects opening the hatch of the postbox and triggers sending an Email Notification displayed in the [Alert Message](#).

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 section 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).



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 This solution is in place.	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 server.

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 Domoticz server 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 one row for the HMIP-SWDO device. The device details are: Type description: HMIP-SW DO; Picture: Homematic IP Window / Door Contact - optical; Description: Homematic IP Window / Door Contact - optical; Serial number: 0000DA498D5859; Interface category: HmIP-RF; Transmission mode: Secured; Name: HMIP-SWDO 0000DA498D5859; Function: ; Room: ; Functional test: Test (button), Delete (button), OK (button), Set (button); Action: operable, visible, logged; Done: Done button. There is a blue oval around the "Add new device" link.

(2) Device Status and control: A screenshot of the "Status and control > Devices" page. It shows a table with five rows of devices. The fifth row is highlighted in blue and corresponds to the HMIP-SWDO device from the first screenshot. The columns are: Channel, Room, Function, Last modified, Control. The last modified time is 08.07.2019 09:46:03. The control section shows two buttons: "Open" (blue) and "Closed".

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
SWDO5859	Postbox Alert	Channel status: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated	Script: ... immediately run	<input type="checkbox"/> intrinsic
Condition: If... Device selection: HMIP-SWDO 0000DA498D5859:1 when open trigger when updated OR DR				
Activity: Then... <input checked="" type="checkbox"/> Stop all current delays before performing the activity (e.g. retriggering). Script: Function: Postbox Alert string sFunction = "Postbox Alert"..., immediately				
Activity: Else... <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-
address: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-address:8080/json.htm"; ! Production
string sUrl =
sDomUrl#"?type=command#cAmp#param=udevice#cAmp#idx="#nIdx#cAmp#"nvalue="#nLevel#cAmp#"svalue="#sText#cAmp#";
;

! 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_RETS").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

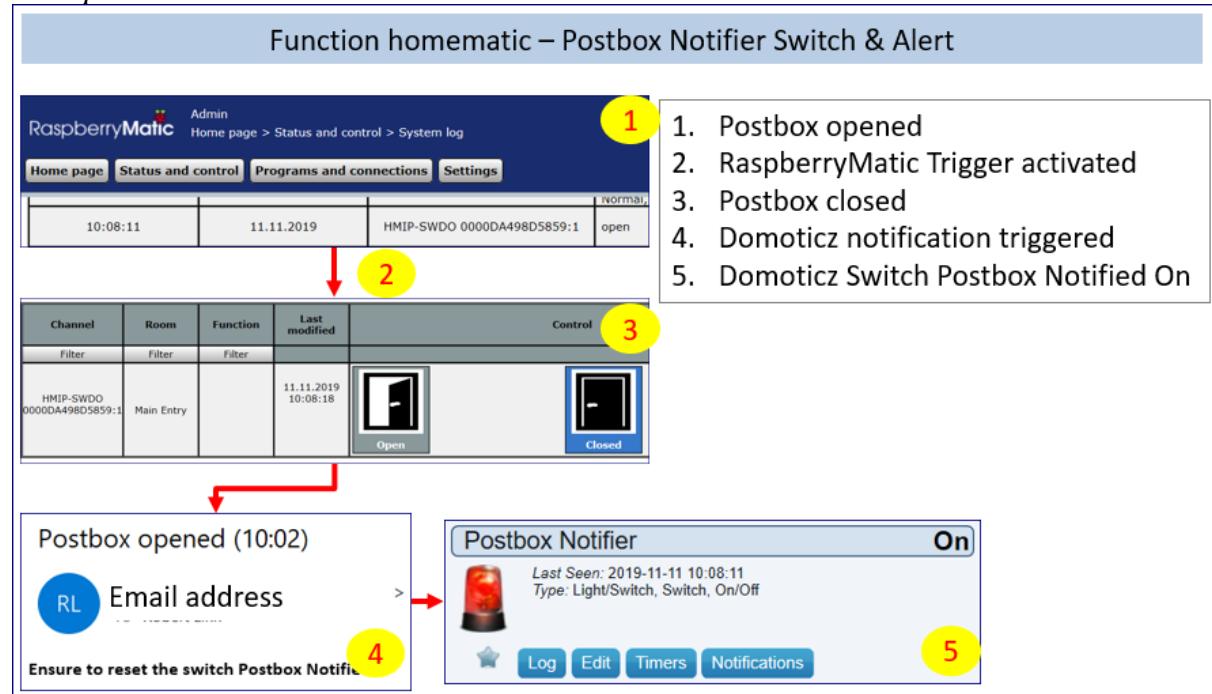
To avoid sending bulk emails within short timeframe, the notified time is checked against the previous notified time.

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 server 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).

(1) Add new device

(2) Device Status and control

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
```

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-address: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-address: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
}
```

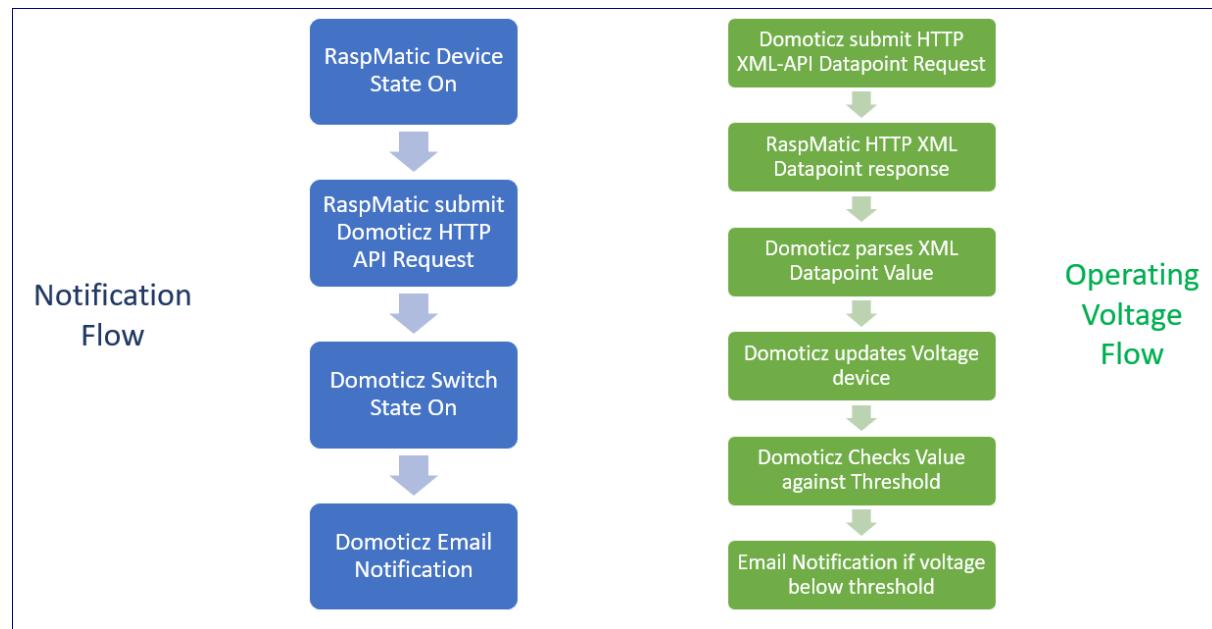
Solution Alert + Switch + Voltage

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

Log Edit Timers Notifications

Postbox Notifier State On



Last Seen: 2019-12-11 18:52:49
Type: Light/Switch, Switch, On/Off

Log Edit Timers Notifications

Email Notification

Robert Linn
Postbox opened (11:38)
Wed 11/12/2019 11:38
47 KB

Postbox Notifier: Ensure to reset switch. <end>

Domoticz Log Entry Switch On

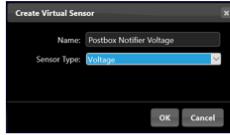
```
2019-12-11 11:38:25.383 Status: User: Admin initiated a switch command (189/Postbox Notifier State/On)
2019-12-11 11:38:25.578 Status: dZvents: Info: Handling events for: "Postbox Notifier State", value: "On"
2019-12-11 11:38:25.578 Status: dZvents: Info: ----- Start internal script: postbox_notifier: 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.



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 of variables, where a new row 'TH_POSTBOXNOTIFIER_VOLTAGE' has been added with a value of 1.0. The bottom part shows the 'Devices' section, specifically the 'Set device / channel parameter' page for a device named 'Postbox Notifier'. In the 'Channel parameter' table, there is a row for 'Ch.: 0' with a 'Low. sat. threshold' set to 1.00 V. Another row for 'Ch.: 1' shows 'Message in position open' and 'Message in position closed'.

Domoticz Log Entries

2019-12-11 18:24:00.564 Status: d2Vents: Info: ----- Start internal script: postbox_notifier:, trigger: every minute
 2019-12-11 18:24:00.571 Status: d2Vents: Info: ----- Finished postbox_notifier

2019-12-11 18:24:00.942 Status: d2Vents: Info: Handling httpResponse-events for: "RESPOSTBOXNOTIFIER"
 2019-12-11 18:24:00.942 Status: d2Vents: Info: ----- Start internal script: postbox_notifier: HTTPResponse: "RESPOSTBOXNOTIFIER"
 2019-12-11 18:24:00.942 Status: d2Vents: Info: Postbox Notifier Voltage:Voltage=1.5
 2019-12-11 18:24:00.959 Status: d2Vents: 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: d2Vents: Info: Handling events for: "Postbox Notifier State", value: "On"
 2019-12-11 18:26:52.032 Status: d2Vents: Info: ----- Start internal script: postbox_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189
 2019-12-11 18:26:52.033 Status: d2Vents: Info: Device Postbox Notifier State changed On = 0
 2019-12-11 18:26:52.034 Status: d2Vents: Info: POSTBOXNOTIFIER Postbox Notifier notification sent: Postbox opened (18:26) (Postbox Notifier: Ensure to reset switch.)
 2019-12-11 18:26:52.034 Status: d2Vents: 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: d2Vents: Info: Handling events for: "Postbox Notifier State", value: "Off"
 2019-12-11 18:28:31.009 Status: d2Vents: Info: ----- Start internal script: postbox_notifier: Device: "Postbox Notifier State (VirtualSensors)", Index: 189
 2019-12-11 18:28:31.010 Status: d2Vents: Info: Device Postbox Notifier State changed Off = 1
 2019-12-11 18:28:31.011 Status: d2Vents: Info: ----- Finished postbox_notifier

Domoticz Roomplan & Dashboard

The screenshot shows the 'Roomplan' and 'Dashboard' sections of the Domoticz interface. The 'Roomplan' section displays a list of devices, including 'Postbox Notifier' and 'Postbox Notifier Voltage'. The 'Dashboard' section shows a summary of the current status of the 'Postbox Notifier State' (On) and the 'Postbox Notifier Voltage' (1.5 V).

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 IDX_TH_POSTBOXNOTIFIER_VOLTAGE = 16
-- homematic device datapoint id (device HMIP-SWDO)
local URL_RASPMATIC = 'http://192.168.1.225/config/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":"RESPOSTBOXNOTIFIER"}');

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 sent an email if user var TH_ALERTTOEMAIL = 4
                domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, message)
            end

            -- Reset the notify flag if 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..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);
                -- Select the callback
                if (item.callback == DATAPOINT2543.response) then
                    -- Get the operating voltage
                    local voltage = tonumber(domoticz_applyXPath(item.data, DATAPOINT2543.xpath))
                    domoticz.log(DATAPOINT2543.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= DATAPOINT2543.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 (Plugin)

For testing this solution, the temperature for the room MakeLab is controlled.

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.

Solution

The homematicIP Radiator Thermostats HmIP-eTRV-B & HmIP-eTRV-2 are used.

The homematicIP devices are connection partner radiator thermostat (manual operation, transmitter) and connected to the [RaspberryMatic](#) system - [HomeMatic](#) compatible operating system.

These are used for several radiators in the house.

Timers are put in place to control the temperature depending the radiator location.

Each device has a battery which is regulary checked and alert given if below threshold 2.2V (defined in the HomeMatic WebUI for each device).

A Domoticz Python Plugin has been created, called **homematicIP Radiator Thermostat (HmIP-eTRV)**.

Installation

These steps apply for every Radiator Thermostat Device:

1. Add the device to RaspberryMatic
2. Download the plugin repository from [here](#)
3. On the Domotocz server, create folder /home/pi/domoticz/plugins/hmip-etrv
4. Copy the file plugin.py to /home/pi/domoticz/plugins/hmip-etrv
5. Restart Domoticz: sudo service domoticz.sh restart
6. For the hardware parameters are required:
 - a. CCU IP Address
 - b. Device ID of the Device HmIP-eTRV-B or HmIP-eTRV-2
 - c. Datapoint IDs: SET_POINT_TEMPERATURE, ACTUAL_TEMPERATURE, LOW_BAT
as comma separated list in this order
 - d. Obtain the ID's by requesting the device state list from the CCU (<http://ccu-ip-address/config/xmlapi/statelist.cgi>)
7. Goto GUI > Setup > Hardware
8. Select Type: homematicIP Radiator Thermostat (HMIP-eTRV)
9. Give a name: MakeLab Thermostat
10. Set the parameters; Keep debug to true
11. Ensure Domoticz Accepts New hardware (GUI > Setup > Settings > Hardware)
12. Add

13. Check the Domoticz Log if the devices MakeLab Thermostat - Setpoint, MakeLab Thermostat - Temperature & MakeLab Thermostat - Battery are created and updated
14. Check the widgets
15. If all OK, set parameter Debug to false
16. Optional additional steps (see also screenshots below):
 - a. Add the devices to a room plan, i.e MakeLab
 - b. Add a device timer to ensure the thermostat is turned Off to avoid heating overnight
 - c. Add a switch device to turn the thermostat off

Add new Hardware

Idx **Name** **Enabled** **Type** **Address** **Port** **Data Timeout**

Idx	Name	Enabled	Type	Address	Port	Data Timeout
10	MakeLab Thermostat	Yes	homematicIP Radiator Thermostat (HmIP-eTRV)	192.168.1.225		Disabled

Showing 1 to 1 of 1 entries (filtered from 6 total entries) [First](#) [Previous](#) [1](#) [Next](#) [Last](#)

[Update](#) [Delete](#)

Enabled:

Name: MakeLab Thermostat

Type: homematicIP Radiator Thermostat (HmIP-eTRV)

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 Radiator Thermostat (HmIP-eTRV) v1.0

- Set the setpoint (degrees C).
- Get the actual temperature (degrees C).
- Get the low battery state (true or false). Threshold set in the HomeMatic WebUI.
- Supported are the devices HmIP-eTRV-B, HmIP-eTRV-2

Domoticz Devices (Type,SubType)

- SelPoint (Thermostat,Setpoint)
- Temperature (Temp,LaCrosse TX3)
- Battery (General,Alert)

Hardware Configuration

- Address (CCU IP address, default: 192.168.1.225)
- IDs (obtained via XML-API script <http://ccu-ip-address/config/xmlapi/statelist.cgi>):
 - Device ID HmIP-eTRV-B or HmIP-eTRV-2 (default: 1541)
 - Datapoint IDs(#3): SET_POINT_TEMPERATURE, ACTUAL_TEMPERATURE, LOW_BAT as comma separated list in this order (defaults: 1584,1567,1549)
- Note: After configuration update, the setpoint is 0. Click the setpoint to set the value.

CCU IP: nnn.nnn.n.nnn

Device ID: 1541

Datapoint IDs: 1584,1567,1549

Check Interval (sec): 60

Debug: True

Devices Overview

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
71	MakeLab Thermostat	000A0003	3	MakeLab Thermostat - Battery	General	Alert	Battery status low
70	MakeLab Thermostat	000A0002	2	MakeLab Thermostat - Temperature	Temp	LaCrosse TX3	20.7 C
69	MakeLab Thermostat	000A0001	1	MakeLab Thermostat - Setpoint	Thermostat	SetPoint	0.0

MakeLab Thermostat - Temperature 20.5° C
Last Seen: 2019-12-14 13:19:19
 [Log](#) [Edit](#) [Notifications](#)

MakeLab Thermostat - Setpoint 0.0° C
Last Seen: 2019-12-14 12:40:09
 [Log](#) [Edit](#) [Timers](#) [Notifications](#)

MakeLab Thermostat - Battery
Battery status low
Last Seen: 2019-12-14 13:20:19
 [Log](#) [Edit](#) [Notifications](#)

Edit Device
Idx: 69
Name: MakeLab Thermostat - Setpoint
Description:
Protected:
Value: 22 °C
[Update](#) [Remove Device](#) [Cancel](#)

Temperature Sensors
MakeLab Thermostat - Temperature 20.5° C

Utility Sensors
MakeLab Thermostat - Setpoint 22.0 °C
MakeLab Thermostat - Battery

Actual 22.0 **22.0** **Set**
Battery status low

Roomplan & Dashboard for selected room

The roomplan MakeLab has 3 devices as created by the hardware plugin.

The thermostat setpoint has a timer which sets the setpoint to 0 at 1900.

Enhancement 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-address/config/xmlapi/statechange.cgi?ise_id=1584&new_value=0

Domoticz Log Example

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)
```

Timers

Each radiator thermostat has one or more timers to control the temperature.
The timers are defined in the Domoticz Database table SetpointTimers.

Example of an 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
```

Name	ID	Active	DeviceRowID	Date	Time	Type	Temperature	TimerPlan	Days	Month	MDay	Occurence
Htz Bad Sollwert	1	1	207	2019-12-15	06:00	2	20.0	0	128	0	0	0
Htz Bad Sollwert	2	1	207	2019-12-15	11:00	2	19.0	0	128	0	0	0
Htz Bad Sollwert	3	1	207	2019-12-14	23:00	2	17.0	0	128	0	0	0
Htz EZ Sollwert	4	1	204	2019-12-14	06:00	2	20.0	0	128	0	0	0
Htz EZ Sollwert	5	1	204	2019-12-14	22:00	2	17.0	0	128	0	0	0
Htz WZ Sollwert	6	1	201	2019-12-15	06:15	2	21.0	0	128	0	0	0
Htz WZ Sollwert	7	1	201	2019-12-14	22:00	2	0.0	0	128	0	0	0
Htz Dusche Sollwert	8	1	198	2019-12-14	21:30	2	21.0	0	128	0	0	0
Htz Dusche Sollwert	9	1	198	2019-12-14	23:00	2	0.0	0	128	0	0	0
Htz MakeLab Sollwert	10	1	195	2019-12-14	19:00	2	0.0	0	128	0	0	0

Example Timer Definition

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

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 Monitoring

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

1	Motherboard	Yes	Motherboard sensors
Showing 1 to 8 of 8 entries			
		Update	Delete
Enabled: <input checked="" type="checkbox"/> Name: <input type="text" value="Motherboard"/> Type: <input type="text" value="Motherboard sensors"/> Data Timeout: <input type="text" value="Disabled"/> <small>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!</small>			

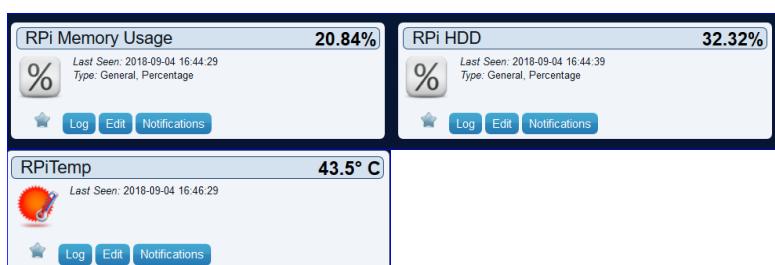
Devices derived from Hardware Motherboard

	Idx	Hardware	ID	Unit	Name	Type	SubType	
<input type="checkbox"/>	1	Motherboard	0000044C	1	RPi Memory Usage	General	Percentage	20.93%
<input type="checkbox"/>	2	Motherboard	0000044E	1	HDD /boot	General	Percentage	1.5%
<input type="checkbox"/>	3	Motherboard	0000044F	1	RPi HDD	General	Percentage	32.32%
<input type="checkbox"/>	4	Motherboard	0001	1	RPiTemp	Temp	LaCrosse TX3	44.0 C
<input type="checkbox"/>	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 send!

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
}
```

River Elbe Tide

Purpose

To monitor the tide of the river "Elbe" at location "Schulau" near Hamburg (Germany).

Solution

A Node-RED flow is used to obtain, every 5 minutes river "Elbe" information via HTTP request 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 "Abs NHN" value is compared to the previous "Abs NHN" value to determine the tide Ebb or Flood.

Only the river level "Abs NHN" is used for now.

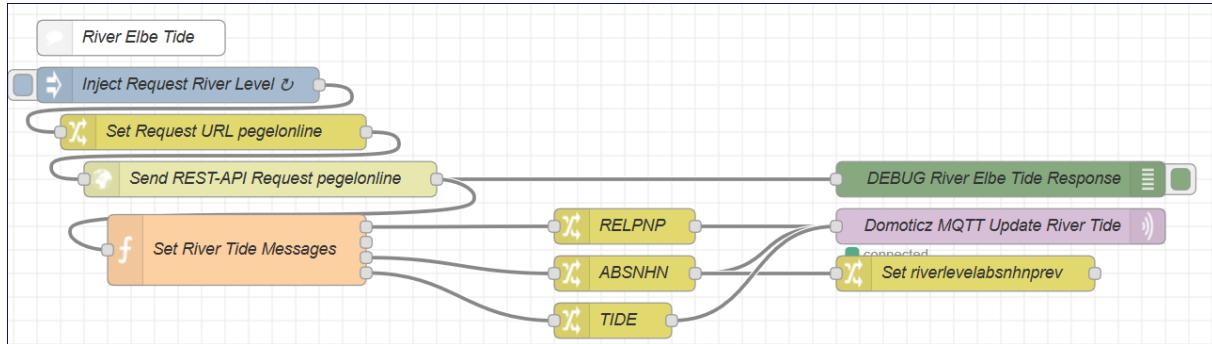
Example HTTP JSON Response

From which two keys are used in the Node-RED Function Node "Set River Tide Messages" (see below):

```
var riverlevelrelpnp = msg.payload.timeseries[0].currentMeasurement.value;
var riverlevelpnp = msg.payload.timeseries[0].gaugeZero.value;
```

```
Response: {
  "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": "2019-09-15T11:50:00+02:00",
        "value": 393.0,
        "trend": 1,
        "stateMnwMhw": "unknown",
        "stateNswHsw": "unknown"
      },
      "gaugeZero": {
        "unit": "m. ü. NHN",
        "value": -5.006,
        "validFrom": "2001-11-01"
      }
    }
  ]
}
```

Node-RED Flow



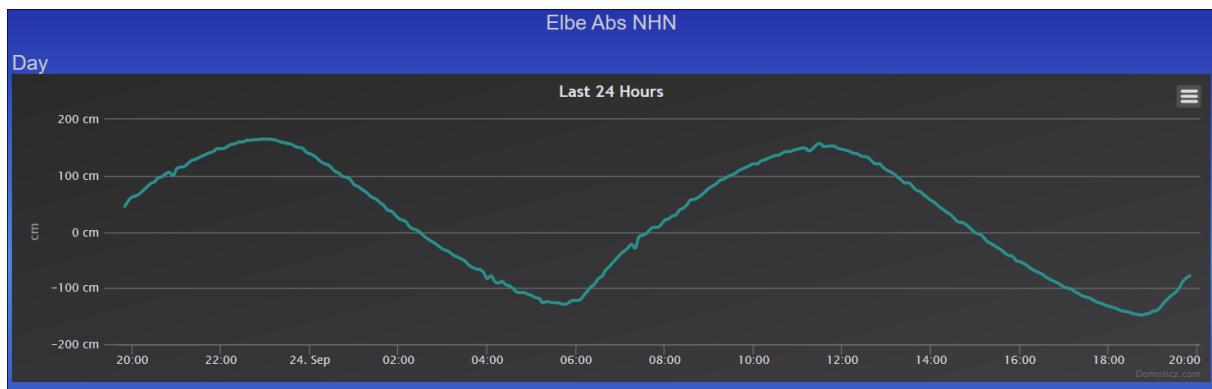
Domoticz Configuration

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
188	VirtualSensors	00082188	1	Elbe Tide	General	Text	Ebbe
187	VirtualSensors	00082187	1	Elbe Rel PNP	General	Custom Sensor	418 cm
186	VirtualSensors	00082186	1	Elbe Abs NHN	General	Custom Sensor	-82 cm

The devices have been added to the newly created roomplan “River Elbe”.



