Domoticz Home Automation Workbook

Home Automation & Information System

Robert W.B. Linn

26.05.2020

DISCLAIMER

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

Table of Contents

Table of Contents 1
Purpose 11
Components 14
Hardware 14
Software 15
External Services 15
Setup 16
Development Device 16
Raspberry Pi 16
Installation 16
Network Static IP Address 17
Wired Network 18
WLAN Power Save Mode 19
Raspian Check Version & Update 20
Samba 21
Persistent USB Devices 25
Domoticz 26
Installation 26
Settings 27
Folders 28
Events 28
Functions 29
Introduction 29
Air Pressure (BME280) 30
Purpose 30
Circuit 30
I2C 31
Domoticz Configuration 32
Barometer Widget 33
Automation Script 33
Alert Message 35
Purpose 35
Solution 35
Automation Script 36
Alert Message Reset 37

Database Tables 38
Ambient Light (BH1750) 39
Purpose 39
I2C 39
Parts Needed 39
Wiring 39
Device Setup Domoticz 39
Device Setup ESP Easy 40
Trigger Switch Light 41
Android App atHome 43
Purpose 43
Solutions 43
Anemometer (TFA 30.3168) 45
Purpose 45
Device 45
Setup 45
Blinds (Somfy RTS Pure) 46
Purpose 46
Domoticz Configuration 46
Dashboard & Switches Widget 47
MQTT Messages 47
Coffee Machine Monitor 48
Purpose 48
Solution 48
Automation Script 49
Contact Detection (433 Mhz) 50
Purpose 50
Solution Sensor PB-62R 50
Domoticz Configuration 51
Alarm Detection 51
Automation Script 52
Alarm State Reset Switch 53
MQTT 54
Days-To-Go 56
Purpose 56
Solution 56
Atomation Script 57

Electric Usage 59
Purpose 59
Solution 59
Electric Usage House (volkszaehler) 60
Purpose 60
Solution 60
Domoticz Configuration 61
Polling 62
Automation Script 64
MQTT 65
HTTP API Request 65
Log Examples 67
Database Table 67
volkszaehler 69
Electric Usage Rooms/Devices 84
Purpose 84
Solution 84
homematicIP HmIP-PSM (Plugin) 84
Revolt SF-436.m 86
Email Control 91
Purpose 91
Solution 92
Event Monitor 99
Purpose 99
Domoticz Configuration 99
Automation Script 100
Maintenance 101
Hardware Monitor Raspberry Pi 102
Purpose 102
Solution 103
Solution RPi volkszaehler 104
Solution RPi Domoticz Production 107
Indoor Air Quality Monitor (Plugin, Tinkerforge) 108
Purpose 108
Solution 108
Info Message 111
Purpose 111

Solution 111
Automation Script 111
Key Dates 112
Purpose 112
Domoticz Configuration 113
Concept 113
CSV Input File 113
Automation Script 114
Python CSV File Generation Script 116
Music Player 120
Purpose 120
Solution 120
Installation 121
Overview Devices & User Variables 122
Roomplan and Floorplan 123
Remote Control Functions 124
Automation Script 125
Notes 126
Purpose 126
Solution 126
Domoticz Configuration 128
Export Notes 129
Database Records 130
Philips Hue 132
Purpose 132
Hue Bridge Configuration 132
Hue Devices 132
Hue Light Add New 133
Hue Control Tests 134
Hue Timed Switch Lights 136
Hue Lights Control Node-RED 138
Hue Direct Access 140
Postbox Notifier (RaspberryMatic) 148
Purpose 148
Solutions 148
Solution Alert 150
Solution Alert + Switch 153

Solution Alert + Switch + Voltage 157
Radiator Thermostat (RaspberryMatic) 162
Purpose 162
Solution Plugin 162
Solution Selector Switch & Thermostat 169
Raspberry Pi Monitor 173
Purpose 173
Solution 173
Monitoring 174
River Elbe Tide 176
Purpose 176
Solution 176
Node-RED Flow 177
Domoticz Configuration 177
Roomplans 178
Purpose 178
Define Roomplan 178
Define Device Names 179
Overview Roomplans 180
RFXCOM RFXtrx433E 182
Purpose 182
Prepare RFXtrx433E 182
Domoticz Configuration 184
Troubleshooting 185
Soil Moisture Monitor (Plugin, Tinkerforge) 186
Purpose 186
Solution 186
Stock Quotes 187
Purpose 187
Domoticz Configuration 188
Solution Automation Script 189
Solution Node-RED 191
Notifications 195
Enhancements 197
Temperature & Humidity 433Mhz 198
Purpose 198
Device 198

Setup 198
Time Control 199
Purpose 199
Solution 200
Domoticz Configuration 200
Various 204
Timers 205
Purpose 205
Rule 205
Device Timers 205
Timerplans 208
Automation Events 209
Web App Site Control 212
Purpose 212
Web Frontend Customized 214
Purpose 214
Web Front End Sample 214
Concept 214
HTML & JavaScript 216
Web Radio Volumio 218
Purpose 218
Volumio Setup 219
Solution Options 219
Solution Automation Script 221
Solution Node-RED 225
Favorites List 230
Wind TFA 30.3168 231
Purpose 231
Solution 231
Automation Script 232
User Variables 233
Syntax 233
List 233
SQL 233
Python 234
Usage in Scripts 235
Usage in Browser Interactive 236

Usage in Node-RED 237
Usage in Node-RED Dashboard....238
Example Ambient Light Threshold 238
Custom Icons....240
Explore....244
API Interaction 244
Purpose....244
Option HTTP....244
Option MQTT....246
Recommendation 248
Example Syntax HTTP vs MQTT 248
Custom Pages 251
Purpose....251
Examples 251
Prepare 252
Examples 258
Events - Overview 265
Purpose....265
Event Execution Order....265
Event Scheduling & Trigger 266
Event Database Tables....266
Event Script Viewer....266
Event Development....267
Example Basement Humidity Monitor 268
Events – dzVents 277
Device Properties....277
Script....277
Property List....278
External Modules....278
Shared Helper Functions....282
ESP8266....283
Purpose....283
Experiment 283
Prototype 283
Prepare Arduino IDE 283
Parts Needed....284
Wiring 284

Circuit 284
Source Code 285
ESP Easy 286
Purpose 286
Overview Experiments 286
Flash ESP 287
Update ESP 287
Domoticz Controller 288
Rules 289
GPIO Commands 290
Experiments 291
Domoticz switching ESP LED 291
ESP Ambient Light to Domoticz 293
ESP controlling Hue Brightness 299
ESP BMP280 to Domoticz 304
Kodi Music Player 307
Purpose 307
Hardware 307
Installation 307
Kodi JSON-RPC API Communication 309
Domoticz Automation Events 310
Domoticz Explore 313
Appendix: Kodi JSONRPC Command (Selection) 325
Appendix: Various Hints 328
MQTT 331
Purpose 331
Install 331
Mosquitto 333
Python 335
Node-RED 340
Purpose 340
Install & Update 340
Access Flows & Dashboard UI 342
Start, Stop, Log 342
Manage Node Packages 342
Folder Locations 343
Example Flows 344

Example Dashboard UI Flows 350
Plugin Development 355
Purpose 355
Hints 356
Plugin Traffic Light (Tinkerforge Building Blocks) 360
RaspberryMatic 367
Purpose 367
Solution 367
Installation 367
Addon XML-API 368
Addon CUxD 385
Plugin Considerations 390
CCU Device(s) Information 391
CCU Devices homematicIP 393
CCU Duty Cycle Monitor 394
Experiments 399
SQL 415
Purpose 415
SQLite Shell 416
Python Scripts 431
Tinkerforge 437
Purpose 437
Hardware 437
Software 437
Concepts 438
Appendix Domoticz Hints 454
Start, Stop, Status 454
Lost Username and Password 454
Create Backup 456
Change Loglevel 456
Troubleshooting 457
Device Widget turns yellow/red 457
Check Domoticz Status 458
GLIBC Version not found 458
Various 459
Appendix Domoticz Build Source 460
Appendix volkszaehler Setup 462

Hardware 462
Software 462
Setup Raspberry Pi 463
Option 1 WLAN static IP address 463
Option 2 Ethernet static IP address 463
volkszaehler 465
PowerMeter Setup 465
USB Configuration 466
Define Channels 467
vzlogger Setup 468
Appendix Tools 475
Domoticz Internal Script Viewer 475
Purpose 475
Domoticz MQTT Logger 476
Purpose 476
Solution 476
Alternative Logger 479
RaspberryMatic Statelist 481
Purpose 481
Solution 481
Node-RED 481