Nice tide graph.



Outlook

A notification could be given if very high tide has reached – threshold for example at 200 cm.

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](#).

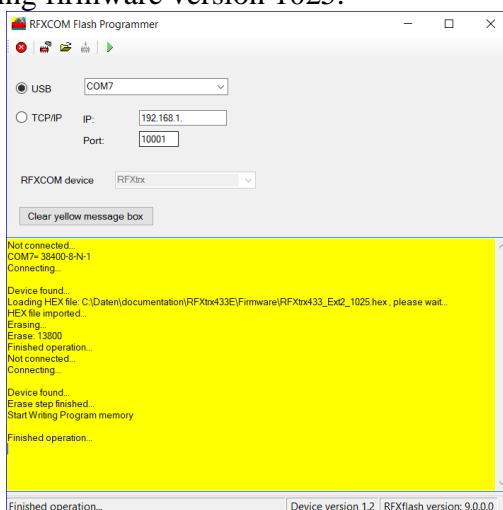
Note

Use the same procedure for updating the firmware, i.e.

Steps to install/upgrade

- Shutdown the Domoticz server (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 server
- Reboot the Domoticz server
- Open Domoticz in web 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

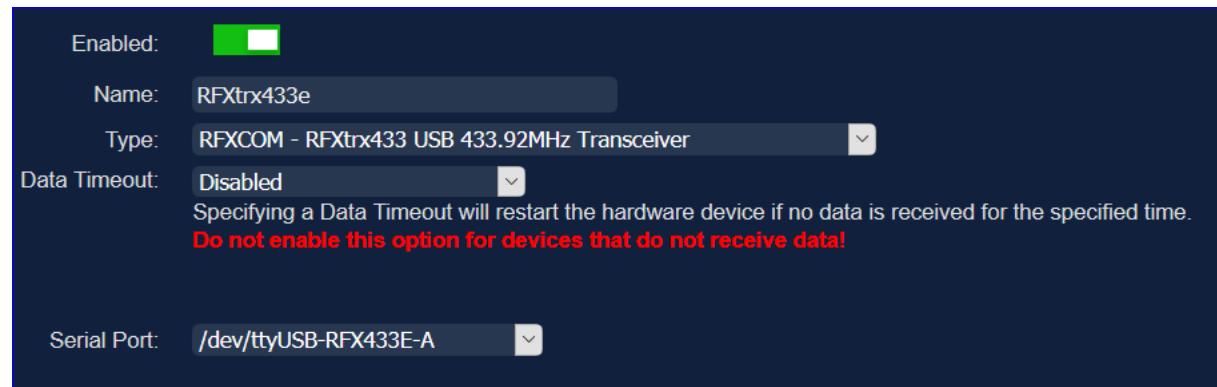
The hardware device is added / updated via **Settings > Hardware**.

4	RFXtrx433e	Yes	RFXCOM - RFXtrx433 USB 433.92MHz Transceiver Version: Ext2/1022 Set Mode	/dev/ttyUSB-RFX433E-A	Disabled
---	------------	-----	--	-----------------------	----------

Note

When updating the firmware, ensure to use the version Ext2/NNNN.

The firmware can be downloaded [here](#).



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 cmd = 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.

Soil Moisture (Plugin, Tinkerforge)

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.

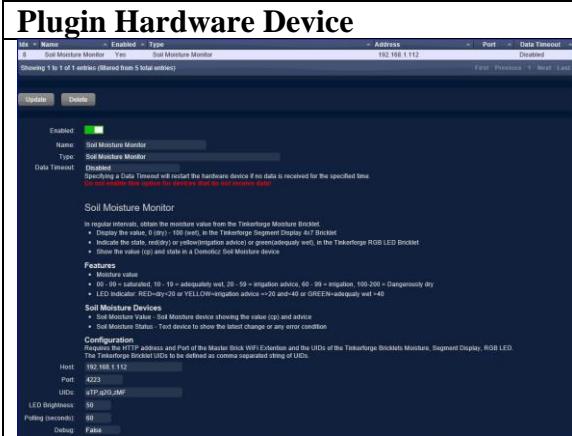
The Moisture value is converted to a range 0 (dry) – 100 (saturated) and displayed in a Tinkerforge Segment Display 4x7.

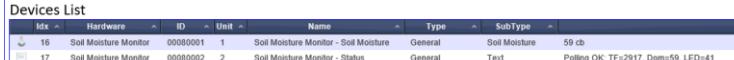
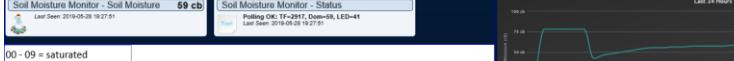
In addition, a Tinkerforge RGB LED bricklet indicates the state RED (dry) or YELLOW (irrigation advice) or GREEN (saturated).

A prototype device is built and tested on a mini palm tree in author's MakeLAB.
It is planned to create a weather proof 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
 <p>The screenshot shows the 'Hardware' tab of the Domoticz configuration interface. It lists two entries for 'Soil Moisture Monitor'. The first entry has 'Enabled' checked, 'Name' set to 'Soil Moisture Monitor', 'Type' set to 'Soil Moisture Monitor', and 'Data Timeout' set to 'Checked'. Below this, there is a detailed description of the device's features and configuration. The second entry is similar. At the bottom, there are buttons for 'Update' and 'Delete'.</p>	 <p>The photograph shows a mini palm tree in a black pot. A Tinkerforge Moisture Bricklet is attached to the soil. A Tinkerforge Master Brick with a WiFi Extension is connected to it. A Tinkerforge Segment Display 4x7 Bricklet is connected to the Master Brick, showing the moisture value '0000'. A Tinkerforge RGB LED Bricklet is also connected. The setup is mounted on a red Tinkerforge Mounting Plate 22x30. Labels identify each component: Tinkerforge Moisture Bricklet, Tinkerforge Master Brick, Tinkerforge Master Brick WiFi Extension, Tinkerforge Master Brick Power Supply, Tinkerforge Segment Display 4x7 Bricklet Value:0000, and Tinkerforge RGB LED Bricklet Color:Green.</p>	

Domoticz – Hardware & Soil Moisture Device	
 <p>The screenshot shows the 'Devices List' section of the Domoticz interface. It lists two entries under the 'Hardware' category: 'Soil Moisture Monitor' (ID 16) and 'Soil Moisture Monitor - Status' (ID 17). Both entries have 'Soil Moisture' as the type and 'General' as the sub-type. The status for both is 'OK'.</p>	 <p>The screenshot shows the 'Utility Sensors' section. It displays the 'Soil Moisture Monitor - Soil Moisture' sensor with a value of '59'. A note below it says '00 - 09 = saturated, 10 - 19 = adequately wet, 20 - 59 = irrigation advice, 60 - 99 = irrigation, 100-200 = Dangerously dry'. To the right, there is a 'Room Plan „Soil Moisture Monitor“ + Trend Soil Moisture' section.</p>
 <p>The screenshot shows the 'Room Plan „Soil Moisture Monitor“' section. It lists the 'Soil Moisture Monitor' device and its status. Below it is a 'Trend Soil Moisture' graph showing the moisture level over the last 24 hours.</p>	
 <p>The screenshot shows the 'Devices' section. It lists the 'Soil Moisture Monitor' device and its status. Below it is a 'Room Plan „Soil Moisture Monitor“' section.</p>	

Stock Quotes

Purpose

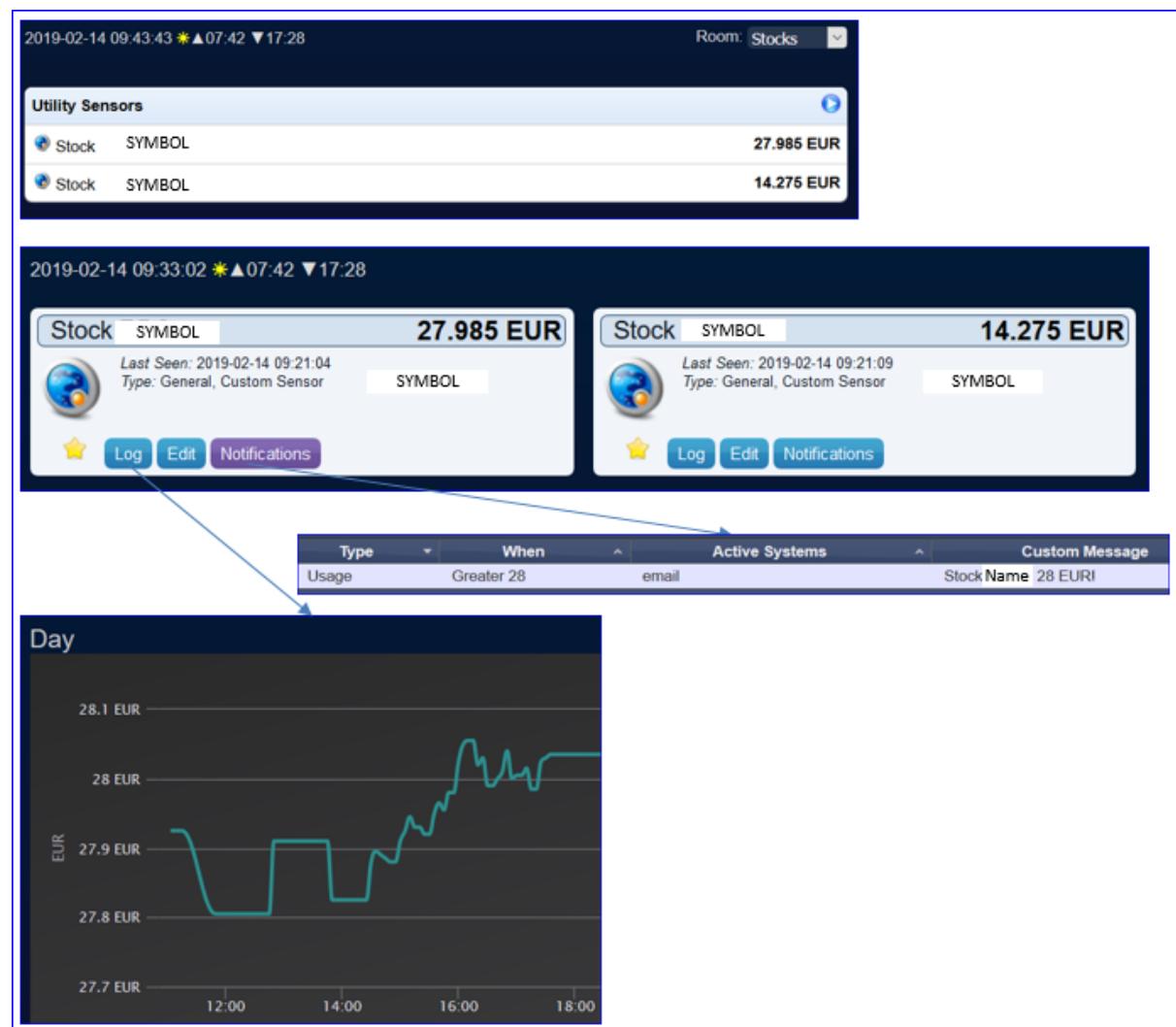
To monitor stock quotes (values):

- 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 replaced 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 Entry for a single stock

```

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

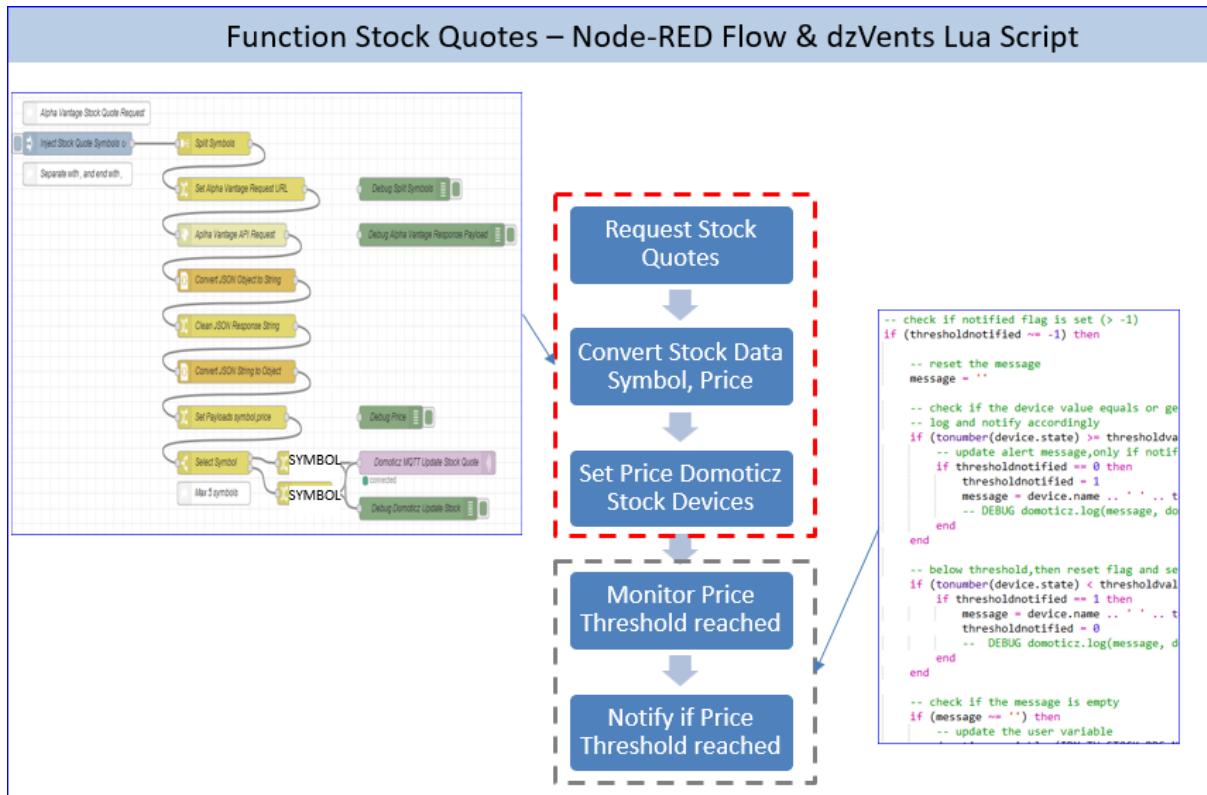
```

Solution Node-RED

NOT USED – replaced by the 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

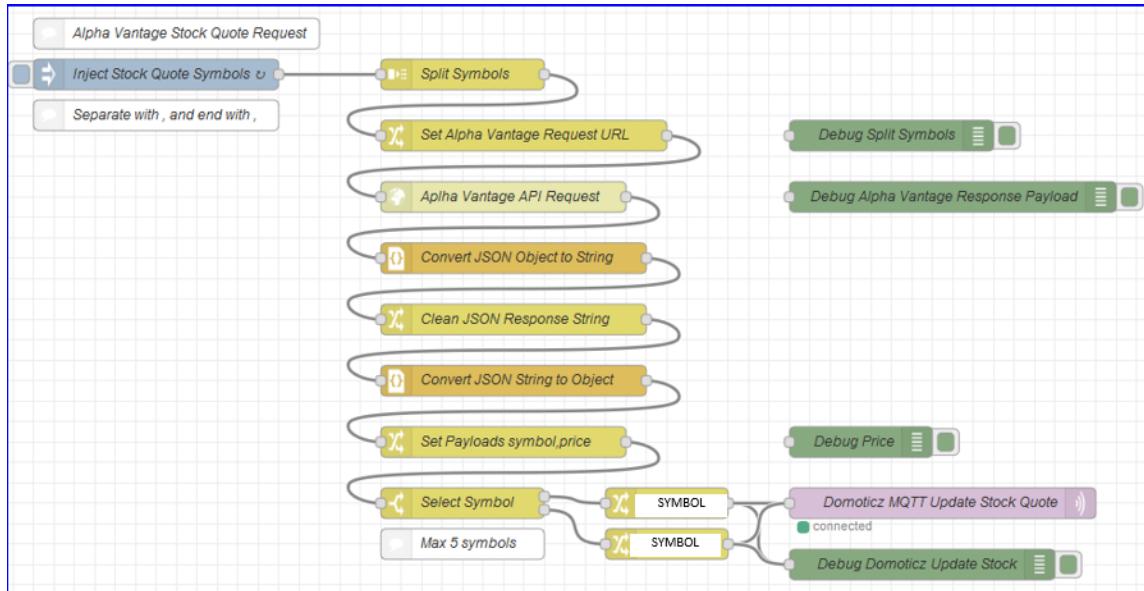
https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=STOCKA&apikey=YOURAPIKEY

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:

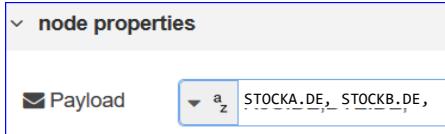
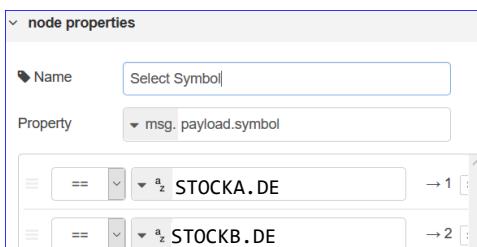
```
STOCKA.DE, STOCKB.DE,
```

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 
Select Symbol Node	Properties 
Change Node	Properties for each of the Change Nodes for a Stock 

Domoticz Log Entry

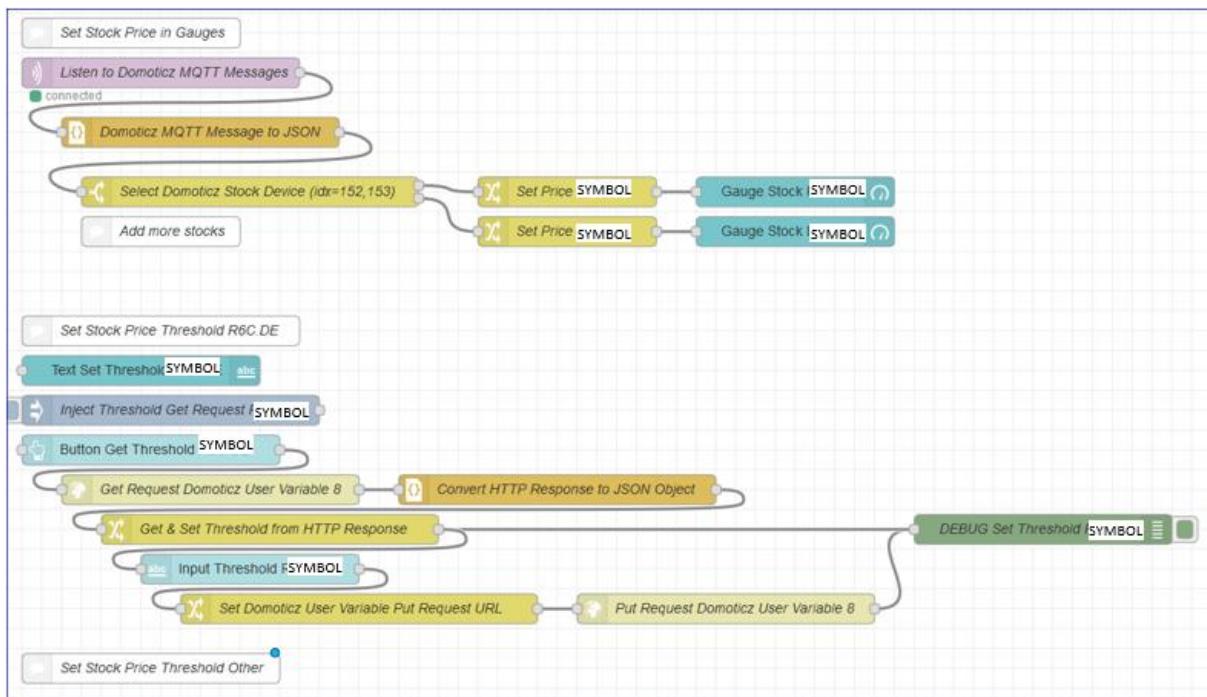
Example Domoticz log when the devices are updated.

```
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.

	<p>Two stocks, one with just a gauge to show the value, the other with the option to change the threshold.</p> <p>This solution can be enhanced, to f.e. set the threshold message or show simple trend in a graph.</p>
--	---



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: idx=12

Create Custom Sensor

Name: Stock Name Total

Axis Label: EUR

Note the idx: 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
...
```

Temperature & Humidity 433Mhz

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.

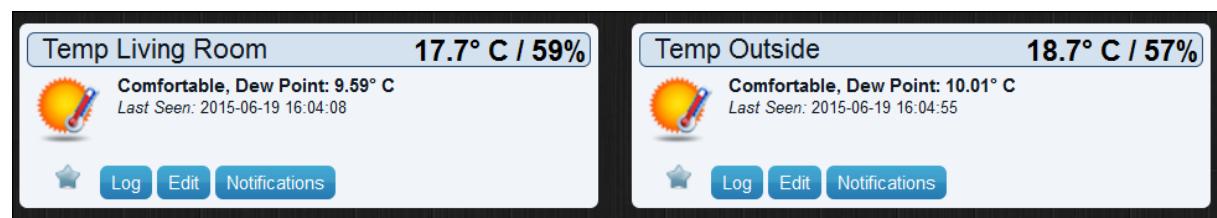
Setup

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	Tel	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



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 Timers (device type Switches only).

Note

Started off by defining timers for all devices by using Automation Events dzVents Lua scripts only.

In the mean time, more and more all kind of Swiches have been added which perform “On Time” tasks.

Rule

Dediced (as a rule) to use single device timers if a task has been performed “On Time” with a single action, i.e. switch device Off, change setpoint.

For complex tasks make use of an Automation Event with dzVents Lua script.

Device Timers

The device timers are device specific for Switches only.

They are created via the Device widget > option Timers.

Enables to perform simple tasks i.e. On | Off, change setpoint value etc.

Types supported (not the full list, these are used for the 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	Hwg Bad Sollwert	Thermostat	SetPoint	20.0

Timers

Name: Hwg Bad Sollwert				
2019-12-15 19:27:34 ⏪ 08:31 ⏴ 16:01				
Show 25 entries	Search:			
Active	Type	Date	Time	Command
Yes	On Time		06:00	Temperature, 20
Yes	On Time		11:00	Temperature, 19
Yes	On Time		23:00	Temperature, 17

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

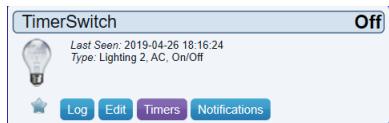
SQL Select statement to obtain all setpoint timers from the Domoticz Production database. The device names are in German.

```
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
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

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>Exploere 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	

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

Automation Events

The events are created with the Domoticz Event Editor using dzVents scripts.
Enables to create complex tasks for all devices.

Below some examples. More Automation events sing timers can be found in functions like Coffee Machine Monitor, Days-To-Go, Stock Quotes, Waste Calendar.

Example Astro Info

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
121	VirtualSensors	00082121	1	Sonne (Tageslänge)	General	Text	04:53 - 21:55 (17:02)

Sonne (Tageslänge)

04:53 - 21:55 (17:02)
Last Seen: 2019-06-28 01:00:00
Type: General, Text

★ Log Edit

```
-- [[
astro_info.lua
Display the sunrise and sunset time plus daylength in a virtual text sensor.
Updated once a day.
]]-- 

-- Idx of the devices
local IDX_SUNRISESET = 121

return {
  on = { timer = { 'at 01:00' } },
  execute = function(domoticz, timer)
    local daylength = domoticz.time.sunsetInMinutes - domoticz.time.sunriseInMinutes
    local msg = domoticz.helpers.MinutesToClock(domoticz.time.sunriseInMinutes).. ' - ' ..
domoticz.helpers.MinutesToClock(domoticz.time.sunsetInMinutes).. ' (' ..
domoticz.helpers.MinutesToClock(daylength).. ')'
    domoticz.devices(IDX_SUNRISESET).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 {  
    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)  
        if (timer.trigger == '30 minutes before sunset') then hueswitchon(domoticz) end  
        if (timer.trigger == 'at 23:00') then hueswitchoff(domoticz) 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://192.168.1.225/config/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
}
```

Waste Calendar

Purpose

To display 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.

A room plan “Termine” is defined (Domoticz GUI Setup > More Options > Plans > Roomplan).

Example Room Termine

Sensor	Date
Biotonne Pi	06-09-2018
Gelber Sack Pi	30-08-2018
Papier Pi	21-09-2018
Restmüll Pi	30-08-2018
Schadstoff Pi	08-09-2018
Biotonne Ho	27-08-2018
Gelber Sack Ho	28-08-2018
Papier Ho	17-09-2018
Restmüll Ho	27-08-2018

Waste Type	Date
Biotonne Pi	06-09-2018
Gelber Sack Pi	30-08-2018
Papier Pi	21-09-2018
Restmüll Pi	30-08-2018
Schadstoff Pi	08-09-2018
Biotonne Ho	27-08-2018
Gelber Sack Ho	28-08-2018
Papier Ho	17-09-2018
Restmüll Ho	27-08-2018

Devices Configuration

The devices are created as Virtual Sensors using the Dummy Hardware.

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 script event, “waste_calendar_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 within a given interval (i.e. 16 or 32 days).

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).

A date is published if within the next 13 days from the actual date.

CSV Line Syntax: idx, interval, date1, dateN

idx	NN 98, 99...
interval	Number of days between waste dates. The intervals might not be exact with 14 days or a month. To go for sure, use 16 instead of 14 16, 31
date	Format Day-Month-Year dd-mm-YYYY 09-08-2018

Example CSV file line entry

The example shows for idx 98 (“Biotonne Pi”), an interval every 16 days and number of dates for the remaining year 2018 and full year 2019.

```
98,16,13-12-2018,28-12-2018,10-01-2019,24-01-2019,07-02-2019,21-02-2019,07-03-2019,21-03-2019,04-04-2019,18-04-2019,03-05-2019,16-05-2019,31-05-2019,14-06-2019,27-06-2019,11-07-2019,25-07-2019,08-08-2019,22-08-2019,05-09-2019,19-09-2019,04-10-2019,17-10-2019,01-11-2019,14-11-2019,28-11-2019,12-12-2019,27-12-2019,09-01-2020
```

CSV File name & location

The CSV file wastecal.csv, located in folder /home/pi/domoticz/scripts/dzVents/scripts.

The file is used by a dzVents Lua script *waste_calendar_update* created using the Domoticz Script editor.

Automation Script

A dzVents Lua script vent is time triggered once per day at 00:30. See source for details.

Event name: *waste_calendar_update*

```
--[[[  
    waste_calendar_update.lua  
    Read a csv file with wastecalendar dates per device.  
    Update the device text with the date if the date is within interval specified.  
    The wastecalender file "wastecal.csv" is located in the dzVents scripts folder.  
    Line entry csv file: idx,interval,date1,dateN.  
]]--  
  
-- External modules: /home/pi/domoticz/scripts/dzVents/scripts  
local utils = require('utils')  
local msgbox = require('msgbox')  
  
-- threshold in days to notify  
-- if the difference of the current day to a wastecalendarday equals threshold  
-- then notify  
local TH_WASTECAL = 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 wastecalender file is updated once per year  
local wastecalfile = '/home/pi/domoticz/scripts/dzVents/scripts/wastecal.csv'  
local sep = ","  
  
-- read the csv file  
function readandupdate(domoticz)  
    -- read the file line by line.  
    -- each line contains an array with device[1], wasteinterval[2], dates[2...]  
    for line in io.lines(wastecalfile) do  
        -- split the line by , and assign to a table  
        local wastetable = line:split(sep)  
        local idx = tonumber(wastetable[1])  
        local interval = tonumber(wastetable[2])  
        -- flag to check if a wastedate has been found  
        local datefound = 0  
  
        -- check the dates, start at 3 because 1=idx, 2=interval and number of table entries  
        for i = 3, #wastetable do  
  
            -- get the wastedate array entry  
            local wastedate = wastetable[i]  
            -- split the wastedate into an array with 3 entries day, month, year  
            local wastedatesplit = wastedate:split("-")  
            -- calculate the daysdiff between now and the wastedate  
            local daysdiff = utils.datediffnow(wastedatesplit[1], wastedatesplit[2], wastedatesplit[3])  
            --  
            local month = tonumber(wastedatesplit[2])  
  
            -- check if daysdiff within interval
```

```

-- if daysdiff >= 0 and daysdiff < interval then
if daysdiff >= 0 and datefound == 0 then
    datefound = 1
    print (domoticz.devices(idx).name .. '(' .. idx .. ')=' .. wastedate .. ', days=' .. daysdiff
.. ', interval=' .. interval)

    -- update the device text with the new date
    domoticz.devices(idx).updateText(wastedate)

    -- 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 waste_calendar_notifier
if daysdiff == TH_WASTECAL 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 wastedate is sep or oct
if i == #wastetable and daysdiff < 0 then
    datefound = 1
    print (idx .. '=' .. wastedate .. ', days=' .. daysdiff .. ', interval=' .. interval .. ',
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

    -- check if a wastedate 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, #wastetable do
    --   print( wastetable[i] )
    -- end
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 wastecal.csv file.
The script reads a iCal file.

Script

```
# -*- coding: cp1252 -*-
## File: wastecalictstocsv.py (Python 2)
## Project: atHome
## Purpose: Create a CSV file with waste dates from an ICS iCal file Pinneberg and empty entries for
Hohenfelde
## Author: Robert W.B. Linn
## Version: 20181209

## 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 has to be merged with the wastecal.csv file located on the Domoticz server:
## /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 = []
hobiolist = []
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(wastecalendars, 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 build 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

# Example to show the content of a list
#print len(biolist)
#for wastedate in biolist:
#    print wastedate
#print csv2string(biolist)

# 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 tobe 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()

```

Webfrontend Customized

Purpose

To build a Custom Web Frontend running on a dedicated web server.

Web Front End Sample

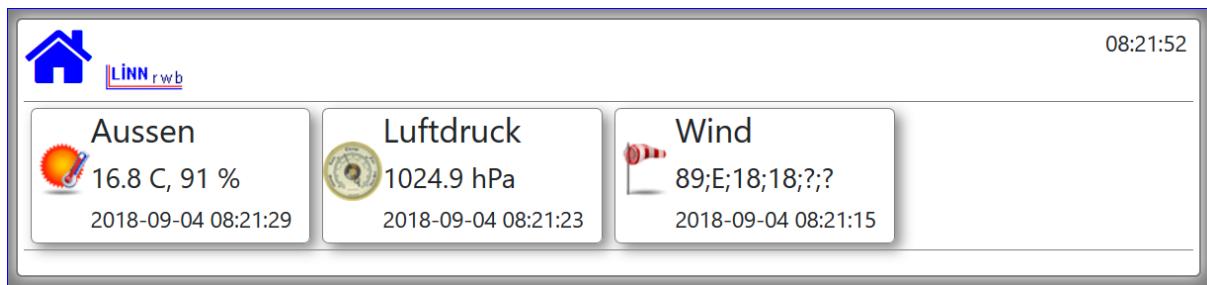
Display weather data from 3 devices and update every 5 seconds. A clock is running every second.

Devices Used

```
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 server 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>AtHome 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://rpi-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 server
            // /home/pi/domoticz/www/images
            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">'+
                ''+
                        '<h4>' + result.result[i].Name + '</h4>'+
                        '<h5>' + result.result[i].Data + '</h5>'+
                        '<h6>' + result.result[i].LastUpdate+ '</h6>'+
                    '</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.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>
```


Webradio 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 interface. In the 'Light/Switch Devices' section, there are four items: 'Volumio Play' (On), 'Volumio Refresh' (On), 'Volumio Volume' (a slider set to approximately 80, with On/Off buttons), and 'Volumio Bookmark' (On). In the 'Utility Sensors' section, there are three items: 'Volumio Current Song' (This is the song title) and 'Volumio Favorites' (This is the last bookmarked song title), both with edit icons; and 'Volumio Status' (PLAY button).

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
Volumio Status PLAY = switch Volumio Play set to On	
<input type="checkbox"/> Volumio Status	PLAY
Volumio Status STOP = if the server is started or if switch Volumio Play is set to Off	
<input type="checkbox"/> Volumio Status	STOP
Volumio Status OFF = server not reachable (network) or server turned off (shutdown)	
<input type="checkbox"/> 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 web 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

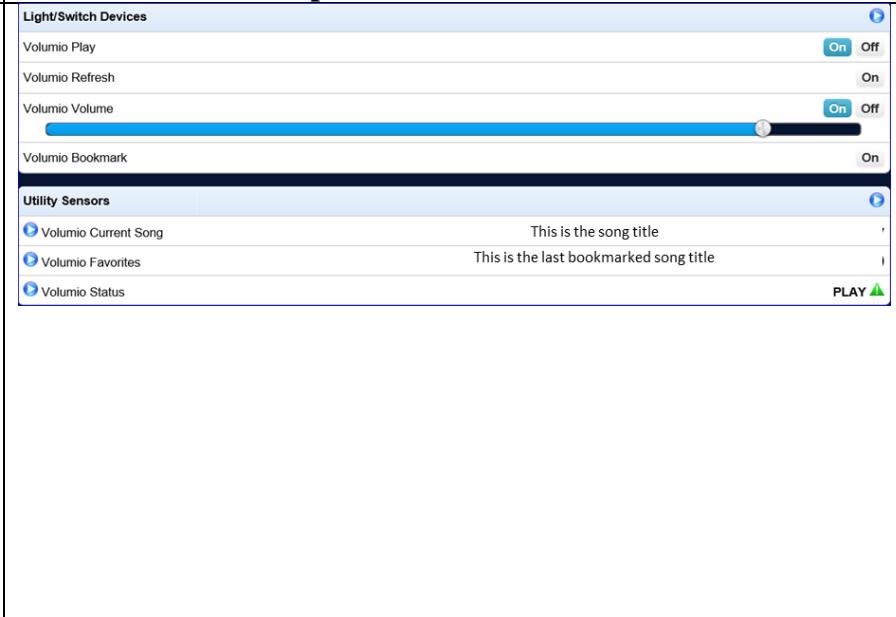
Devices

Idx	Name	Type	SubType	Function
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><TODO>Find solution how to shutdown.</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 Showing 1 to 5 of 5 entries Edit Delete Devices (Select Plan first to Edit...) Show [dropdown] entries <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> 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>Light/Switch Devices</p> <ul style="list-style-type: none"> Volumio Play (On) Volumio Refresh (On) Volumio Volume (On) Volumio Bookmark (On) <p>Utility Sensors</p> <table> <thead> <tr> <th>Icon</th> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Bluetooth</td> <td>Volumio Current Song</td> <td>This is the song title</td> </tr> <tr> <td>Bluetooth</td> <td>Volumio Favorites</td> <td>This is the last bookmarked song title</td> </tr> <tr> <td>Bluetooth</td> <td>Volumio Status</td> <td>PLAY </td> </tr> </tbody> </table>	Icon	Name	Description	Bluetooth	Volumio Current Song	This is the song title	Bluetooth	Volumio Favorites	This is the last bookmarked song title	Bluetooth	Volumio Status	PLAY 
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																														
Icon	Name	Description																													
Bluetooth	Volumio Current Song	This is the song title																													
Bluetooth	Volumio Favorites	This is the last bookmarked song title																													
Bluetooth	Volumio Status	PLAY 																													

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

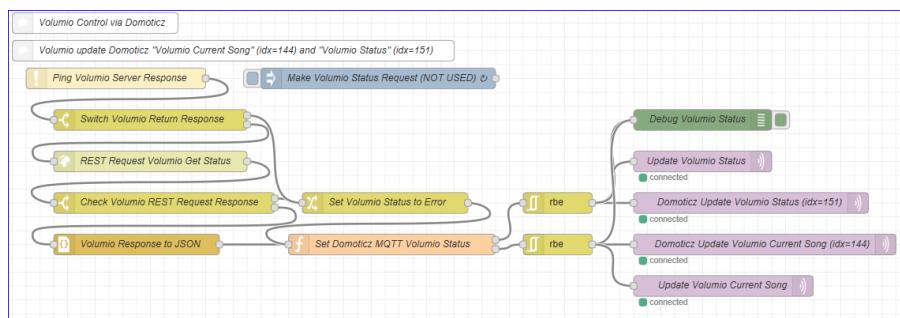
Switch Volumio Refresh

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 shutdown 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

Idx	Name
149	Volumio Play
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
- 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;).

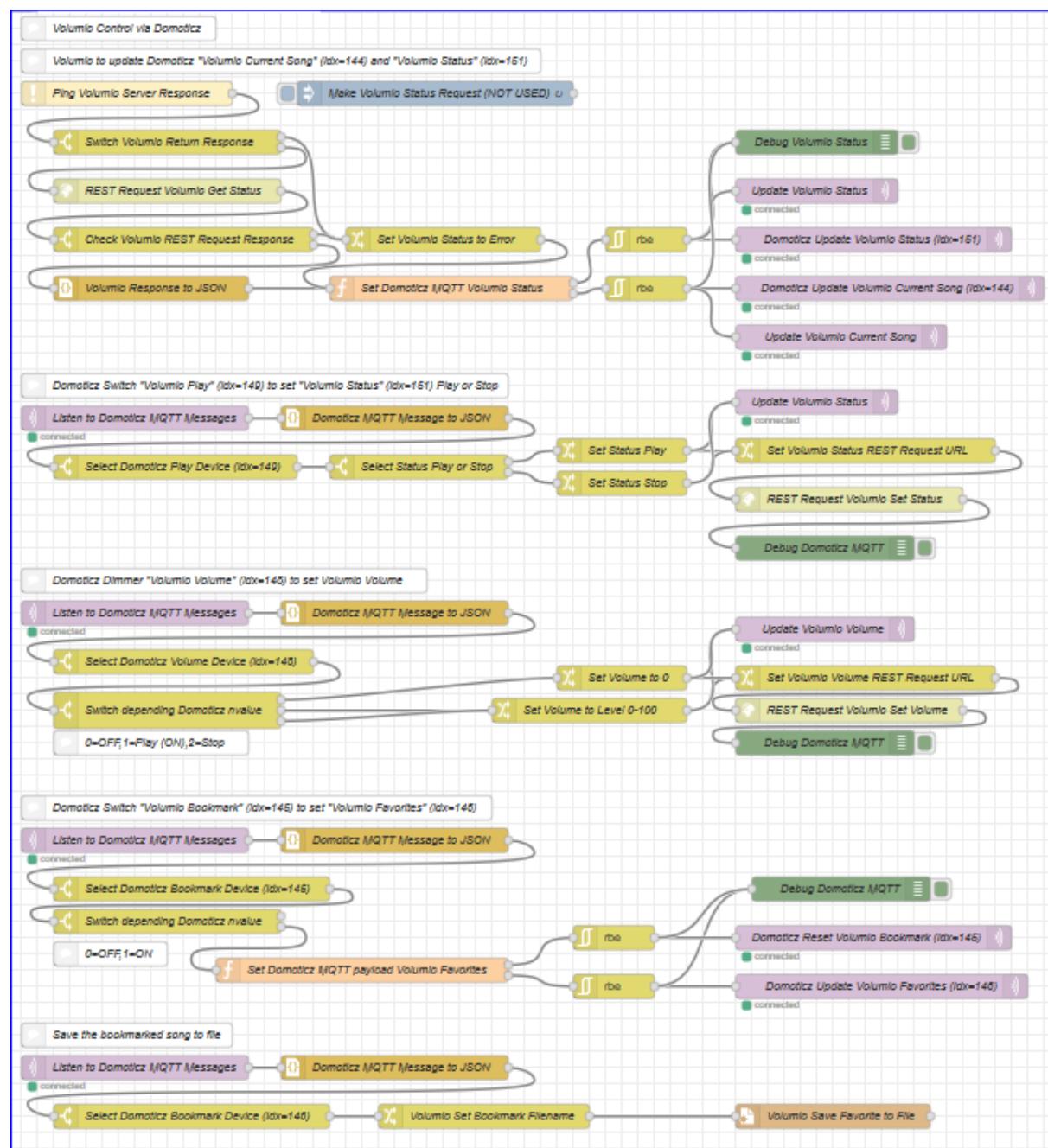
The output of the *Function Node* goes to a *MQTT Out Node*, which connects to the Domoticz MQTT broker (localhost:1883) and publishes the messages.

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

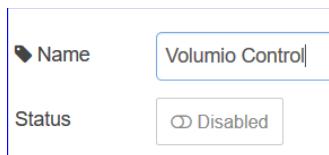


Notes

- 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 web 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 “make”.

Disable Flow

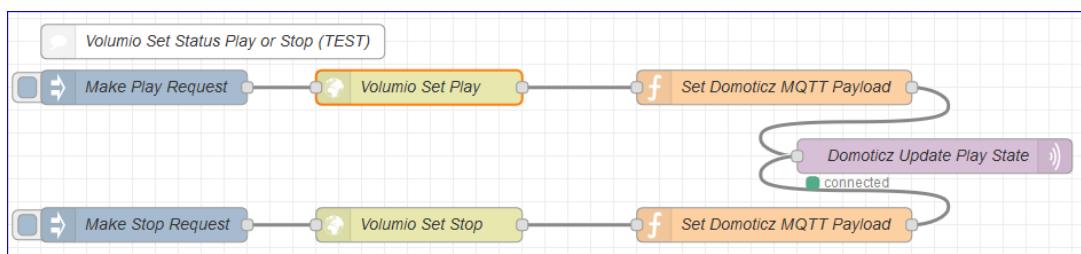
The Node-RED flow *Volumio Control* can be enabled and disabled by clicking on the Tab *Volumio Control*:



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*.



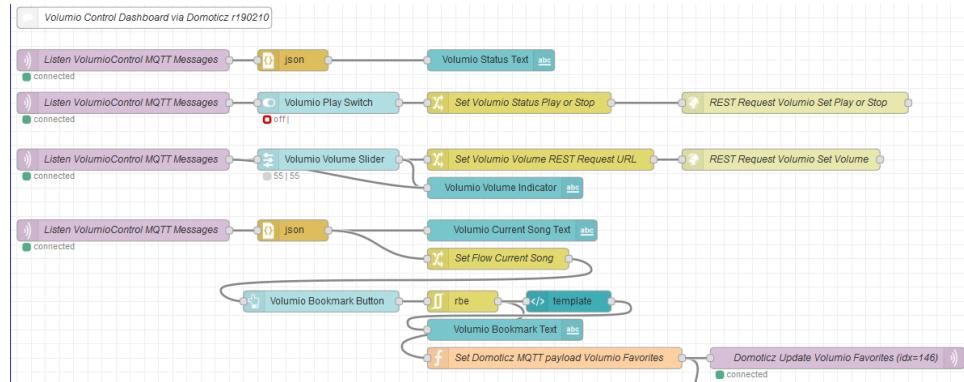
Note

This flow can be simplified using Change Node to set the status to then trigger Volumio and update the Domoticz Device.

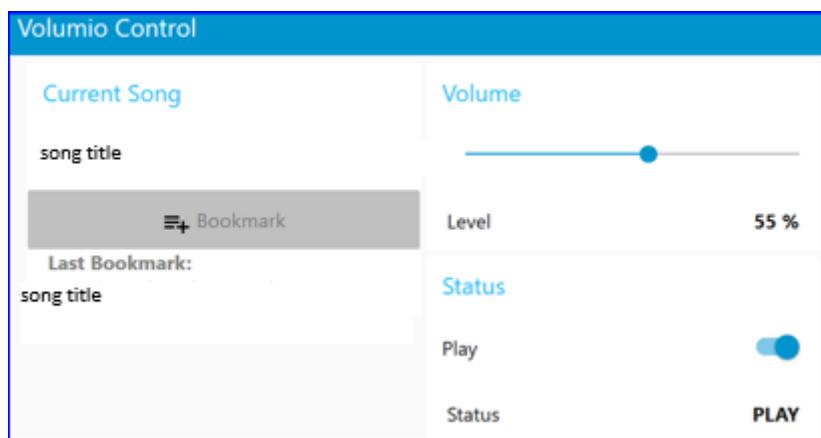
Volumio Control Dashboard

For Node-RED, [Dashboard UI](#) nodes are available, which could be used to create a Domoticz Alternate Dashboard (accesses via web browser).

Created a flow *Volumio Control Dashboard*, to explore the use of Dasboard UI nodes.