Purpose

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

Objectives

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

Remarks

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

Functions

- Air Quality monitor [Plugin].
- Ambient light (from ESP8266 running ESP Easy) with threshold.
- Charts for selective weather items, room temperature & humidity.
- Coffee machine monitor – start and end time, info message.
- Control Somfy roller shutters with RTS motors in rooms.
- Control Volumio music player whilst listening to web radio.
- Custom icons.
- Custom Pages - simple to more complex examples.
- Display temperature & humidity measured in rooms.
- Electricity power & energy for selected devices / rooms (washer, MakeLab etc.).
- Electricity power & energy for the house from "volkszaehler" with charts.
- Event monitor for selected devices.
- Hardware Monitor for the Raspberry Pi's used (see HardwareMonitor.zip).
- HomeMatic (RaspberryMatic) CCU3 integration with variety of devices and plugins [HmIP-eTRV], [HmIP-PSM].
- Key Dates management and information organised in a roomplan.
- MQTT Logger.

- MQTT subscribe & publish messages to trigger actions or information.
- Monitor stock quotes.
- Music Player (using LibreELEC with Kodi).
- Notes with a subject & content and save, copy or notify.
- Philips Hue Lighting System control via Hue Bridge for ZigBee devices.
- Philips Hue Light controlled via ESP8266 with slide potentiometer & 4-digit-7-segment-display.
- Postbox Notifier.
- RaspberryMatic Statelist with devices and datapoints.
- Raspberry Pi system information with charts and threshold email notification.
- Remote Control Domoticz devices using Email client - includes a simple Android test app.
- Security door & window wireless contact detectors.
- Soil Moisture monitor for plants [Plugin].
- Time Control to track & control the time spent in hours, per activity block (with start- & end-time) and total/day, on a generic task.
- Timers for single devices or complex tasks (Automation events dzVents script).
- Web App Site Control [Info] – Node-RED alternative to the standard Domoticz Web UI.

Explore How to Use/Develop

- Domoticz running on a Raspberry Pi (setup, configure).
- Scripting dzVents (Automation), JavaScript (Custom Pages), Lua, Python (Plugin).
- Domoticz RESTful API generating JSON returns from HTTP requests.
- Domoticz Python Framework (mainly with Tinkerforge Building Blocks).
- MQTT messaging.
- Node-RED as an alternative script engine and User Interface.
- SQLite3 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 CCU and integrate into Domoticz.
- External services.
- Domoticz Android App (native client, to be determined).
- Advanced User Interfaces, i.e. Node-RED, Bootstrap …

Credits

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

Drawings are created with Fritzing.

Abbreviations

domoticz-ip = Domoticz system IP address, GUI = Domoticz UI in browser, dzVents=Automation events using dzVents Lua scripts, HmIP = homematicIP, TF=Tinkerforge

Components

Hardware and software used for this project.
More details to be found in chapters Setup , Functions and other.

Hardware

Overview of the various hardware used.

- Raspberry Pi 3B+ v1.2 (Domoticz Production System)
- RFXCOM RFXtrx433E USB 433.92MHz Transceiver
    - TFA Dostmann TS34C (temperature & humidity)
    - TFA Dostmann 30.3168 Wind Meter (speed, direction, temperature)
    - Somfy Blinds RTS Pure (blinds & 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 CCU (Raspberry Pi 3B+) with variety of connected devices like HmIP-PSM, HmIP-eTRV-B, HmIP-eTRV-2, HmIP-SDWO, HmIP-SDWM
- Pine A64 1GB - Volumio Music Player
- Raspberry Pi GPIO - BPM280 Barometer (I2C)
- Raspberry Pi 2B - Kodi Music Player
- Raspberry Pi 3B+ (Domoticz Development System) for testing
    - Domoticz latest BETA version to ensure up-to-date with development
    - Interfacing, i.e. GPIO, RaspberryMatic; Automation events dzVents scripts
    - Node-RED …

Software

- Domoticz Production System:
  - Raspbian GNU/Linux 9 (Stretch) [NOOBS 2.8.2]
  - Domoticz V4.10417 Stable
- Domoticz Development System:
  - Raspbian GNU/Linux 10 (buster) [NOOBS 3.2.1]
  - Domoticz latest BETA version (i.e. V4.11707)
- 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 3.49.17.20200131
  (full compatible HomeMatic CCU3 3.45.7 firmware)
- Node-RED v1.0.3 – various functions
- WinSCP - to manage & exchange files between the development device and the Raspberry Pi's
- PuTTY - to run terminal commands

Note

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 system, has a monitor, keyboard and mouse connected. This to ease the setup.
After initial installation, the Raspberry Pi is running headless and VNC is used to remote access the Raspberry Pi desktop.