Displayed in web browser with <http://domoticz-ip-address: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
Topic	volumiocontrol/status
Payload Sample	{"command":"udevice","idx":151,"nvalue":1,"svalue":"PLAY"}
Usage	To set the status of Volumio in a Dashboard ui_text node:
Topic	volumiocontrol/currentsong
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}}
Topic	volumiocontrol/volume
Payload Sample	NN containing the level
Usage	To set the volume level of the slider and text of the indicator. Value format: {{msg.payload}} %

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 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 measured temperature is not used as the temperature is 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 the 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 Entry

```
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 dzVents events.

In addition, used by Node-RED flows to f.e. set a new value.

Node-RED is an option, to configure User Variables other then the Domoticz GUI.

Syntax

The naming convention for a User Variable uses as syntax

- 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

List

SQL

Created list direct from the Domoticz database via SQL statement:

```
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	Reset	2018-09-13 13:52:44
8	TH_STOCK_RDS	1	27	2019-03-03 09:53:40
9	TH_STOCK_RDS_NOTIFIED	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_COUNT	0	1500	2019-05-17 11:17:39
13	DEF_SECURITY_ALARM	0	0	2019-05-21 09:20:00

Note

User variables are stored in table UserVariables of the Domoticz database domoticz.db.

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.

```
import urllib.request
import json
'''

{
    "result" : [
        {
            "LastUpdate" : "2018-09-11 19:36:34",
            "Name" : "TH_RPI_TEMPERATURE",
            "Type" : "0",
            "Value" : "50",
            "idx" : "3"
        },
        {
            "LastUpdate" : "2018-09-05 11:15:20",
            "Name" : "TH_RPI_HDDUSAGE",
            "Type" : "0",
            "Value" : "70",
            "idx" : "4"
        },
        ...
    ],
    # define url
    url = 'http://rpi-address:8080/json.htm?type=command&param=getuservariables'
    # request list
    try:
        req = urllib.request.Request(url)
    except HTTPError as e:
        print('HTTP Error code: ', e.code)
    except URLError as e:
        print('URL Reason: ', e.reason)
    else:
        # parsing response
        response = urllib.request.urlopen(req).read()
        content = json.loads(response.decode('utf-8'))
        # get the status
        print(content['status'])
        # 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=';')
```

Run

```
pi@DoPro:~/domoticz/scripts/python $ python3 user_variables_list.py
```

Result

```
OK
Name;Idx;Value
TH_RPI_TEMPERATURE;3;50
TH_RPI_HDDUSAGE;4;70
TH_RPI_MEMORYUSAGE;5;80
IDX_ALERTMSG;6;55
DEF_ALERTMSG;7;No alert
TH_STOCK_RDS;8;27
TH_STOCK_RDS_NOTIFIED;9;0
TH_AMBIENT_LIGHT;10;111
DEF_VOLUMIO_TRACK;11;0
DEF_STOCK_RDS_COUNT;12;1500
DEF_SECURITY_ALARM;13;0
TH_ALERTTOEMAIL;14;4
```

Usage in 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')
```

Usage in 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-address/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-  
address/json.htm?type=command&param=updateuservariable&vname=TH_STOCK_STOCKA&vtype=1&vvalue=29
```

Response

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

Usage in Node-RED

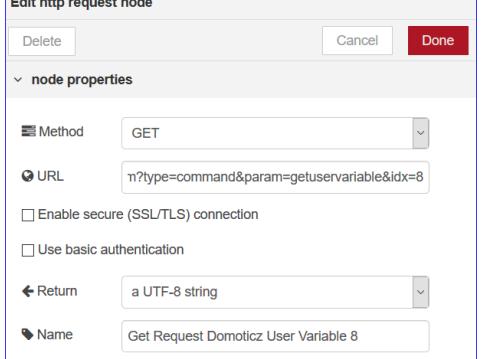
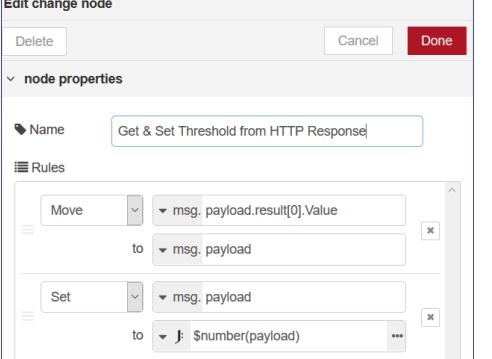
Examples on how to use the User Variables in Node-RED.

For Node-RED, there are several possibilities.

This example uses HTP Response Nodes.

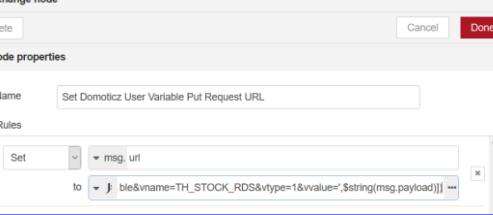
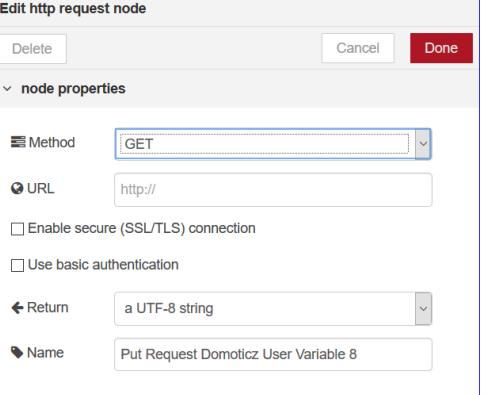
Read Value

Same request as previous used in 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>
--	---

Usage in 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

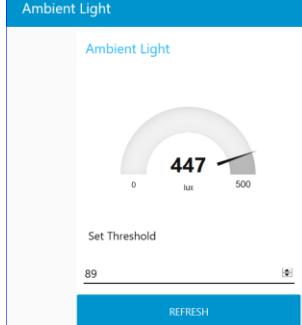
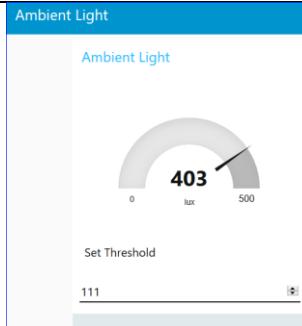
Define a User Variable (like TH_AMBIENT_LIGHT_SWITCH) holding the idx of the light (better switch) to be turned ON when the value is below threshold. A ui_dropdown list could be defined to select the light (switch) and update the User Variable.

Node-RED Dashboard

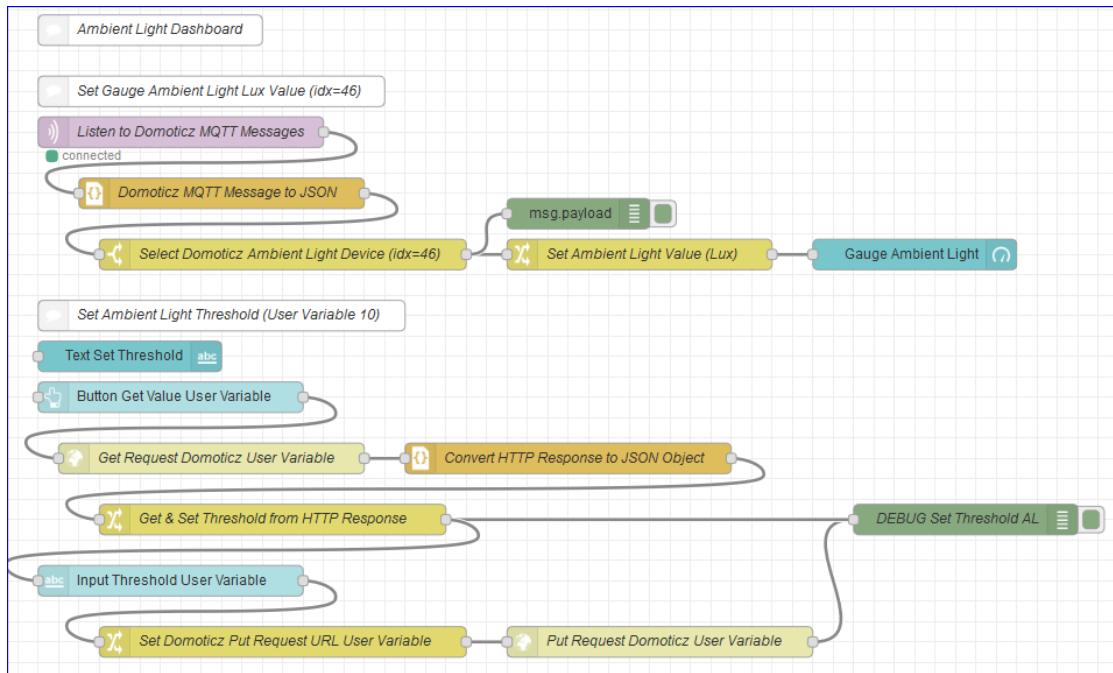
A ui_gauge node displays the actual Ambient Light Lux value, which is obtained from MQTT payload of the Ambient Light device. Example from which property "svalue1" is used by the gauge:

```
{"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.

 <p>Value 89 obtained via Refresh</p>	<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> <p>Domoticz User Variable List (extract for idx=10)</p>	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						
 <p>Changed to 111 and pressed enter</p>	<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> <p>Refresh the page in the browser, shows the updated value for the User Variable</p>	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¶m=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¶m=updateuservariable&vname=TH_AMBIENT_LIGHT&vtype=0&vvalue=NNNN

Note

vtype is 0 which is an Integer.

HTTP Response (JSON Format)

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

Custom Icons

Custom icons are used for several devices.

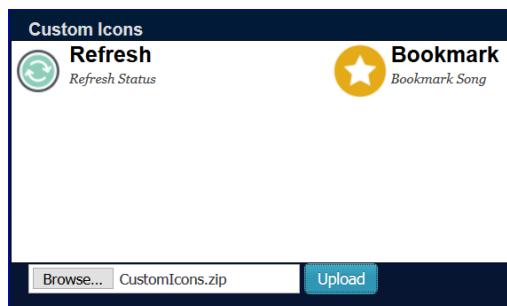
Note

Not all devices support icon customizing, i.e. General, Text.

Sharing an example for two icons *Refresh* and *Bookmark* used by the Function Volumio.

Steps

1. Open Domoticz GUI > Setup > More Options > Custom Icons.
2. Browse for the file *CustomIcons.zip* and upload.
3. After upload, the custom icons are shown and listed when editing the icon on a device (via Widget > Edit).



Each icon has 3 files (size width pixel x height pixel):

- Icon.png (size 16x16 pixel)
- Icon_On.png (size 48x48 pixel)
- Icon_Off.png (size 48x48 pixel)

Note

The icons Icon_On.png & Icon_Off.png are mandatory (even if the device is not a switch type)!

The archive *CustomIcons.zip* contains the files.

- icons.txt
- Refresh.png
- Refresh48_On.png
- Refresh48_Off
- Bookmark.png
- Bookmark48_On.png
- Bookmark48_Off.png

The content of the file *icons.txt* for the two icons Refresh and Bookmark:
(icon file name, icon interface name, icon description)

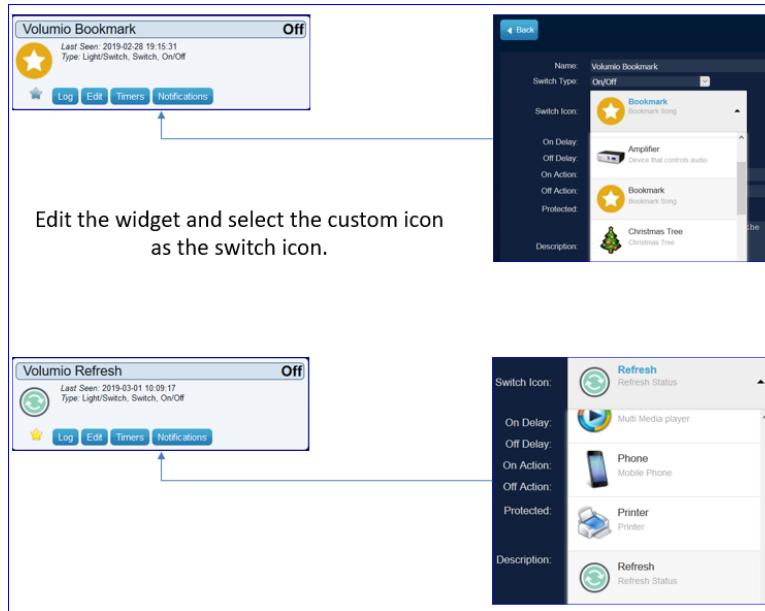
```
Refresh;Refresh;Refresh Status
Bookmark;Bookmark;Bookmark Song
```

Note

Ensure the first two entries match the icon file name!

Add Icon to a Device Widget

The icons are added to the device widgets by selecting from the icon list (edit):



Get List of Icons

HTTP API Request

```
http://rpi-ip-address:8080/json.htm?type=custom_light_icons
```

Returns a JSON string

This is an extract of the list showing the two icons added with their idx. The icons are selected from the file /home/pi/domoticz/www/switch_icons.txt and Domoticz database query 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 in icons.txt, has table 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 of the Custom Images

Domoticz Database

This is an example querying, the Domoticz database on the development server, for custom images. Only 1 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-address: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 command to set a Custom Icon

Setting a custom icon via SQL example.

Set the custom icon “Air Quality” with idx=101, to the device with idx 41.

```
UPDATE DeviceStatus SET CustomImage = 101 WHERE ID=41
```

Explore

This section 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 section 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.

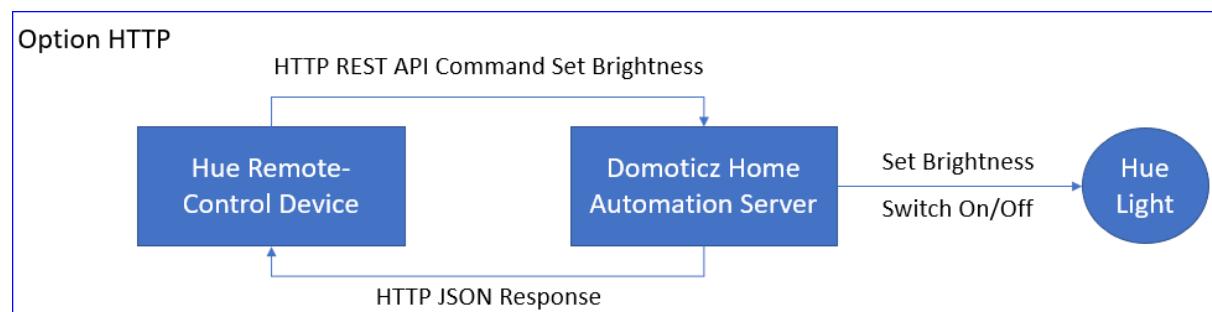
The remote-control device requires communication with the Domoticz Home Automation server which is integrated in the SMART Home network.

Option HTTP

Domoticz allows you to interact with all devices using HTTP API Commands generating JSON returns.

This means, it is possible to sent commands to the Domoticz server to act upon and receive a response back.

Sending a command and view the server response can be easily done via web 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 web browser

Goal

Set the brightness to 50% for Hue Device with Domoticz idx=118.

HTTP API Request URL

```
http://rpi-domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=50
```

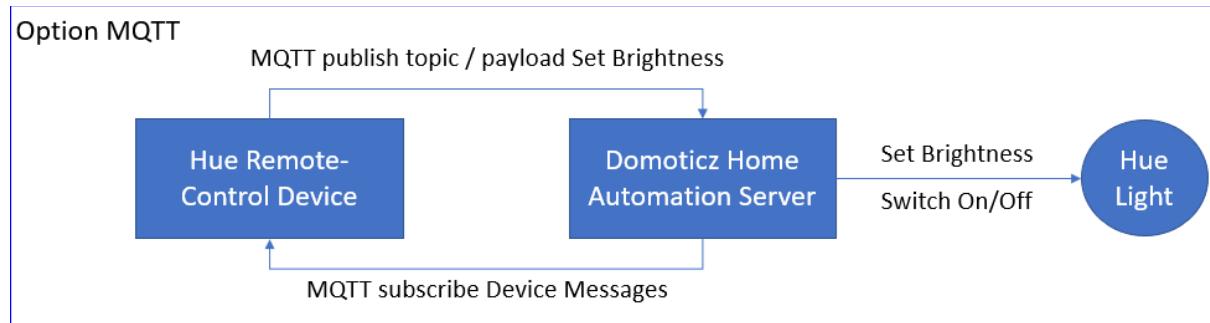
HTTP JSON Response

```
{  
  "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 Entry:
# 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://rpi-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://rpi-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 Entry

```
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"

Events - Overview

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's no longer possible to enable one without the other.

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.

Event Database Tables

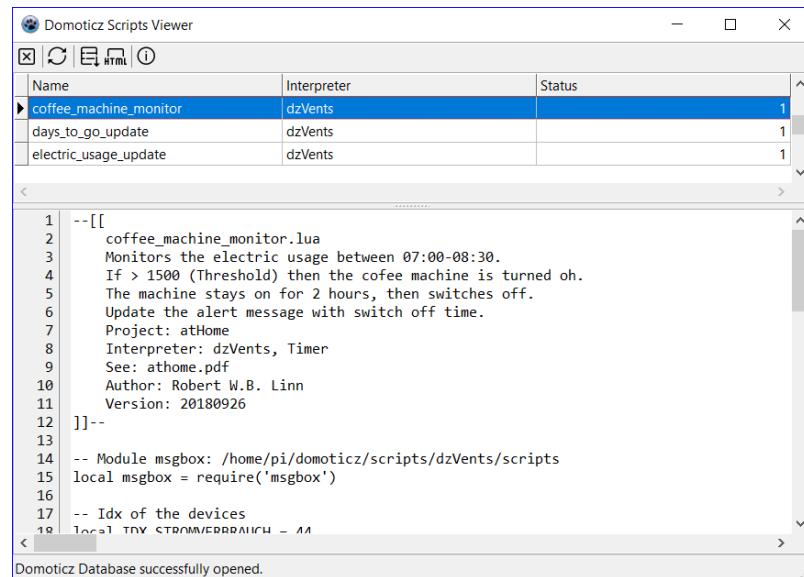
The events which are created using the Domoticz Event Editor, are stored in tables [EventMaster] and [EventRules]

```
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 the 'Domoticz Scripts Viewer' application window. At the top, there are tabs for 'Domoticz Scripts Viewer', 'File', 'Edit', 'HTML', and 'Help'. 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 displaying the script content for 'coffee_machine_monitor.lua'. The script 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 TNY_STROMVERBRAUCH = 11

```

At the bottom of the text area, it says '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.



Event Sample 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 ≥ 70 (threshold), write a message to the Domoticz Log.

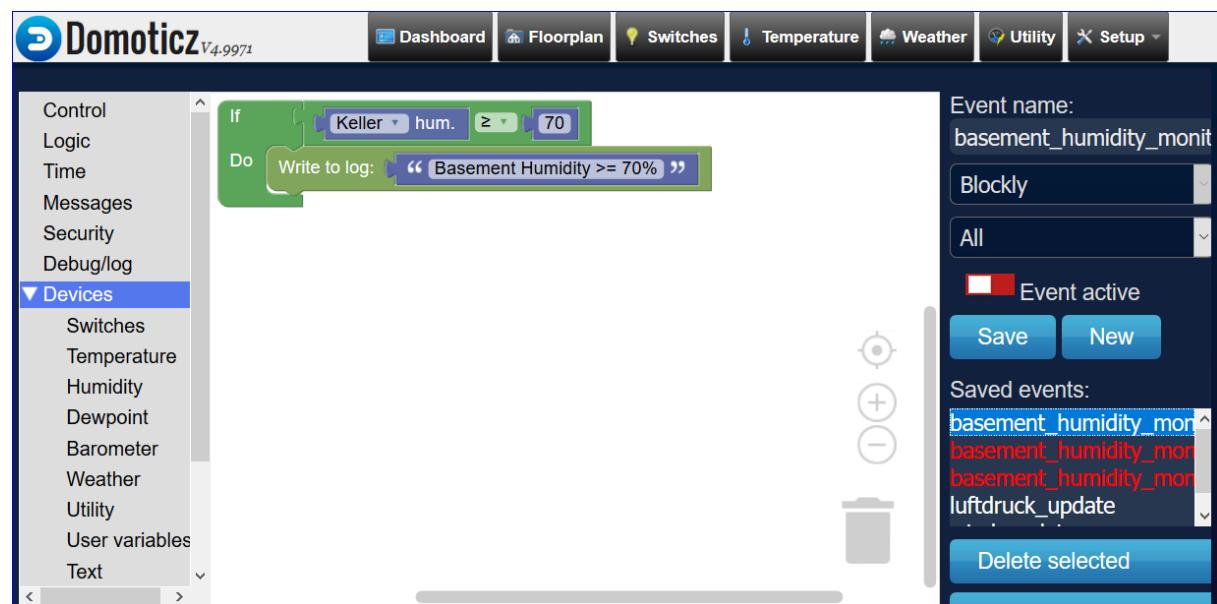
Note

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 Entry

```

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 -- Split a string by delimiter
14 function split(s, delimiter)
15     result = {}
16     for match in (s..delimiter):gmatch("(.-)"..delimiter) do
17         table.insert(result, match);
18     end
19     return result;
20 end
21
22 -----
23 -- Domoticz IDX and names of the needed devices
24 local KellerIdx = 11;
25 local KellerName = "Keller";
26
27 -- Thresholds
28
29

```

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 Entry

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 Entry

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.

The screenshot shows the Domoticz software interface. On the left, the Python script editor displays a script named 'basement_humidity_monitor_p'. The script includes comments, imports (domoticz, DomoticzEvents), device definitions (KellerIdx=11, KellerName='Keller'), and logic to monitor humidity. On the right, the event configuration panel is open, showing the event name as 'basement_humidity_monitor_p', selected as 'Python' and 'Device', with the 'Event active' checkbox checked. Below the event panel, a list of saved events shows multiple entries for 'basement_humidity_monitor_p'.

```

5  #<interpreter>: device
6  #@see: readme.txt
7  @author: Robert W.B. Linn
8  @version: 20180902
9  -----
10 """
11
12 # Imports
13 import domoticz
14 import DomoticzEvents as DE
15
16 # Devices
17 KellerIdx = 11
18 KellerName = 'Keller'
19
20 # Set the threshold to monitor the humidity (%RH)
21 HumidityThreshold = 70
22
23 * if DE.changed_device_name == KellerName:
24     # Test Log
25     # 2018-09-02 11:32:46.085 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType: 0, s_
26     DE.Log("Python: Changed: " + DE.changed_device.Describe())
27     # Test s_value
28     # 2018-09-02 11:32:46.085 18.7;75;3
29     svalue = DE.Devices[KellerName].s_value
30     DE.Log(svalue)
31     # Get the humidity, which is at the 2nd entry of the svalue, i.e. 75
32     # Split the svalue by ;. The result is a list with length 3, data[0]=T,data[1]=H,data[2]=Comfort
33     data = svalue.split(";")
34 <

```

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_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 :
    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 Entry

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
project: AtHome
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 server 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 ({}%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 to escape 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 Entry

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

The event system Domoticz Easy Events, dzVents, is the next generation of Domoticz Events.

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 vars with the same name etc.

Device Properties

To analyse the properties of a device, use device.dump(), to then select from the list.

Script

This example checks if the defined devices (see idx of the devices) have changed, to then dump the properties but also show 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 Entry

```
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.