NOOBS
Download the Raspberry Pi NOOBS version from here .
NOOBS Version 2.8.2, 2018-06-27, Zip archive NOOBS_v2_8_2.zip.

On the Development PC
- insert a new SD card (used 32GB)
- format the SD card (used SDFormatter)
- unzip NOOBS_v2_8_2.zip to the SD card

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

Additional configuration Desktop Menu Preferences > Raspberry Pi Configuration >
Interfaces enabled SSH, VNC, SPI, I2C, 1-Wire
Any other software for the project Functions are described under Functions.

Network Static IP Address

It is useful to define a static IP address for the Raspberry Pi running Domoticz, i.e. fix IP address to access Domoticz from a browser (http://domoticz-ip:8080) and for a Samba shared folder used 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
...
```

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

Wired Network

Whilst this Home Automation solution is evolving, decided to use wired connections for the several 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.
Hint
Do not forget to set the password and restart Samba.

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

Domoticz Home Folder as a share

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

Prework Raspberry Pi Domoticz System

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

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

Content

```
[DoProDomoticz]
Comment = Raspberry Pi Domoticz Production folder
Path = /home/pi/domoticz
Browseable = yes
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 *net view* command and browsing network shares.
writable	yes|no - Allows user to add/modify files and folders. Default samba shares are readonly
guest ok	yes|no - Allows “non authenticated” users to access the share

After configuration: Perform following steps after the Samba configuration.

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

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

Restart samba:

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

Restart domoticz:

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

Prework Windows

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

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

To connect from the Windows explorer:
Network drives:
Add new drive with URL: \\domoticz-ip\DoProDomoticz
Username: pi, Password: ******

Example Music Shared Folder

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

Create the folder music:

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

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

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

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

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

Restart Samba:

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

Restart Domoticz:

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

Connect from Windows Explorer using:

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

Example Samba Installation with Share

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

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

Note on the [global] configuration in file /etc/samba/smb.conf
This is a basic global configuration as set by the section [global]. All settings apply to the defined shares.

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

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

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

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

```
Y: = \\domoticz-ip\DoDevDomoticz
```

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

y [dodevdomoticz] 19.948.808 k of 27.784.728 k free

y:\domoticz*.*

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

Persistent USB Devices

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

USB Symlink for RFXCOM RFXtrx344E

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

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

Get USB device information

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

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

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

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

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

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

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

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

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

Domoticz

Installation

The Domoticz installation is initiated by running terminal command:

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

Answers to the questions

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

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

Example Domoticz Installation Complete Message (for the development system)

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

Settings

Changes made to the default Domoticz settings:

System	
User Interface	Language: English Theme: Default
Location	Name: Pinneberg (Pro) Latitude: 53.636470 Longitude: 9.798251 *Note* 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
	Note 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 excepts 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:

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 terminal 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 1K resistors between VCC and the Data (SDA) & Clock (SCL) lines.

Raspberry Pi B+ GPIO Header

I2C

Check if the I2C module (i2c_bcm2835) is loaded using the terminal command “lsmod”.
If not, activate the I2C interface in the Raspberry Pi preferences.
Below list is only an extract of the various command output.

```
$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

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

Add new Device

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

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

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

Add the device, named BME280 and check the widgets.

TemperatureWidget	Weather Widget

Barometer Widget

The BME280 measures pressure, temperature and humidity.
As the BME280 is directly connected to the Raspberry Pi, the temperature and the humidity are used to monitor the Server Room (holding various hardware).
In the Domoticz GUI, a Barometer Widget holds the value taken from the BME280.

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

Luftdruck Virtual Sensor Device List Entry (idx=116)

116	VirtualSensors	00082116	1	Luftdruck	General	Barometer	1027.6 hPa

Weather Widget Luftdruck

Automation Script

Created a dzVents script to update the pressure and pressure forecast from the BME280 data.
This is done using the Domoticz Event Editor.

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

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

dzVents Lua Script
Event name: 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
```

```
    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_Forecase_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 > Dashboard and GUI > Utility.

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

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

- Raspberry Pi monitor (Script: rpi_monitor.lua)
- Hue Lights living room timed switch (Script: hue_wz_timer.lua)
- More see the dedicated functions

Solution

Add a Virtual Device 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 (which is the idx of the virtual device “Alert Messages” - see previous).

6	IDX_ALERTMSG	Integer	55

Dashboard

Widget

Log Entries

Automation Script

Monitor the level of the Alert Messages Device with idx=55.
If the level (nValue) > 4 (or any other value set by uservar TH_ALERTTOEMAIL), send the Alert Message to Email receipients.
Event name: alertmsg_monitor

```
local IDX_ALERTMSG = 55
local IDX_TH_ALERTTOEMAIL = 14

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

Switch the Hue MakeLab ON and update the Alert Messages Device with a message.
Event name: hue_control_makelab
Depends on: global_data

```
local IDX_HUE_MAKELAB = 118;

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

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

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

- domoticz.helpers.isnow(domoticz)
- domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, msg)

Alert Message Reset

Purpose

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

Solution

Create 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

Event name: alertmsg_reset
Depends on: global_data

```
local IDX_ALERTMSG_RESET = 119

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

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

Database Tables

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

```
SELECT * FROM DeviceStatus WHERE ID=55;
```

ID	Har...	Devi...	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		

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

```
SELECT * FROM LightingLog WHERE DeviceRowID=55;
```

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

To delete entries, option are:
The GUI > Utility > Alert Message Widget > Log > Clear

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

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

Ambient Light (BH1750)

Purpose

To measure, using an ESP8266 microcontroller running ESP Easy, the Ambient Light in Lux with 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 addreses depending the ADDR pin connection:

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

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

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 a browser by opening:
http://esp-device-ip > goto Devices > select Task Edit and configure.
The Domoticz MQTT controller is used and Domoticz device with idx=46 (as previous defined).
The Lux value is updated every 300 seconds (5 minutes) to Domoticz.

Domoticz Log 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

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 switchescould 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://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)

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 swiches OFF after two hours.

Solution

Monitor the value of the Electric Usage House Device (idx=171,).
If 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 an other device is turned on prior coffee machine resulting in power usage above threshold, then the Coffee Machine Alert Message is send – 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_COFFEEMACHINEMONITOR = 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_COFFEEMACHINEMONITOR).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 anf alert message
    if (domoticz.devices(IDX_ELECTRICUSAGEHOUSE).WhActual >
domoticz.variables(IDX_TH_COFFEEMACHINEMONITOR).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**
From: KOBERT-GOODS Costs: ~10EUR Battery: 12V (MN21/23) with duration ~4 month.	From: KOBERT-GOODS Costs: ~10EUR

• Left = Magnet
• Right = Detector with
  ○ top left red LED (lights up if the contact is “broken”),
  ○ Middle right reset button used for Domoticz to learn the device.
	(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

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 sent a change state, which means a manual or timer trigger is required to reset the state to “Normal”.	<TODO> Watch for RFXCOM updates and check if this device is supported.

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		
160	RFXtrx433e	5C0700	0	Security Main Door	Security	X10 security	Alarm + Tamper	7	

The device widget is listed in the tab Switches.

Alarm Detection

	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!
	State reset to Normal via HTTP API Request: http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=160&switchcmd=Normal

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

HTTP JSON Result

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

MQTT JSON Result

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

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

Domoticz Log

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

Another HTTP API Request Test

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

The MQTT nvalue is 2.
Note
For the state "Alarm + Tamper" the nvalue is 130.

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

For more information, read the Appendix volkszaehler Setup, but some screenshots of the volkszaehler setup and web frontend.

Hardware Setup volkszaehler

Web Frontend volkszaehler

Domoticz Configuration

Device

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

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

The widget is added to the Utility tab.

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

The 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)

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.

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 tupels, data.average, data.consumption.
3. Parse the JSON string to get the power usage from key “data.average”.
   (which calculates the average electric usage over the last minute- the polling interval)
4. Update the Domoticz device Electricity House
   The device property “Energy read” is set to Computed, which means Domoticz calculated the Energy (kWh) from the given Power (Watt).