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 Entry

```
-- 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”.

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)
```

ESP8266

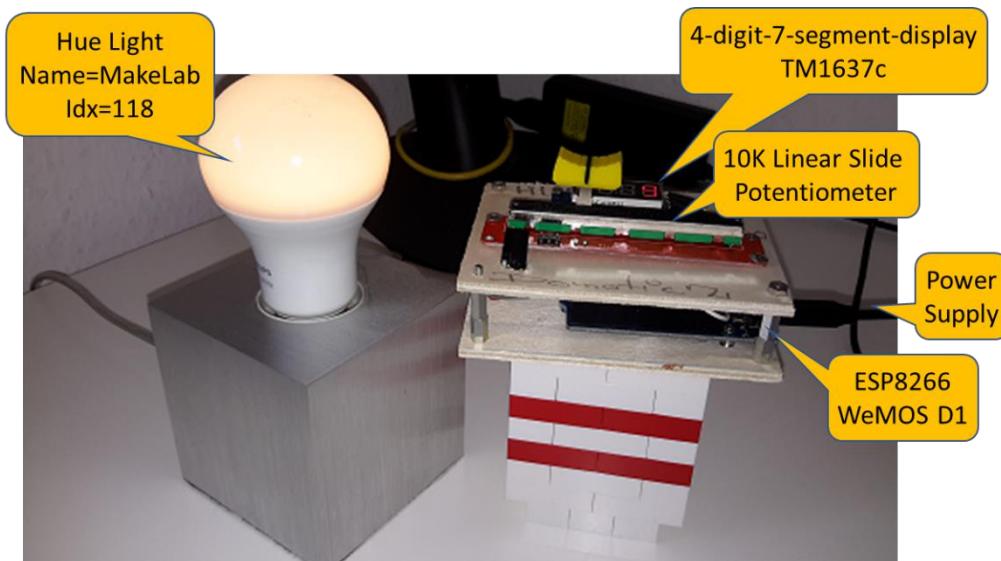
Purpose

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 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 Needed

- 1x ESP8266 WeMOS D1 R1
- 1x 4-digit-7-segment-display TM1637
- 1x 10K Linear Slide Potentiometer
- 1x LED [on board]

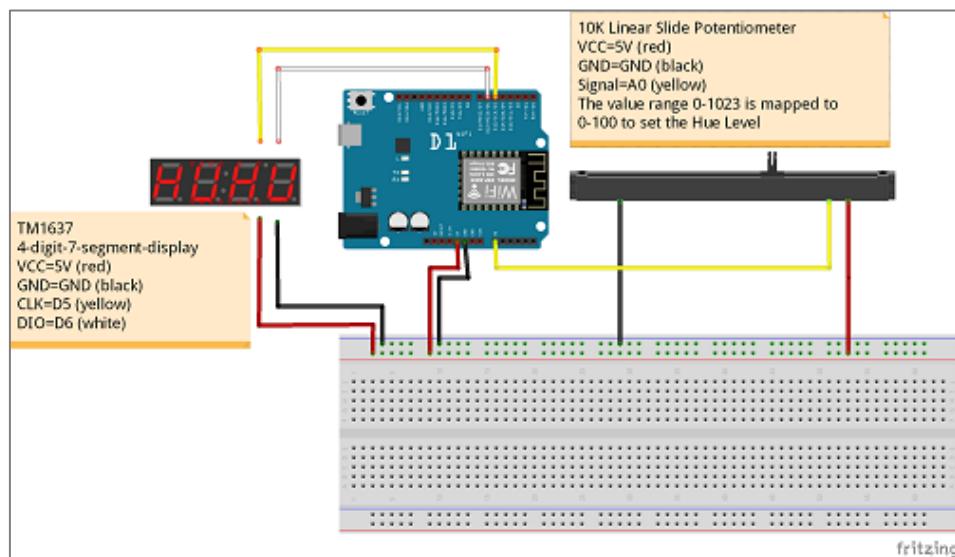
Wiring

Potentiometer=ESP8266 WeMOS D1
 VCC=5V (red)
 Signal=A0 (yellow)
 GND=GND (black)

4-digit-7-segment-display=ESP8266 WeMOS D1
 VCC=5V (red)
 CLK=SLK/D5 (yellow)
 DIO=MISO/D6 (white)
 GND=GND (black)

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 Entry when 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 microcontrollers (like the NodeMCU, WeMOS and many more), running [ESP Easy](#) with associated sensors, sending data from and to Domoticz.

If not familiar with ESP Easy, start with the previous link shared.

Further more learning [rules](#) and [commands](#) are needed.

Hard-/Software used:

Hardware: ESP8266 NodeMCU DevKit.

Software: ESP Easy Beta Mega Release ESPEasy_mega-20190227.zip (see GitHub [releases](#)).

Note

The software has been updated regulary.

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
- *More to follow*

Flash ESP

Download

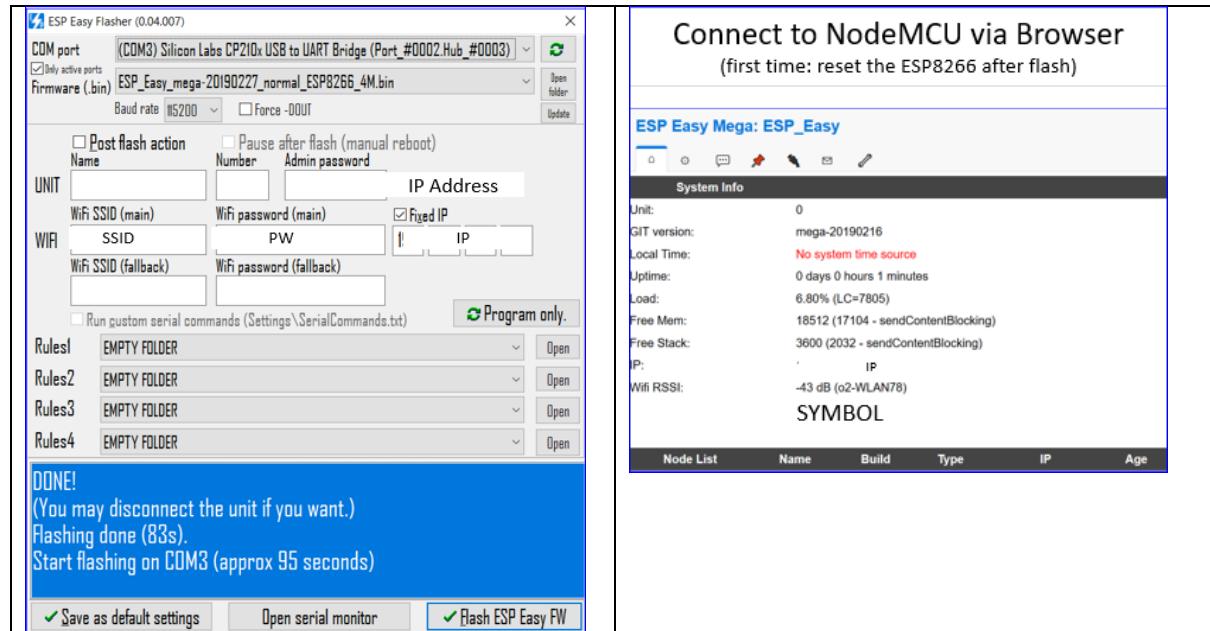
Downloaded ESPEasy_mega-20190227.zip from GitHub and extracted to a temp folder, i.e. domoticz-home-automation-workbook\functions\ESPEasy\r200\.

There are daily builds available – checkout the change logs.

Flashing

From the temp folder, start *ESP.Easy.Flasher.exe* and configure as below.

Used the firmware with 4M for an ESP8266 – the NodeMCU can handle 4M.



After pressing “Flash ESP Easy FW” it took about 2 minutes to complete.

Update ESP

Update the ESP Easy Firmware using the ESP Easy Web Frontend.

- Download the latest version of the firmware, i.e. https://github.com/letscontrolit/ESPEasy/releases/download/mega-20190315/ESPEasy_mega-20190315.zip
- Unzip to a temp folder, i.e. domoticz-home-automation-workbook\functions\ESPEasy\r200\.
- Open Browser
- Enter ESP Easy URL
- Select menu Tools > Update Firmware
- Select ESP_Easy_mega-20190315_normal_ESP8266_4M.bin
- Press Update
- Wait for completion which is followed by a reboot of the NodeMCU
- Refresh the Browser window (F5)
- Check menu Tools > System Info if version is correct updated

Firmware	
Build:	20103 - Mega
System Libraries:	ESP82xx Core 2_4_2, NONOS SDK 2.2.1(cfd48f3), LWIP: 2.0.3 PUYA support
Git Build:	mega-20190315
Plugins:	46 [Normal]
Build Md5:	825459376171e63f211c028ecbf1f69
Md5 check:	passed.
Build Time:	Mar 15 2019 03:15:17
Binary Filename:	ESP_Easy_mega-20190315_normal_ESP8266_4M.bin

Domoticz Controller

Communication between ESP Easy and Domoticz is possible via Controllers with the protocol's HTTP and MQTT, configured in ESP Easy using the Domoticz Server IP address. For other settings, the defaults are used.

ESP Easy Mega: ESP_Easy <input type="radio"/> Main <input type="radio"/> Config <input checked="" type="radio"/> Controllers <input type="radio"/> Hardware	
Controller Settings	
Protocol:	Domoticz HTTP
Locate Controller:	Use IP address
Controller IP:	IP Address
Controller Port:	8080
Minimum Send Interval:	100 [ms]
Max Queue Depth:	10
Max Retries:	10
Full Queue Action:	Ignore New
Check Reply:	Ignore Acknowledgement
Client Timeout:	1000 [ms]
Controller User:	
Controller Password:	
Enabled:	<input checked="" type="checkbox"/>
Protocol:	Domoticz MQTT
Locate Controller:	Use IP address
Controller IP:	IP Address
Controller Port:	1883
Minimum Send Interval:	100 [ms]
Max Queue Depth:	10
Max Retries:	10
Full Queue Action:	Ignore New
Check Reply:	Ignore Acknowledgement
Client Timeout:	1000 [ms]
Controller User:	
Controller Password:	
Controller Subscribe:	domoticz/out
Controller Publish:	domoticz/in

Rules

ESP Easy Rules can be used to create simple flows to control devices on your ESP.
Read more [here](#).

Example Rules Set #1

```
// 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
```

Rules are logged (Menu Tools > System Log)

```
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
```

This example shows setting the LED, connected to GPIO14, to on (= state changes from 0 LOW to 1 HIGH) because the Lux value went below threshold of 100.

Device Type (Light/Lux – BH1750), Name (ESPEasy_BH1750) and Value (Lux)

Light/Lux - BH1750	ESPEasy_BH1750	2 (46)	GPIO-4 GPIO-5	Lux:	183.33
Values					
#	Name				
1	Lux				

GPIO Commands

It is possible using URL commands, to control basic GPIO output.

This could be used by Domoticz event to trigger a LED or else i.e. controlling a server motor.

Turn an LED ON/OFF

ON

[http://espeasy-ip-address/control?cmd= gpio,14,1](http://espeasy-ip-address/control?cmd=gpio,14,1)

Result JSON Response

```
{"log": "GPIO 14 Set to 1", "plugin": 1, "pin": 14, "mode": "output", "state": 0}
```

OFF

[http://espeasy-ip-address/control?cmd= gpio,14,0](http://espeasy-ip-address/control?cmd=gpio,14,0)

Result JSON Response

```
{"log": "GPIO 14 Set to 0", "plugin": 1, "pin": 14, "mode": "output", "state": 1}
```

Experiments

To explore connecting sensors to an ESP controller running ESP Easy and test communication between the ESP and Domoticz v.v.

The ESP controller used, is a NodeMCU v1.0.

Domoticz switching ESP LED

Purpose

To switch, via a Domoticz Switch, the LED connected to pin D5 (GPIO14) of the NodeMCU.

Parts

1x NodeMCU

1x LED

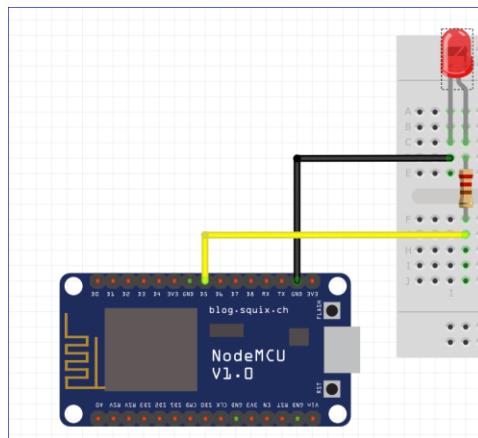
1x Breadboard

2x Wires male-male

Wiring

LED	NodeMCU [wirecolor]
Anode = PLUS connected to a resistor 220 Ohm to pin	D5 (GPIO14) [yellow]
Cathode = MINUS	GND [black]

Circuit



ESP Easy

In ESP Easy there are no devices required as the LED is controlled via HTTP API commands submitted by Domoticz.

Domoticz

Create a switch device (Hardware Virtual Sensor) with properties:

Switch Type	HTTP API Commands to switch On (1) / Off (0)
--------------------	---

On Action	http://espeasy-ip-address/control?cmd=gpio,14,1
Off Action	http://espeasy-ip-address/control?cmd=gpio,14,0

After saving, click the switch On and check if the LED connected to GPIO14 of the ESP device is turned on.

ESP Ambient Light to Domoticz

Purpose

To measure the ambient light from the sensor BH1750FVI connected to the NodeMCU and send the Lux value to a Domoticz Lux device.

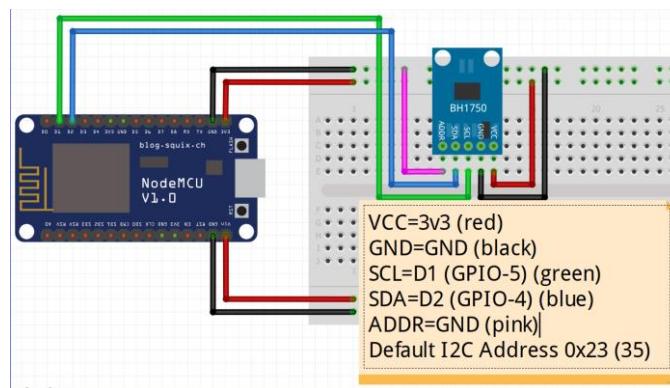
Parts

- 1x NodeMCU
- 1x BH1750FVI sensor
- 1x Breadboard
- 7x Wires male-male

Wiring

LED	NodeMCU [wirecolor]
SCL	D1 (GPIO5) [green]
SDA	D2 (GPIO4) [blue]
ADDR	GND [pink]. Default I2C address used: 0x23 (35)
VCC	3v3 [red]
GND	GND [black]

Circuit



ESP Easy

Hardware	<p><i>BH1750 = ESP8266 NodeMCU</i></p> <p>VCC = 3v3</p> <p>GND = GND</p> <p>SCL = D1 (GPIO5)</p> <p>SDA = D2 (GPIO4)</p> <p>ADDR = GND</p> <p>Default I2C address 0x23 (35)</p>
Devices	

Domoticz

Create a Lux device (Hardware Virtual Sensor) with name ESPEasy Ambient Light.

	Idx	Hardware	ID	Unit	Name	Type	SubType	Data
	46	VirtualSensors	82045	1	ESPEasy Ambient Light	Lux	Lux	197 Lux

The idx is used in ESP Easy device ESPEasy_BH1750 to set the property IDX for controller 2.

Check the Domoticz Log, if data from the sensor is received:

```
2019-04-26 18:49:27.040 MQTT: Topic: domoticz/in, Message: {"idx":46, "RSSI":10, "nvalue":0, "svalue":"210.00"}
```

Enhancement

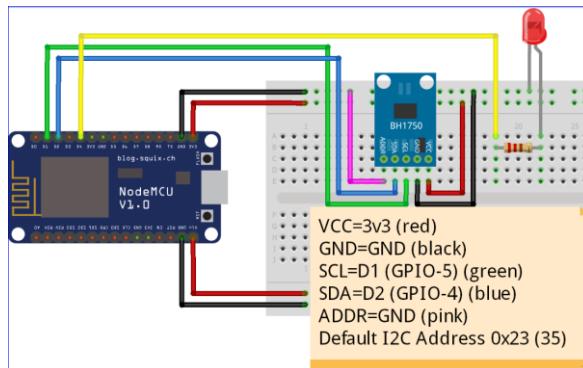
Purpose

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 NodeMCU.

There is no interaction with Domoticz, its just to understand using variables with Generic – Dummy Device.

Circuit



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
Edit	6	✓	Generic - Dummy Device	AmbientLight_Settings			Threshold: 100 State: 0 ...

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.

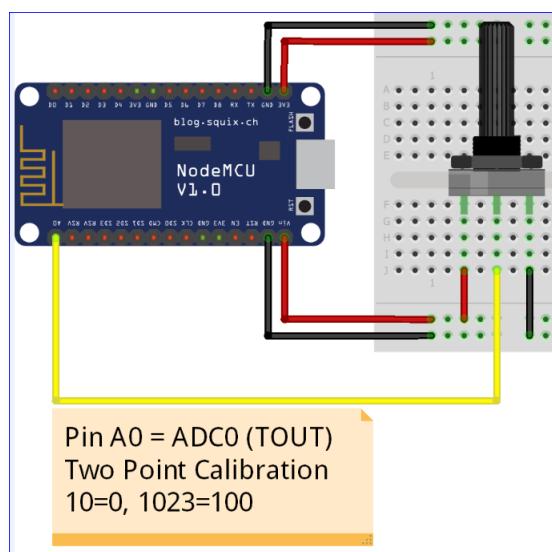
Parts

- 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 configuration interface for an Analog input device. The device is named "HUE_Dimmer". It is enabled and has Two Point Calibration enabled. The current value is 9 ± 0.099. The minimum and maximum values are 0 ± 0.987 and 1023 ± 100.000 respectively. The step size is 1 ± 0.099. The data acquisition interval is set to 2 seconds. There is one value entry for "Analog" with 0 decimals.

ESP Easy Device Configuration – Dummy Devices

The screenshot shows the configuration interface for a Generic - Dummy Device named "Varstore". It is enabled and simulates SENSOR_TYPE_SINGLE data type. The data acquisition interval is set to 60 seconds. There are two value entries: "HUE_Dimmer" and "HUE_Dimmer_Delta", both with 0 decimals.

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 server to set brightness level
    if [Varstore#HUE_Dimmer_Delta]>2

        SendToHTTP rpi-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 can not 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.

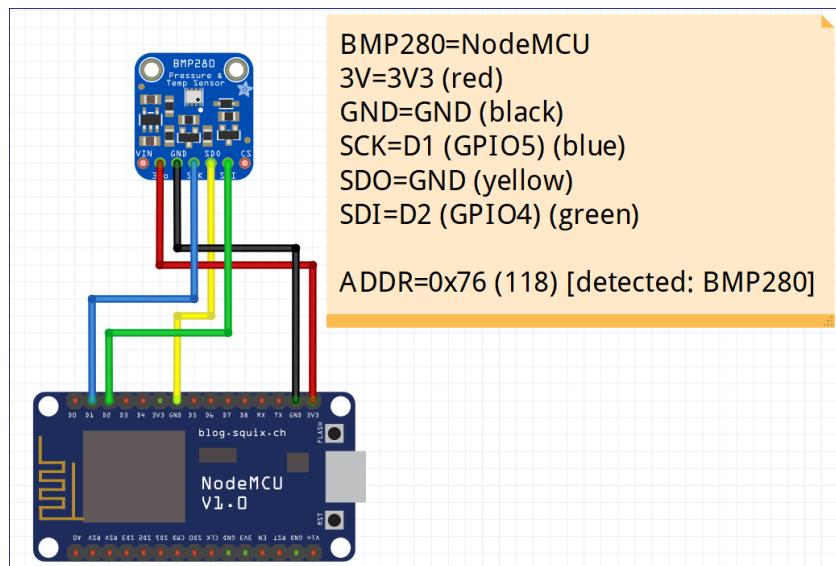
Parts

- 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

#	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.

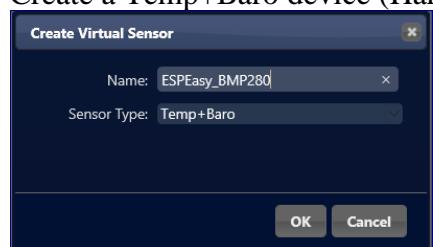
After adding check the if the values of the device are updating.

Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
Edit	1 ✓	Environment - BMx280	ESPEasy_BMP280			GPIO-4 GPIO-5	Temperature: 21.81 Humidity: 0.00 Pressure: 1017.75

The data is OK. Note that Humidity is 0 as not supported by this device BMP280.
To measure also 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:

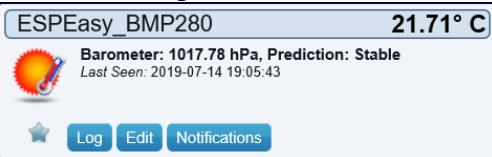
```
2019-07-14 19:00:47.890 MQTT: Topic: domoticz/in, Message:  
{"idx":60,"RSSI":9,"nvalue":0,"svalue":"21.71;1017.74;0;0"}
```

Note

The svalue contains:

Temperature;Barometric pressure;Barometer forecast;Altitude (not used, set to 0)

The widgets.

GUI Tab Temperature	GUI Tab Weather
 <p>ESPEasy_BMP280 21.71°C Barometer: 1017.78 hPa, Prediction: Stable Last Seen: 2019-07-14 19:05:43 ★ Log Edit Notifications</p>	 <p>ESPEasy_BMP280 1017.78 hPa Prediction: Stable Last Seen: 2019-07-14 19:06:39 ★ Log Edit Notifications</p>

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.