Cross Check Domoticz Calculation vs Smart Meter Display

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

The result:
Domoticz 3.2 kWh vs Smart Meter 3 kWh (display delta 8077 – 8074).
As the Smart Meter does not display digits, the result is getting close.
<TODO>Performsome more cross checks.

HTTP API GET Request

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

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

Request Examples with JSON Response

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

Obtain the Power Usage from key: data.average

Data for the last minute

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


```
{"version":"0.3","data":{"tuples":[[1559376630383,296.907,1],[1559376632787,299.501,1],[1559376635056,3
17.32,1],[1559376637366,311.688,1],[1559376639582,324.91,1],[1559376641817,322.148,1],[1559376643928,34
1.071,1],[1559376646081,334.417,1],[1559376648138,350.024,1],[1559376650187,351.391,1],[1559376652233,3
51.906,1],[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],[15593766818
44,400.445,1],[1559376683575,415.945,1],[1559376685339,408.163,1],[1559376687028,426.288,1],[1559376688
804,405.405,1],[1559376690535,415.945,1]],"uuid":"5b8ea2a0-b342-11e8-bec6-
15c040e6d041","from":1559376627958,"to":1559376690535,"min":[1559376630383,296.90721677136],"max":[1559
376664129,806.72268794865],"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":[1559
377278643,334.72803378454],"average":334.728,"consumption":0.2,"rows":1}}
```


The average power is 334.728 Watt calculated from 33 tuples.

Automation Script

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

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

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

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

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

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

Domoticz Log 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: "electricusagehouse
2019-07-29 10:03:00.396 Status: dzVents: Info: ------ Start internal script: electric_usage_house:
HTTPResponse: "electricusagehouse"
2019-07-29 10:03:00.403 Status: dzVents: Info: Electric Usage vz: 2528 W, Status: 200
2019-07-29 10:03:00.420 Status: dzVents: Info: Wh = Actual: 2408, Today: 2579, Total: 334381
2019-07-29 10:03:00.420 Status: dzVents: Info: ------ Finished electric_usage_house
```

MQTT

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


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


Properties

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

HTTP API Request

Request

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

JSON Response (extract)


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

Properties

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

HTTP API Information

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

Log Examples

The log for the device “Electricity House” shows the

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

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

Note
This information can also be obtained by amending the previous dzVents Lua script “electricity_usage_update”:

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

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

Database Table

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

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

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

Example SQL SELECT on table Meter

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

volkszaehler

Overview

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

Note
See Appendix volkszaehler Setup for actual update and more details.

References: Raspberry-Pi, API, Debug, EMH ED300L, Node-RED .

Setup Raspberry Pi

The Raspberry Pi is installed following the installation procedure described here.

Fix IP Address

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

/etc/dhcpcd.conf

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

/etc/wpa_supplicant/wpa_supplicant.conf

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

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

USB Port Symlink

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

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

Check

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

Serial Link

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

```
stty -F /dev/ttyUSB-VZ
1:0:8bd:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
```

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

```
sudo nano /etc/rc.local
# Set ttyUSB-VZ
stty -F /dev/ttyUSB-VZ
1:0:8bd:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0