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.
```

MQTT Hints

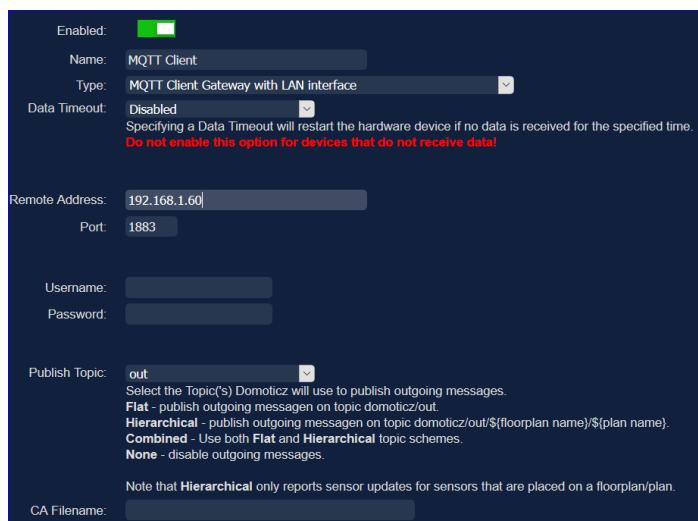
Add the host to the mosquitto command:

```
-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	rpi-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 rpi-domoticz-ip:1883
2018-08-27 12:21:12.228 Status: MQTT: connected to: rpi-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's 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 Domoticz Log entry (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

Run from a terminal

```
mkdir /home/pi/python  
git clone https://github.com/eclipse/paho.mqtt.python.git  
cd paho.mqtt.python  
sudo python setup.py install
```

Output

```
pi@DomoticzDev:~ $ mkdir python  
pi@DomoticzDev:~ $ 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.  
pi@DomoticzDev:~ $ cd paho.mqtt.python  
pi@DomoticzDev:~/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)

Run from a terminal with sample output

Command

```
pip install paho-mqtt
```

Output

```
Collecting paho-mqtt  
  Downloading  
    https://files.pythonhosted.org/packages/2a/5f/cf14b8f9f8ed1891cda893a2a7d1d6fa23de2a9fb4832f05cef02b79d  
      01f/paho-mqtt-1.3.1.tar.gz (80kB)  
        100% |81kB 989kB/s  
Building wheels for collected packages: paho-mqtt  
  Running setup.py bdist_wheel for paho-mqtt ... done  
  Stored in directory:  
/home/pi/.cache/pip/wheels/38/ca/67/86c7e4acc659ce5ab74cbb8cc38de50c90ed4f827133e36994  
Successfully built paho-mqtt  
Installing collected packages: paho-mqtt  
Successfully installed paho-mqtt-1.3.1
```

Command

```
pip3 install paho-mqtt
```

Output

```
Collecting paho-mqtt
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-1.3.1-py3-none-any.whl (57kB)
    100% |61kB 437kB/s
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.3.1
```

Subscribe Sample Script

This Python sample script connects to the MQTT broker running on the Raspberry Pi, subscribes to the Domoticz Out topic ('domoticz/out') and filters out the value for the device with idx 1, which is the Raspberry Pi motherboard CPU usage in percentage.

	Idx	Hardware	ID	Unit	Name	Type	SubType	
<input type="checkbox"/>	1	Motherboard	0000044D	1	CPU_Usage	General	Percentage	0.18%

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

import time
import paho.mqtt.client as mqtt
import json

broker= "localhost"
topic = "domoticz/out"
idx = 1

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)

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 mqtttest.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" : 1,
    "name" : "CPU_Usage",
    "nvalue" : 0,
    "stype" : "Percentage",
    "svalue1" : "0.77",
    "unit" : 1
}

message topic= domoticz/out
message qos= 0
message retain flag= 0
1 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
```

Publish Sample Script

This sample script connects to the MQTT broker running on the Raspberry Pi and publishes text to a Virtual Device with idx 9, name Biotonne Pi, Type General, SubType Text

9	Virtual Devices	00082009	1	Biotonne Pi	General	Text	Moin Moin
---	-----------------	----------	---	-------------	---------	------	-----------

The script sets the text from “Moin Moin” to “Hello Again”.

Python3 Script

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

import time
import paho.mqtt.client as mqtt

version="MQTT Test v20180827"
broker= "localhost"
topic = "domoticz/in"
idx = 9
payload='{"command": "udevice", "idx": %s, "svalue": "Hello Again"}' % (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)

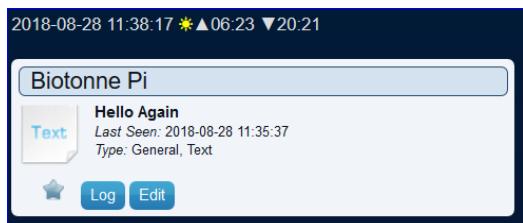
def on_log(client, userdata, level, buf):
    print("log: ",buf)

print(version)
print("Creating new instance")
client = mqtt.Client("P1")
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)
print("Publishing message to topic",topic, "with payload",payload)
domoticz/in, Message: {"command": "udevice", "idx": 9, "svalue": "Hello Again"}
client.publish(topic,payload)
time.sleep(4)
client.loop_stop()
```

Output

```
pi@DomoticzDev:~/python $ python3 mqtt-pub-test.py
MQTT Test v20180827
Creating new instance
Connecting to the broker localhost
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
Publishing message to topic domoticz/in with payload {"command": "udevice", "idx": 9, "svalue": "Hello Again"}
log: Sending PUBLISH (d0, q0, r0, m1), 'b'domoticz/in'', ... (57 bytes)
```

Widget with updated text



Node-RED

Purpose

To explore Node-RED functionality for this Home Automation Solution:

- 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 process

[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.

Installation

Desktop > Preferences > Recommended Software > All Programms > Click on Node-RED.
Press OK.

The package will be installed.

After installation, a new Desktop Menu Entry is available:
Programming > Node-RED.

To enable autostart of Node-RED at every boot, run 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://rpi-domoticz-ip:1880>).

Use disable, to disable autostart:

```
sudo systemctl disable nodered.service
```

Update

<TODO> Check if it is possible to update via Desktop Check Updates.

Important

Prior updating Node-RED, backup the flows file, located in /home/pi/.node-red/

The flows have, depending on the Raspberry Pi device, following name:

flows_DomoticzDev.json

or

flows_DoPro.json

After updating, reboot the Raspberry Pi.

```
cd /home/pi/.node-red
update-nodejs-and-nodered

or if not working

bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-
package/master/resources/update-nodejs-and-nodered)
```

Note

on using command *update-nodejs-and-nodered*:

If this message occurs, then use the bash command:

This Node-RED install doesn't support using apt. Please see the online docs at <https://nodered.org/docs/hardware/raspberryPi> for the best way to update and upgrade.

Update Log Example

```
This script will remove versions of Node.js prior to version 6.x, and Node-RED and
if necessary replace them with Node.js 8.x LTS (carbon) 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 really sure you want to do this ? [y/N] ? y
Running Node-RED update for user pi at /home/pi
This can take 20-30 minutes on the slower Pi versions - please wait.
```

Stop Node-RED	✓
Remove old version of Node-RED	✓
Remove old version of Node.js	-
Update Node.js LTS	✓ Node v8.11.4 Npm 6.4.1
Clean npm cache	✓
Install Node-RED core	✓ 0.19.3
Move global nodes to local	-
Install extra Pi nodes	-
Npm rebuild existing nodes	✓
Add menu shortcut	✓
Update systemd script id=Node-RED.desktop	✓

```
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-address}:1880
```

```
Started Thu 6 Sep 15:10:56 CEST 2018 - Finished Thu 6 Sep 15:13:05 CEST 2018
```

Access Flows & Dashboard UI

To access Node-RED, point a browser at
`http://localhost:1880` (when using a Raspberry Pi with a connected monitor)
or
`http://rpi-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:

```
http://rpi-domoticz-ip:1880/ui
```

Start, Stop, Log

Task	Terminal Command
Node-RED Start	<code>node-red-start</code>
Node-RED Stop	<code>node-red-stop</code>
Node-RED view recent log output	<code>node-red-log</code>

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

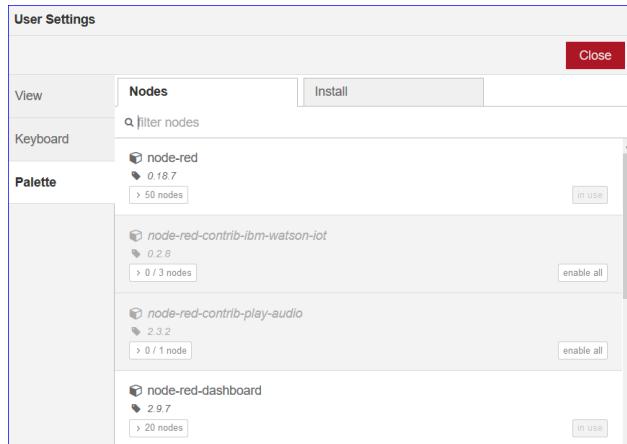
Important

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_DomoticzDev.json
Additional Node Packages	/home/pi/.node-red/node_modules/<package name>
Additional Node Packages Sample Dashboard	/home/pi/.node-red/node_modules/node-red-dashboard

Flow Domoticz MQTT Messages

A simple example flow, to test Node-RED.

Subscribe to MQTT topic “domoticz/out” to listen to all Domoticz MQTT messages.

Node-RED Flow



The MQTT Broker is localhost with default port 1883.

The JSON node is used to convert JSON message to JavaScript message.

Node-RED Flow Source

```
[{"id": "45e7c703.b9b9f", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 340, "wires": [{"id": "cc4fd39c.684108"}]}, {"id": "cc4fd39c.684108", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 340, "wires": [{"id": "552cdad.37816a4"}]}, {"id": "552cdad.37816a4", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG idx=ALL", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 810, "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": ""}]
```

Output debug

	<p>With a payload example (Hue Light with idx=118)</p> <pre>{ "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} }</pre>
<p>To access a property, convert the Domoticz message to a JSON object, then get the property, like msg.payload.idx or msg.payload.Level</p>	

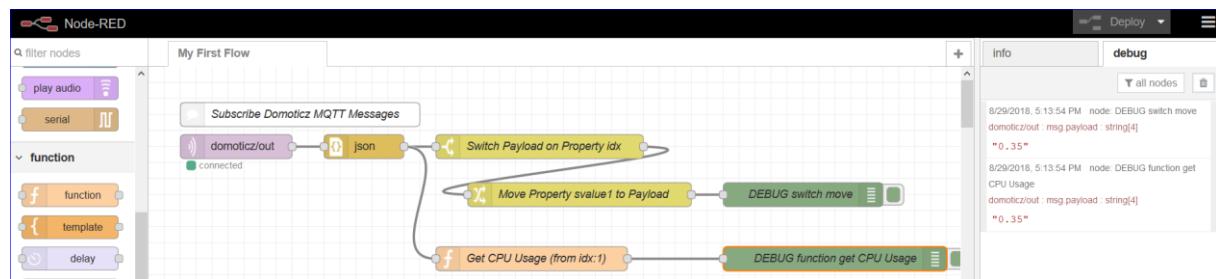
Flow Raspberry Pi CPU Usage

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}
```

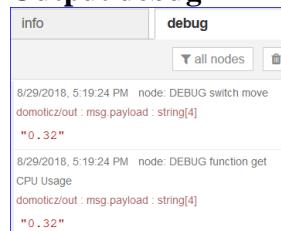
Node-RED Flow



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\nnidx = msg.payload.idx;\nncpu_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\nmsg;\n"}, {"id": "912289d2.e8c1", "type": "change", "z": "76cb1798.6a823", "name": "Move Property svalue1 to Payload", "rules": [{"t": "move", "p": "payload.svalue1", "pt": "msg", "to": "payload", "tot": "msg"}], "action": "", "property": "", "from": "", "to": "", "reg": false, "x": 540, "y": 160, "wires": [{"35214a2f.f89656"}]}, {"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



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.

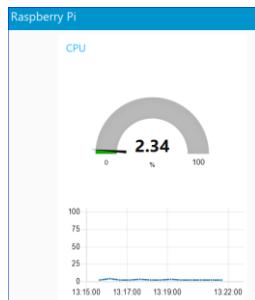
Dashboard UI

The Node-RED [Dashboard](#) module offers UI nodes to build a Dashboard.

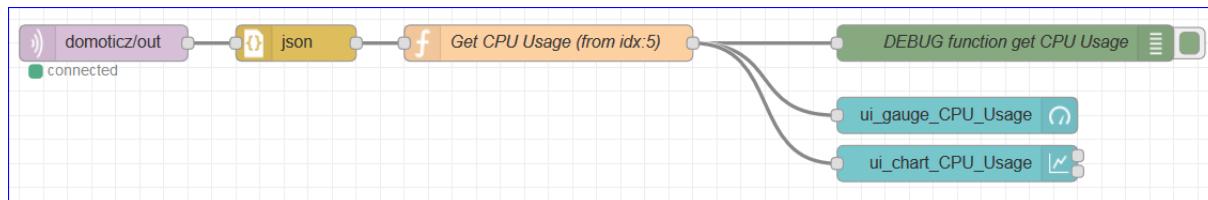
Flow Monitor CPU Usage

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": [[[ "db5fff92.295628"]]]}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 240, "wires": [[[ "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// Check 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": [[[ "c030082a.286338", "61fef7cc.4fc02", "d5bc4ce8.ae1f28"]]]}, {"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": "61fef7cc.4fc02", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "gtype": "gage", "title": "", "label": "%", "format": "{{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", "#ff9999", "#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"}]
```

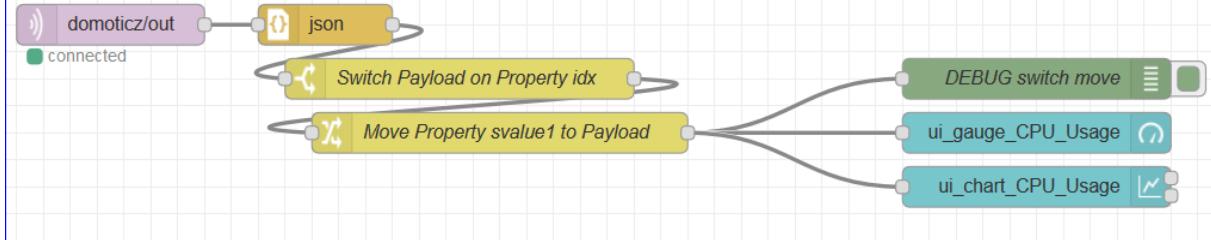
Note

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": [[[ "db5fff92.295628"]]}, {"id": "35214a2f.f89656", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG switch move", "x": 270, "y": 100, "wires": []}, {"id": "5f7a4a09.abb57c", "type": "switch", "z": "76cb1798.6a823", "name": "Switch Payload on Property idx", "x": 270, "y": 140, "wires": [{"x": 270, "y": 140, "x2": 370, "y2": 140}], {"id": "912289d2.e8c1", "type": "change", "z": "76cb1798.6a823", "name": "Move Property svalue1 to Payload", "x": 370, "y": 140, "wires": [{"x": 370, "y": 140, "x2": 400, "y2": 180}], {"id": "35214a2f.f89656", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "x": 400, "y": 180, "wires": [{"x": 400, "y": 180, "x2": 450, "y2": 200}], {"id": "c3b6865e.2b93e", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "x": 450, "y": 180, "wires": [{"x": 450, "y": 180, "x2": 500, "y2": 200}], {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "x": 500, "y": 100, "wires": [{"x": 500, "y": 100, "x2": 550, "y2": 120}]}]
```

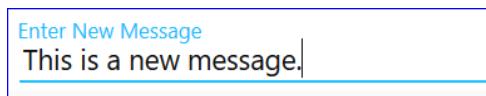
Flow Virtual Sensor Text Update

Update the Virtual Sensor “Gartenhaus Sensor Info” with new text.
The device has idx 52, Type General, Text.

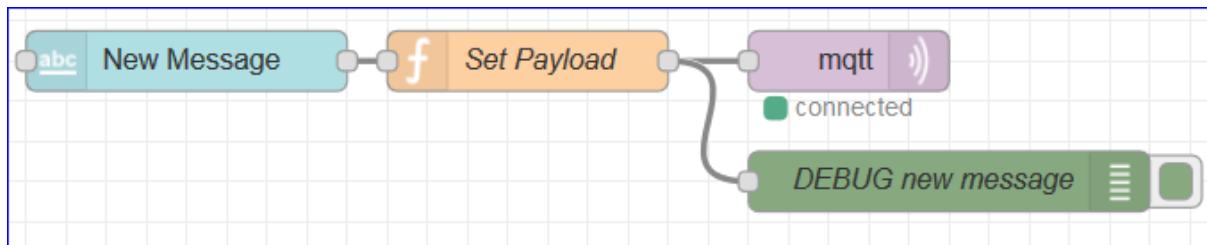
Widget showing the updated text



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\n// Get the text from the incoming payload\nvar msg = msg.payload;
\n\n// Build the new payload\nmsginfo.payload = {
  \"idx\": 52,
  \"nvalue\": 0,
  \"svalue\": text
};\nnode.warn(msginfo.payload);
\n\n// Return the message\nreturn msginfo; ", "outputs": 1, "noerr": 0, "x": 290, "y": 820, "wires": [{"id": "d4be5444.33e23", "x2": 450}], "label": "DEBUG new message", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "true", "x": 500, "y": 880}, {"id": "27ca36a9.4c3cba", "type": "ui_group", "z": "", "name": "Hue", "tab": "8a1743ab.9d931", "display": 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"}]
```

Python 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.

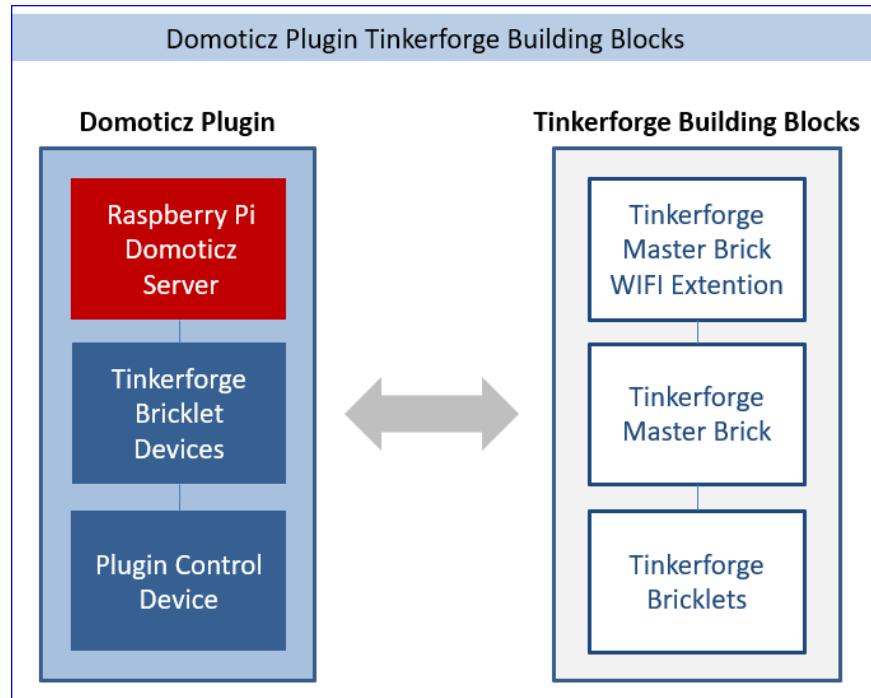
Mandatory to consult this [manual](#) as reference.

The next chapter describes in depth the build of a sample plugin interacting between Domoticz and Tinkerforge.

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) shown in 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 can not 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'.
```

TabError

Ensure to set the tab indents right, else error message in the Domoticz log during start: "TabError".

Example Domoticz Log Error Message

```
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

Log entry

```
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/pl
at-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

Log Entry

```
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="")
```

Plugin Traffic Light (Tinkerforge Building Blocks)

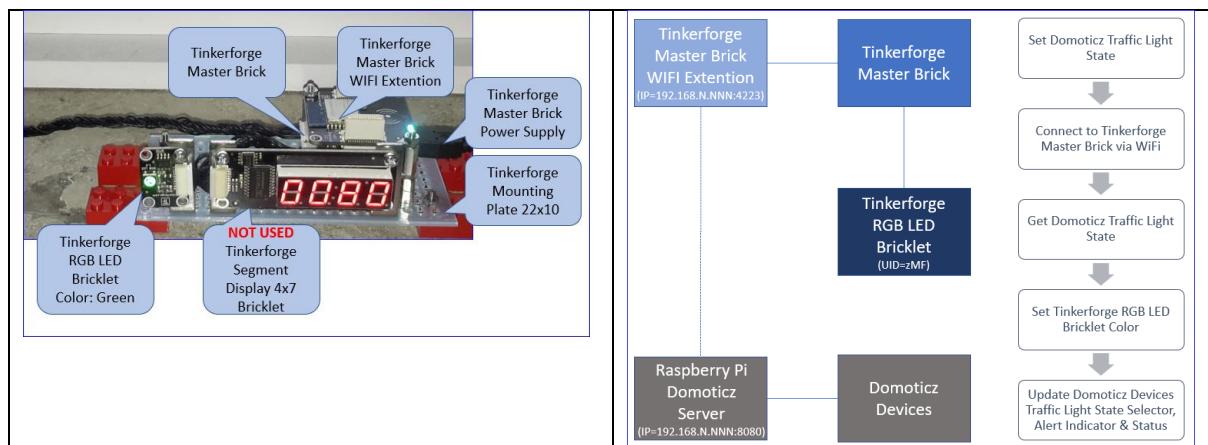
Sample building a Domoticz Plugin using Tinkerforge Building Blocks.

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 Home Automation 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 Raspian 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-address
```

Domoticz Plugin Folder & File

The plugin **plugin.py** is installed in a subfolder of the Domoticz Plugin folder on the Domoticz Server.

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 Server, 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 server
- Putty session connected to the Domoticz server

The web 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':'rpi-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 Entry 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" : "",
"stype" : "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](#) functionality 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 Meter, Radiator Thermostat, Window/Door Contact.
- For exploring & testing only homematicIP devices are used.

Notes

As taken from [here](#): The RaspberryMatic project is a collaborative 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).

Installation

RaspberryMatic

The Raspberry Pi 3B+ has been setup according [these](#) guidelines.

RaspberryMatic version 3.45.7.20190622 is used (check out for newer versions).
Several Addons installed:

Additional software for RaspMatic			
System-Update	Installed version: 1.13.12 Available version: 1.13.12 Download Uninstall Set	rmupdate addon https://github.com/j-a-n/raspberrymatic-addon-rmupdate	
Mosquitto	Installed version: 1.5.8+2 Available version: 1.5.8+2 Download Restart Uninstall	 mosquitto	
XML-API	Installed version: 1.20 Available version: 1.20 Download Uninstall Set	XML-API CCU Addon https://github.com/hobbyquaker/XML-API	
CUx-Daemon	Installed version: 2.3.2 Available version: 2.3.2 Download Restart Uninstall Set	 CUx-Daemon 2.3.2	
Install / update additional software	Select additional software: <input type="text"/> Browse... Install	Please note: Additional software installed by the user can lead to unexpected results, data loss or even system instability. eQ-3 AG does not assume any liability for additional software installed by the user. To finish installation, the CCU will be restarted automatically.	

Note

<TODO>The Mosquitto addon installed but not explored (yet).

Addon XML-API

Information

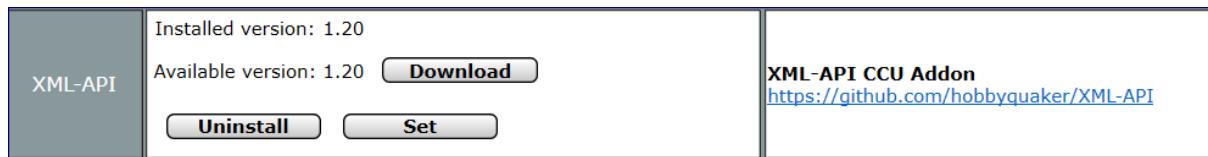
The [XML-API](#) CCU Addon is used to communicate between the CCU and Domoticz.

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).

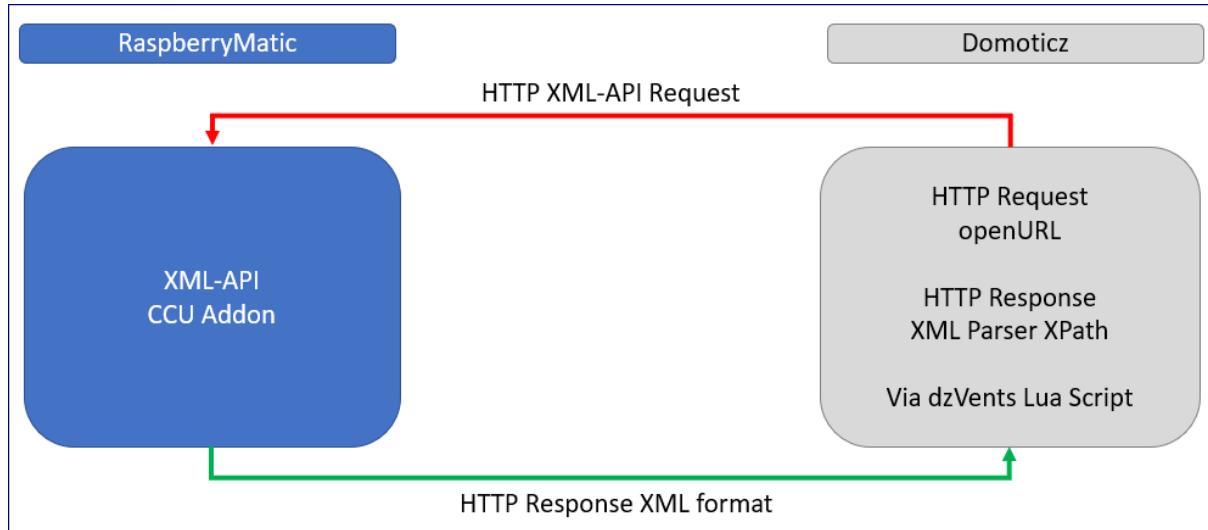
These are used by the automation scripts with relevant parameters.

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-address/config/xmlapi/statelist.cgi>
- List channels and values for single or multiple devices (1234,5678): state.cgi
Example:
http://ccu-ip-address/config/xmlapi/state.cgi?device_id=1541.
- Changes one or more channel states: statechange.cgi
Example:
http://ccu-ip-address/config/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 id's are used. These can be obtained via the statelist.cgi script.

HTTP XML-API Request

```
http://192.168.1.225/config/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”.

The screenshot shows the HomeMatic WebUI interface. At the top, a header reads "RaspberryMatic – General Device Settings with Name change". Below this, a table lists devices. A blue arrow points from the original device entry to its settings dialog, which is shown twice side-by-side: once with the original name and once with the new name "Radiator Thermostat MakeLab".

HmIP-eTRV-2 000A18A9A64DAC	HmIP-eTRV-2		Radiomatic IP Radiator Thermostat	000A18A9A64DAC	HmIP-RF	Secured	Heating	MakeLab	2.8 V -68 dBm -80 dBm
Radiator Thermostat MakeLab	HmIP-eTRV-2		Radiomatic IP Radiator Thermostat	000A18A9A64DAC	HmIP-RF	Secured	Heating	MakeLab	2.8 V -68 dBm -80 dBm

General device settings: 000A18A9A64DAC

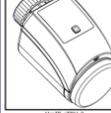


Name: HmIP-eTRV-2 000A18A9A64DAC
Type description: HmIP-eTRV-2
Serial number: 000A18A9A64DAC
Operable:
Visible:
Logged:
Service messages:

Functional test

...:::...
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. Persons or animals must not be near the device while the test is run, as they are operated manually. The test is passed as soon as the first feedback will be received by the device.

General device settings: 000A18A9A64DAC



Name: Radiator Thermostat MakeLab
Type description: HmIP-eTRV-2
Serial number: 000A18A9A64DAC
Operable:
Visible:
Logged:
Service messages:

Functional 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. Persons or animals must not be near the device while the test is run, as they are operated manually. The test is passed as soon as the first feedback will be received by the device.

```
<?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.LOW_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”.

Example Device Request HmIP-eTRV-2

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-address/config/xmlapi/state.cgi?device_id=1541.
2. The CCU sends a HTTP response, in XML format, to Domoticz server.
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 web-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-address/config/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://192.168.1.225/config/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
}
```

Example Datapoint Request HmIP-eTRV-2

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-address/config/xmlapi/state.cgi?datapoint_id=1567.
2. The CCU sends a HTTP response, in XML format, to Domoticz server.
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
1	65	VirtualDevices	14091	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.4 C

HTTP XML-API Request Test

Test in a web-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-address/config/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-address/config/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.

Example Datapoint Request HmIP-SWDO

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 (<http://ccu-ip-address/config/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 web-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-address/config/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://192.168.1.225/config/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);
        -- 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 Entries

```

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

```

Example Datapoint Change Value HmIP-eTRV-2

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

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	▼ ▲ ▼ ▲
10	18	▼ ▲ ▼ ▲
20	19	▼ ▲ ▼ ▲
30	20	▼ ▲ ▼ ▲
40	21	▼ ▲ ▼ ▲

Level name: Add

Level Action

0	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=0	▼ ▲
10	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=18	▼ ▲
20	http://CCU-IP i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=19	▼ ▲
30	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=20	▼ ▲
40	http://i/config/xmlapi/statechange.cgi?ise_id=1584&new_value=21	▼ ▲

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 web-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-address/config/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-address/config/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://192.168.1.225/config/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://192.168.1.225/config/xmlapi/statechange.cgi?ise_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

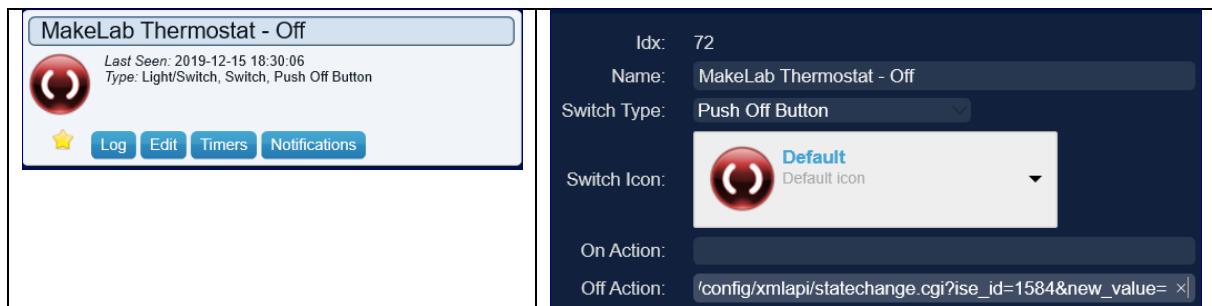
The state change action can also be assigned to a Switch to turn the thermostat Off.

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-address/config/xmlapi/statechange.cgi?ise_id=1584&new_value=0`



Domoticz Log Example

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)
```

Example Update Domoticz Device Value HmIP-eTRV-2

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 server 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 a 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-
address: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 are navigation links: Admin, Home page, Status and control, Programs and connections, Programs, Programming, Home page, Status and control, Programs and connections, Settings. On the right, there are buttons for Alarm messages (0), Logout, Service messages (1), Teach-in devices, and Help.

The main area displays a table for a script named "HmIP-eTRV-2-MakeLab". The table has columns for Name, Description, Condition (if...), Activity (then..., or else...), and Action.

Name	Description	Condition (if...)	Activity (then..., or else...)	Action
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	<input type="checkbox"/> intrinsic

Below the table, there is a "Condition: If..." section with a dropdown menu set to "Device selection" and a condition: "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 "AND" and "OR" dropdowns.

Under the condition, there is a "Activity: Then..." section with a checkbox checked: "Stop all current delays before performing the activity (e.g. retriggering)". Below this, there is a "Script" dropdown set to "Function: Thermostat.MakeLab.Notify.Actual.Temperature.cha..." and a "Run mode" dropdown set to "immediately".

At the bottom, there is an "Activity: Else..." section with a checkbox checked: "Stop all current delays before performing the activity (e.g. retriggering)".

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://192.168.1.179:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=20.89
string amp = "&";
string urldom = "'http://192.168.1.179:8080/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 like
'http://192.168.1.179:8080/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

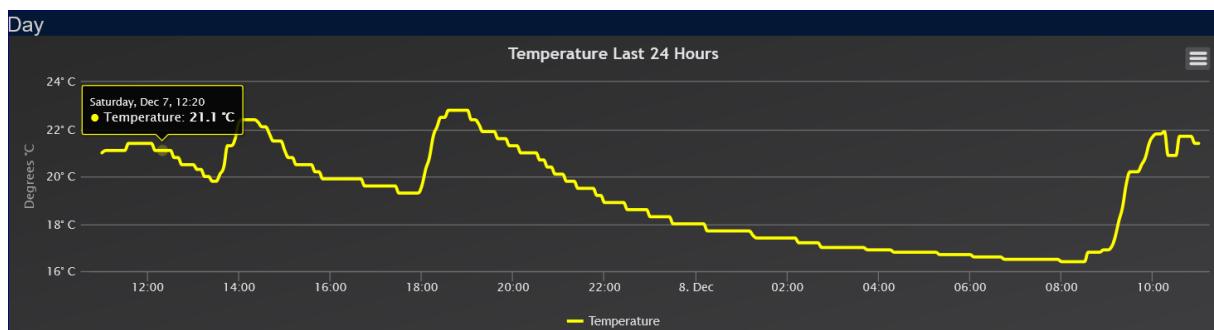
DomoticzLog	Domoticz Log Messages	03.12.2019 23:44:44	Thermostat MakeLab Temperature- Last update OK:08.12 11:14
-------------	-----------------------	---------------------	---

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



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



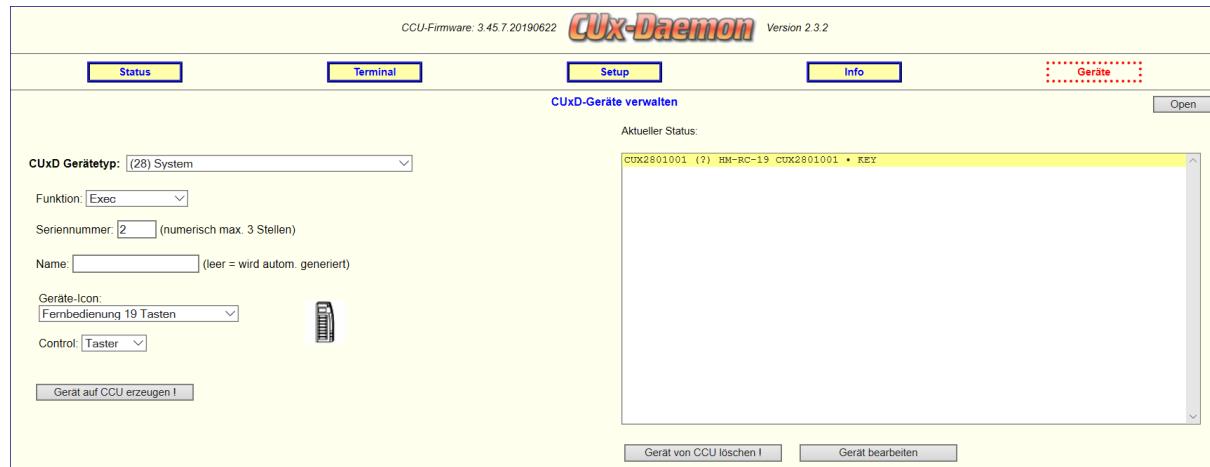
CUxD Set Menu



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.

Name	Picture
HM-RC-19 CUX2801001	
HM-RCV-50 BidCoS-RF	
HmIP-eTRV-2 000A18A9A64DAC	
HmIP-PSM 0001D3C99C6AB3	
HmIP-RCV-50 61A7D7098E047C	

Channel	Room	Function	Last modified	
HM-RC-19 CUX2801001:1	Filter	Filter	07.07.2019 08:58:05	Short button press Long button press
HM-RC-19 CUX2801001:2				

Check the device channels and datapoints with a XML-API HTTP Request:

```
http://ccu-ip-address/config/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="true" visible="true" 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="0" type="PRESS_LONG"/>
      <datapoint ise_id="1612" name="CUxD.CUX2801001:1.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="0" type="CMD_KILL"/>
      <datapoint ise_id="1611" name="CUxD.CUX2801001:1.CMD_EXEC" operations="2" timestamp="0" valueunit="" valuetype="20" value="0" 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="0" type="PRESS_SHORT"/>
      <datapoint ise_id="1653" name="CUxD.CUX2801001:2.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="0" type="PRESS_LONG"/>
      <datapoint ise_id="1638" name="CUxD.CUX2801001:2.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="0" type="CMD_KILL"/>
    </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
Condition: If...				
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).				
<input type="button" value="Script"/> string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.201..." <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-address: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-address: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/config/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-address/config/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/config/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 web browser, but also in the HomeMatic Web-UI.

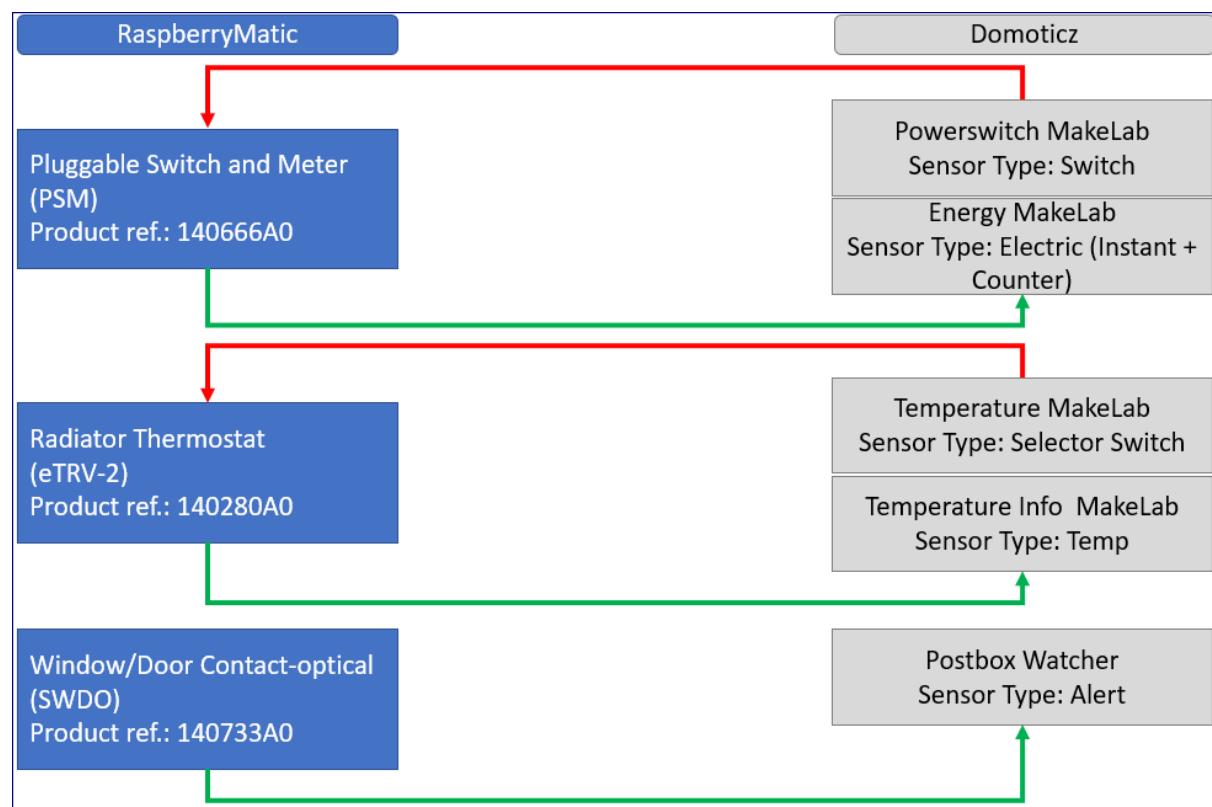
CCU Devices homematicIP

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"> • Room MakeLab • Switch main devices • Measure energy consumption (Wh) & power (W) 	Max. switching capacity 3680 W
Radiator Thermostat (eTRV-2) Product ref.: 140280A0 Device ID: 1541	<ul style="list-style-type: none"> • Room MakeLab • Control room temperature • Measure room temperature 	
Window / Door Contact – optical (SWDO) Product ref.: 140733A0 Device ID: 2530	<ul style="list-style-type: none"> • Postbox Watcher • Notify if the hatch of the postbox has been opened 	

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, Product ref.: 140666A0, 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.

HomeMatic Configuration					Domoticz Configuration (Name/Type/SubType)
Channel	Room	Function	Last modified	Control	
HMIP-PSM 0001D3C99C6AB3:5 Switch actuator			30.06.2019 09:10:57	<div style="display: flex; justify-content: space-around;"> OFF On </div>	Powerswitch MakeLab Light/switch Switch
HMIP-PSM 0001D3C99C6AB3:6 Status report measured value channel	Home office		30.06.2019 09:10:57	<div style="display: flex; justify-content: space-around;"> <div>Energy counter RaspMatic 716.60 Wh</div> <div>Reset</div> </div> <div style="display: flex; justify-content: space-around;"> <div>Energy counter device 0.00 Wh</div> <div>Voltage 230.60 V</div> <div>Current 0 mA</div> </div> <div style="display: flex; justify-content: space-around;"> <div>Power 0.00 W</div> <div>Frequency 50.01 Hz</div> </div>	Electric Usage MakeLab General kWh

Automation Script

Event name: powerswitch_makelab

```

local IDX_POWERSWITCH_MAKELAB = 177;
local ID_DATAPOINT_STATE = 1451
local URL_CCU = 'http://ccu-ip-address/config/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] Can not 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-address/config/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

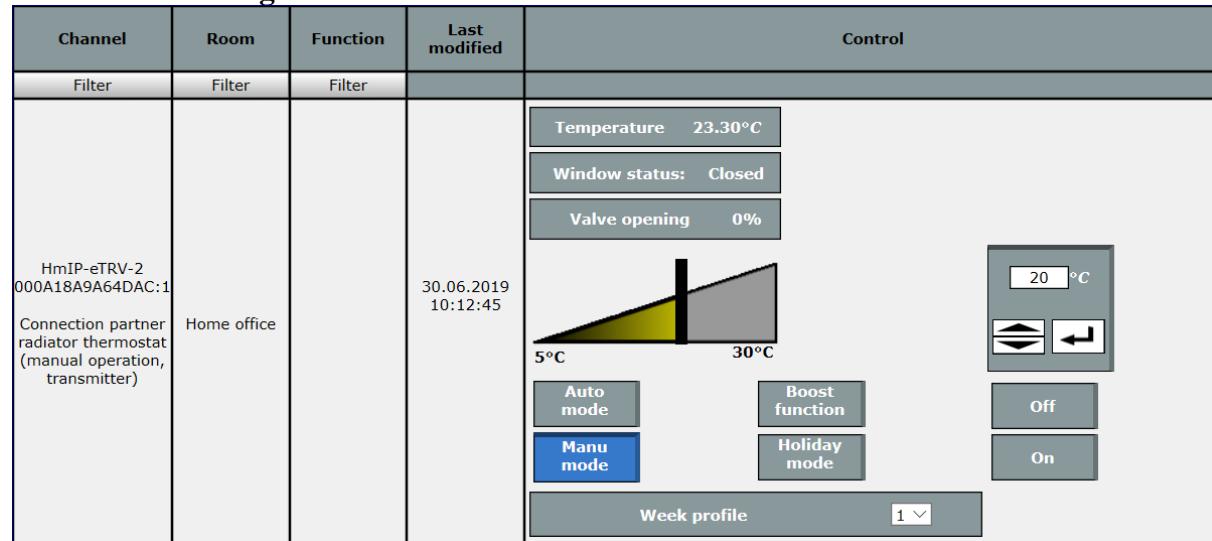
To measure and control the MakeLab room temperature.

A homematicIP Radiator Thermostat (eTRV-2, Product ref.: 140280A0, Device ID: 1541) connected to a RaspberryMatic operating system running a HomeMatic Central-Control-Unit (CCU).

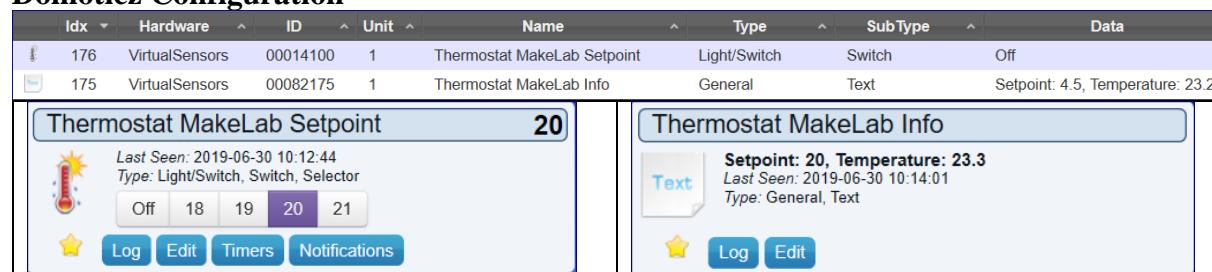
The thermostat setpoint is set by a Domoticz Selector Switch.

Domoticz requests, in regular intervals, the setpoint and room temperature from the CCU via a dzVents Lua script and updates Domoticz device information.

HomeMatic Configuration

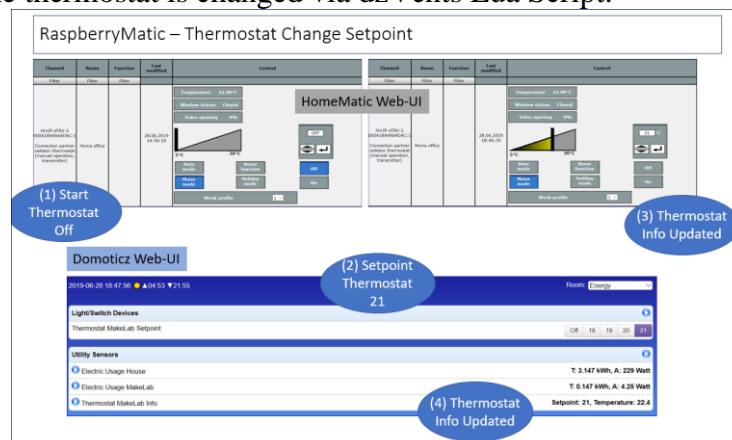


Domoticz Configuration



Change Thermostat Setpoint

The setpoint of the thermostat is changed via dzVents Lua Script.



Automation Script

Event name: thermostat_makelab_setpoint

```
local IDX_THERMOSTAT_MAKELAB_SETPOINT = 176;
local ID_DATAPOINT_SETPOINT = 1584;
local URL_RASPMATIC = 'http://ccu-ip-address/config/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
    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
}
```

SQL

Purpose

To execute CRUD (Create, Read, Update, Delete) operations on the Domoticz SQLite Database located in /home/pi/Domoticz/domoticz.db.

<TODO> Identify and build solution to manage entries, i.e. select or delete data esp. from the Measurement Tables.

Thoughts:

- Python3 scripts triggered by a Switch or Node-RED with Dashboard UI (and input field)
- Python3 with guizero – simple UI
- Node-RED SQLite3 Nodes
- SQLite3 package from a terminal
- B4J application

HINTS

- To analyse the database, external tools like the SQLiteSpy are useful helpers, but also using SQLite3 from the terminal command prompt.
- For tests, the Domoticz database has been exported from the Raspberry Pi to the Development Device (Domoticz GUI > Setup > Settings > Backup/Restore > Backup Database).
- For final solution, need to ensure Domoticz is stopped first, then perform CRIUD operations on the Domoticz database and start Domoticz afterwards.

SQL Command Examples

For testing of SQL statements on the Domoticz database, used

- the Domoticz database “Pinneberg.db” which has been exported via Setup > Settings > Backup/Restore > Backup Database
The database “Pinneberg.db” has been downloaded from the Raspberry Pi to the Downloads folder of the Development Device.
- the table tables [DeviceStatus] and [Temperature] of the Domoticz database “Pinneberg.db”.
- the tool SQLiteSpy.

Tables

The various tables holding measurement data are

```
'MultiMeter', 'MultiMeter_Calendar', 'Percentage', 'Percentage_Calendar', 'Rain', 'Rain_Calendar', 'Temperature', 'UV', 'UV_Calendar', 'Wind', 'Wind_Calendar'
```

SQL Create Statements for the tables [DeviceStatus] and [Temperature]

```
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 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')));
```

Note

Check if the table structure changes depending Domoticz version.

Select Device Status & Temperature

Select the device status and temperature data from the Domoticz database tables [DeviceStatus], [Temperature].

Python Script

Name: sql-select-test.py

```
#!/usr/bin/env python
import sqlite3
from sqlite3 import Error
import time
import os

version="SQL Test v20180828"
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 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 main():
    print(version)
    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection
        conn = create_connection(database)
        with conn:
            select_all_device_status(conn)
            select_all_temperatures(conn)
            select_temperature_by_threshold(conn, 41.0)

    conn.close()

if __name__ == '__main__':
    main()

```

Run

```
python3 sql-select-test.py
```

```

SQL Test v20180828
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Device Status
(1, 4, '0000044D', 1, 'CPU_Usage', 0, 243, 6, 0, 0, 12, 255, 0, '0.33', '2018-08-28 19:04:15', 1, 0.0,
1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(2, 4, '1', 1, 'Internal Temperature', 0, 80, 5, 0, 0, 12, 255, 0, '41.3', '2018-08-28 19:03:35', 2,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(3, 4, '0000044C', 1, 'Memory Usage', 0, 243, 6, 0, 0, 12, 255, 0, '16.82', '2018-08-28 19:03:25', 3,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(4, 4, '000005DC', 1, 'Process Usage', 0, 243, 31, 0, 0, 12, 255, 0, '20.5800', '2018-08-28 19:03:25',
4, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, '1;MB', None)
(5, 4, '0000044E', 1, 'HDD /media/pi/SETTINGS', 0, 243, 6, 0, 0, 12, 255, 0, '1.50', '2018-08-28
19:04:35', 5, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(6, 4, '0000044F', 1, 'HDD /boot', 0, 243, 6, 0, 0, 12, 255, 0, '32.32', '2018-08-28 19:04:35', 6, 0.0,
1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(7, 4, '00000450', 1, 'HDD /media/pi/data', 0, 243, 6, 0, 0, 12, 255, 0, '0.08', '2018-08-28 19:04:35',
7, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(8, 4, '00000451', 1, 'HDD /', 0, 243, 6, 0, 0, 12, 255, 0, '17.52', '2018-08-28 19:04:35', 8, 0.0,
1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
(9, 2, '00082009', 1, 'Biotonne Pi', 1, 243, 19, 0, 0, 12, 255, 0, 'Hello Again', '2018-08-28
11:40:52', 9, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, None, None)
Select all Temperatures
(2, 40.8, 0.0, 0, 0, 0.0, 0.0, '2018-08-28 17:05:00')
(2, 41.9, 0.0, 0, 0, 0.0, 0.0, '2018-08-28 17:10:00')
(2, 41.3, 0.0, 0, 0.0, 0.0, '2018-08-28 17:15:00')
(2, 40.8, 0.0, 0, 0.0, 0.0, '2018-08-28 17:20:00')
Select Temperature by Threshold 41.0
(2, 41.9, 0.0, 0, 0, 0.0, 0.0, '2018-08-28 17:10:00')
(2, 41.3, 0.0, 0, 0.0, 0.0, '2018-08-28 17:15:00')

```

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.

Example of an SQL Select statement via tool SQLiteSpy 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
```

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

Example Timer Definition

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

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.

Delete

Purpose

To delete data from the Domoticz database devices tables using where clause on field Date.

Sample SQL create statement for table [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')));
```

The date is stored in field [Date] with format DATETIME DEFAULT i.e. datetime('now','localtime').

Sample Entries for table [Temperature] for device with idx=28

DeviceRowID	Temperature	Chill	Humidity	Barometer	DewPoint	SetPoint	Date
28	17.5	17.5	0	0	0.0	0.0	2018-08-27 14:30:00
28	18.9	18.9	0	0	0.0	0.0	2018-08-27 14:45:00
28	19.0	19.0	0	0	0.0	0.0	2018-08-27 14:50:00
...							

SELECT statement with WHERE clause

```
SELECT * FROM [Temperature] WHERE [DeviceRowID] = 28 AND [Date] < date('2018-08-28');
```

This works and returns all entries with a date < 2018-08-28.

DELETE statement with WHERE clause

Delete all entries from table [Temperature] with Date < 2018-08-28

```
DELETE FROM [Temperature] WHERE [DeviceRowID] = 28 AND [Date] < date('2018-08-28');
```

Checked result:

```
SELECT * FROM [Temperature] WHERE [DeviceRowID] = 28;
```

The result is OK, i.e. all entries with a date < 2018-08-28 have been deleted.

Delete all data related to device with idx=44 (Stromverbrauch)

Delete all data from Table Meter for the device with idx=44 (Stromverbrauch) with Select statement to check.

```
DELETE FROM Meter WHERE DeviceRowID=44
SELECT * FROM Meter WHERE DeviceRowID=44
DELETE FROM MultiMeter_Calendar WHERE DeviceRowID=44
SELECT * FROM MultiMeter_Calendar WHERE DeviceRowID=44
```

Python

Task: Delete temperature data.

Table: [Temperature]

<TODO> Add function, i.e. delete_temperature_by_date(conn, date).

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

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_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()

```

Command

```
python3 sql-delete-test.py
```

Output

```

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

```

SQLite3 Package

The sqlite 3 package enables to perform SQL operations from the command shell.

Install SQLite3

Login as user Pi and install the package by invoking:

```
sudo apt install sqlite3
```

Output

```
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) ...
```

SQL Statements Examples

For the examples, the Domoticz database domoticz.db is used:

```
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
```

Hint

Do not add a semicolon to the “.open” command, else a new database will be created with the semicolon at name end.

Example: .open domoticz.db; creates a new database with filename domoticz.db;

Schema Table

Use “.schema” tablename with semicolon. The tablename is case sensitive.

.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);
```

.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);
```

Selecting Entries

Select all entries from table LightingLog for the device with idx=161.

```
select * from lightinglog where devicerowid=161;
```

```
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 data for the device Aussen with idx=120.

```
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 data for the device Aussen with idx=120 and show the device Name and limit the number of entries to 5.

```
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
```

Select temperature data for the device Aussen with idx=120 for a specific day.*Note*

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.

```
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  
..
```

Deleting Entries

Delete all entries from table LightingLog for the device with idx=161.

```
delete from lightinglog where devicerowid=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>  
sqlite> delete from lightinglog where devicerowid=161;  
sqlite> select * from lightinglog where devicerowid=161;  
sqlite>
```

Updating Device

Be careful on using this – it is just an example – not recommended to use.

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;
```

SQL 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.

Example:

- .read test.sql (read in this case from the folder /home/pi/domoticz as set as default using cd domoticz)
- .read /home/pi/test.sql

```
File:  
/home/pi/domoticz/test.sql  
Content:  
select * from lightinglog where devicerowid=97;  
select * from lightinglog where devicerowid=98;
```

```
.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|
```

Appendix Domoticz Hints

Start, Stop, Status

Terminal commands

```
sudo service domoticz.sh start
sudo service domoticz.sh stop
sudo service domoticz.sh status
```

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)
 CGroup: /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

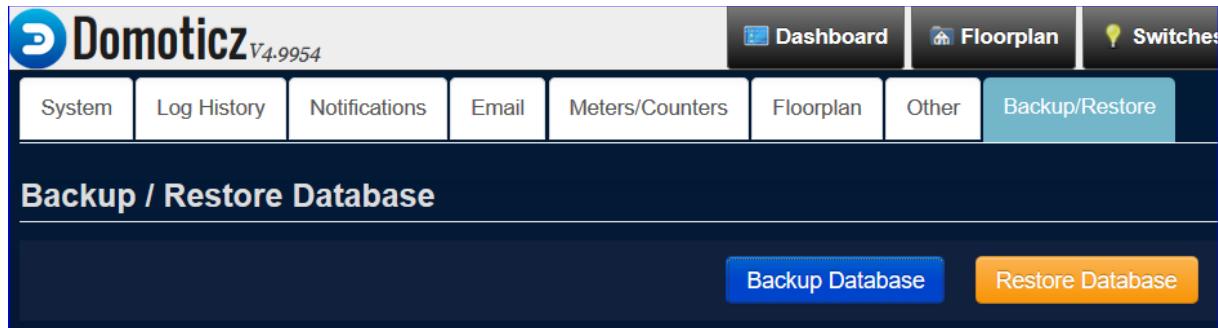
```
sudo ./domoticz -nowwwpwd
```

Create Backup

From the Domoticz UI go to:

Setup > Settings > Backup/Restore : Click Backup Database

This will create a file in the downloads folder on the development device.



Change Loglevel

```
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 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.

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: Queue thread started...
2018-09-12 16:18:42.447 Status: Incoming connection from: NNN.NNN.1.3

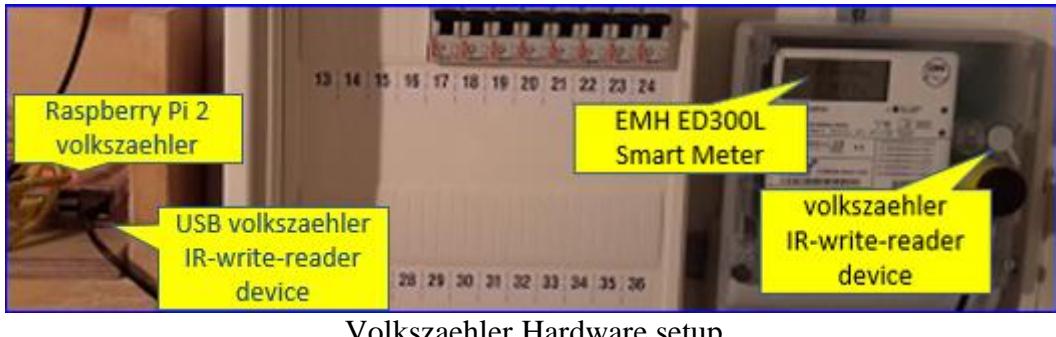
Updating Domoticz
Binary install update:
Use the Web Interface to update Domoticz (Setup->Check for Updates)

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 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.



Volkszaehler Hardware setup

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 rpi-domoticz-ip
netmask 255.255.255.0
broadcast NNN.NNN.0.255
gateway rpi-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 volkszahler Raspberry Pi and the Home Automation Raspberry Pi are direct connected via an Ethernet cable. A standard cable is used, no twisted wires required.

The communication uses ETH0 with a static network addresses on both sides.

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

volkszahler Raspberry Pi [Raspbian GNU/Linux 8 (jessie)] Linux 39 4.1.19-v7+ #858	Home Automation Raspberry Pi [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
0000020 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 Webbrowser the volkszaehler Frontend using its ip address: <http://rpi-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**":

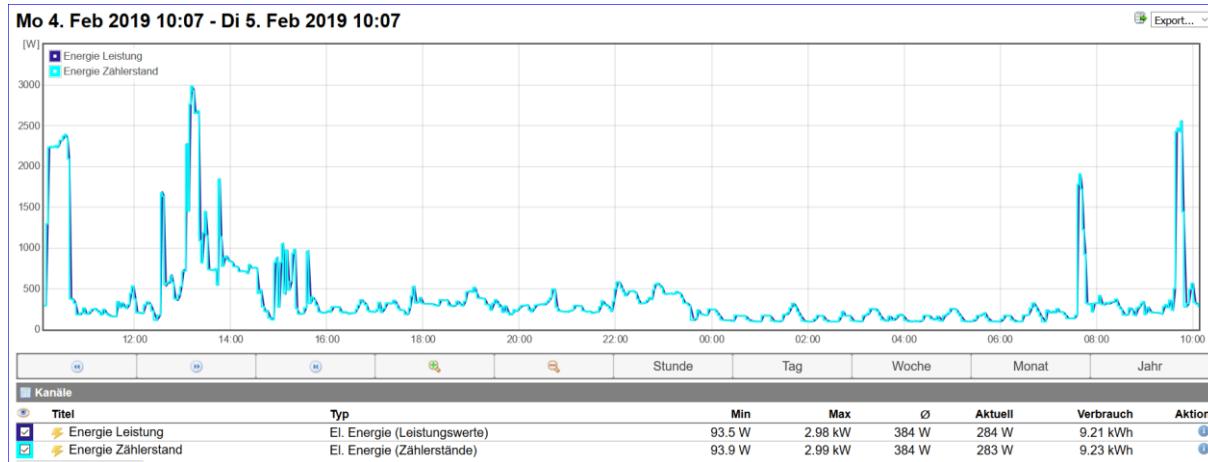
```
/***
 * vzlogger configuration
 * Use properly encoded JSON with javascript comments
 * http://wiki.volkszaehler.org/software/controller/vzlogger#configuration
 * Online configuration editor: http://volkszaehler.github.io/vzlogger/
 */

{
    // 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 serve 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://rpi-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>Details für Energie Leistung</p> <table border="1"> <tr><th>Eigenschaft</th><th>Wert</th></tr> <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> </table> <p style="text-align: center;">Daten Löschen Bearbeiten Schließen</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>Details für Energie Zählerstand</p> <table border="1"> <tr><th>Eigenschaft</th><th>Wert</th></tr> <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> </table> <p style="text-align: center;">Daten Löschen Bearbeiten Schließen</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>Bearbeiten von Energie Leistung</p> <table border="1"> <tr><th>Eigenschaft</th><th>Wert</th></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></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 style="text-align: center;">Speichern Abbrechen</p>	Eigenschaft	Wert	Titel	Energie Leistung	Öffentlich	<input checked="" type="checkbox"/>	Farbe		Stil	lines	Füllgrad		Achse	auto	Auflösung		Kosten		Initialverbrauch		<p>Bearbeiten von Energie Zählerstand</p> <table border="1"> <tr><th>Eigenschaft</th><th>Wert</th></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></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 style="text-align: center;">Speichern Abbrechen</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://rpi-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://rpi-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 a 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://rpi-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://rpi-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://rpi-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://rpi-vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json
```

Get One Record

```
http://rpi-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": 1447262000365, "to": 1447316802613, "min": [1447316802613, 246.49134831111], "max": [1447316802613, 246.49134831111], "average": 246.491, "consumption": 3752.3, "rows": 1}}
```

Get the last value

```
http://rpi-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://rpi-vz-ip/middleware.php/data/23844c00-8895-11e5-a642-  
6d8ce36c9a44.json?ts=1284677961150&value=12
```

Using the volkzaehler Frontend to display data

<http://rpi-vz-ip/frontend/?uuid=23844c00-8895-11e5-a642-6d8ce36c9a44>

Appendix Tools

In progress developing some tools for managing Domoticz.

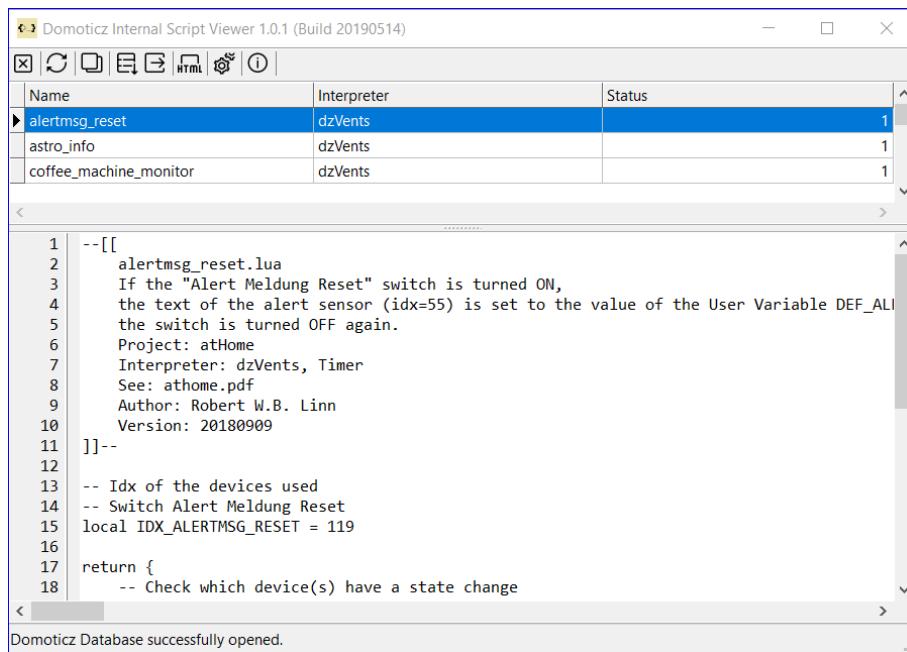
Domoticz Internal Script Viewer

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



The SQL query to select the scripts from the Domoticz database domoticz.db:

```

SELECT
Name, Interpreter, XMLStatement, Status
FROM EventMaster
WHERE Interpreter="dzVents"
ORDER BY Interpreter, Name;

```

Domoticz MQTT Logger

Purpose

To view Domoticz MQTT messages published, by subscribing to the topic “domoticz/out”.

Analyse the properties and the values for a device.

More information about [MQTT](#).

Solution

A Node-RED flow, “**mqtt-logger.flow**”, using a mqtt-in node as entry point from which the message payload is routed to Node-RED Dashboard (node-red-dashboard) nodes.

The log lists three messages listed (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)

The tab Settings enables to set options:

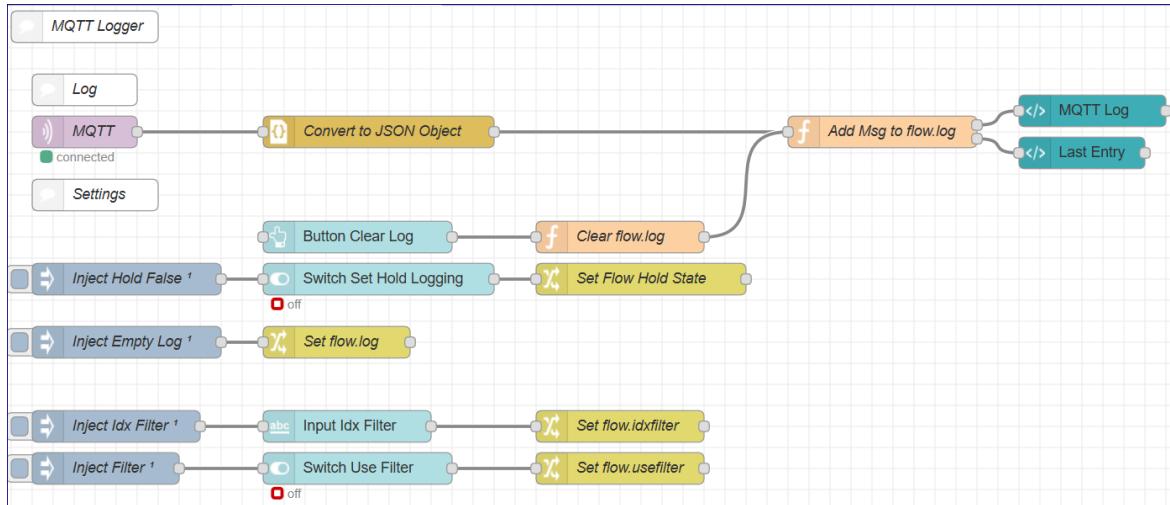
- Define the idx filter as comma separated string
- Switch to use the idx filter
- Switch to put logging on hold
- Button to clear the log

Recommend setting the flow disabled when not using, as rather high message traffic could take place.

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.

Node-RED Flow



Node-RED Dashboard UI

The Tab **MQTT Logger** and two groups **Log** and **Settings**.

Log Entry 1 (Detailed)
<ul style="list-style-type: none"> {"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":"Electricity House","nvalue":0,"stype":"kWh","svalue1":"481","svalue2":"86820.0","unit":1} {"Battery":255,"RSSI":12,"description":"RPi Memory Usage increases over time and needs to be watched.","dtype":"General","id":"00000044C","idx":1,"name":"RPi Speicher","nvalue":0,"stype":"Percentage","svalue1":"21.96","unit":1} {"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"000000DC","idx":107,"name":"Usage","nvalue":0,"stype":"Custom Sensor","svalue1":"46.0000","unit":1}
Last Entry (Idx:107, Name:Process Usage)
{"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"000000DC","idx":107,"name":"Usage","nvalue":0,"stype":"Custom Sensor","svalue1":"46.0000","unit":1}

Settings

- Idx Filter (separate by comma)
115
- Use Filter
- Hold Logging

CLEAR LOG

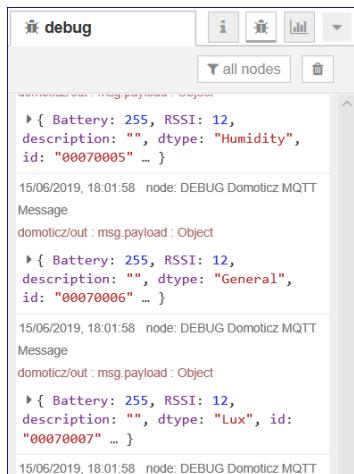
Alternative Logger

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 server	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 Web 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
}
```