exit 0
```

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

```
xxd </dev/ttyUSB-VZ
00000000: 1b1b 1b1b 0101 0101 7607 000f 0016 bc87  ........v.......
00000010: 6200 6200 7263 0101 7601 0107 000f 0553  b.b.rc..v......S
00000020: 942d 0b09 0145 4d48 0000 532f 3801 0163  .-...EMH..S/8..c
00000030: 6a00 7607 000f 0016 bc88 6200 6200 7263  j.v.......b.b.rc
00000040: 0177 010b 0901 454d 4800 0053 2f38 0701  .w....EMH..S/8..
00000050: 0062 0aff 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  ..............EM
```

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

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

vz Setup Channels

Setting up the channels using the vz frontend (url=http://vz-ip) is crucial.
Must read: https://wiki.volkszaehler.org/howto/emh_pv-anlage?redirect=1
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
UUID 5b8ea2a0-b342-11e8-bec6-15c040e6d041	**UUID** 958bce60-b342-11e8-b54f-bbec0573e1f4

// /etc/vzlogger.conf
{
// Zählerstand
"uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041",
"middleware" : "http://localhost/middleware.php",
"identifier" : "1-0:1.8.0"
}
	/etc/vzlogger.conf
	{
	// Leistungswert
	"uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4",
	"middleware" : "http://localhost/middleware.php",
	"identifier" : "1-0:16.7.0"
	}

Web Frontend Settings	**Web Frontend Settings**

{
"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\u00e4hlerstand",
"yaxis":"auto"}
}
	{
	"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"}
	}

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],[1559311267248,1
87.598,1],[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],[155931128987
5,197.044,1],[1559311291775,189.474,1],[1559311293613,195.865,1],[1559311295568,184.143,1],[15593112974
57,190.577,1],[1559311299399,185.376,1],[1559311301330,186.432,1],[1559311303284,184.237,1],[1559311305
205,187.402,1],[1559311307116,188.383,1],[1559311309048,186.335,1],[1559311310979,186.432,1],[155931131
2891,188.285,1],[1559311314865,182.371,1],[1559311316619,205.245,1],[1559311318351,207.852,1],[15593113
20072,209.181,1],[1559311321804,207.852,1],[1559311323589,201.681,1]],"uuid":"5b8ea2a0-b342-11e8-bec6-
15c040e6d041","from":1559311261537,"to":1559311323589,"min":[1559311314865,182.37082168777],"max":[1559
311320072,209.18070811092],"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 datato the middleware.

Configuration File

The configuration file is /etc/vzlogger.conf. Must read to understand the settings.
Ensure no empty lines are spaces after the last }.
Important
After making changes (sudo nano /etc/vzlogger.conf) restart the vzlogger and check status.

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

/etc/vzlogger.conf

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

Start Stop Status

Terminal commands

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

Logfile

Check

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

Delete

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

vz Web Frontend

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

Read more here.

Database Management

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

User vz-admin

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

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

Login phpMyAdmin

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

Check Database Size

URL

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

Result


```
{"version":"0.3", "capabilities": {"database":
  {"data_rows":67235725,"data_size":5935382528,"aggregation_enabled":1,"aggregation_rows":2611868,"aggr
egation_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 ist the hard way of deleting data via MySQL statements.
Example deleting all data older then 24 hours from database volkszaehler, table data.

```
mysql --user=root -praspberry

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

mysql> use volkszaehler

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

Note
This operation might take long…

Check the database size after completion

Option 2

Use terminal command

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

Option 3

Use URL

Prepare mysql

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

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

URL deleting data

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

Result JSON


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


URL Check databse size

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

Result JSON


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

vzclean

Purpose

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

Solution

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

Script location:

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

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

Hint
After running vzclean or to check adhoc, use URL

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

Python

vzcleanconfig.py

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

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

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

    # print(db)
    return db
```

vzclean.ini

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

vzclean.py

```
#!/usr/bin/env python
"""
File: vzclean.py
Author: Robert W.B. Linn, 20180909
Project: AtHome
Purpose: Delete all channel data prior last 24 h from database volkszaehler, table data. Run once per
day via crontab.
The scripts logs to a file. If the script is executed with parameter l, then the log is printed to the
console.
Run the script: python3 vzclean.py  or python3 vzclean.py l
The script is executed via crontab "5 0 * * * /home/pi/scripts/vzclean.sh l" 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 COUN()
    :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 befor 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 == 'l':
            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 l
```

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 control the power and measure the energy consumption, voltage, current, power of connected devices to an power outlet.
The power outlet could be used for selected devices (coffee machine, washer, dryer etc.) or dedicated rooms (MakeLab, Library).

Solution

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

The Domoticz Python plugin “homematicIP Pluggable Switch and Meter (HmIP-PSM)” has been developed for this purpose - described here.

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

homematicIP HmIP-PSM (Plugin)

New hardware added using the plugin.

Devices (created by the plugin)

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

Note
The devices with the idx=226 & 223 are used.

Roomplan MakeLab

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

Revolt SF-436.m

NOT USED – see issues.

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

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

This device communicates with the RFXCOM RFXtrx433E with following requirements:

- RFXtrx type2 firmware,
- ELEC5 protocol,
- RSL option (Hardware > RFXTRX433E > set mode).

RFXCOM Test

Step 1
Plug the Revolt SF-436.m device in the wall plug (without pressing the green button!).

Step 2
Test if the Revolt device is recognized by the RFXtrx433E (with firmware version RFXtrx433E_Ext2_1025) using the RFXCOM RFXmngr.

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

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

Notes

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

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

Domoticz Configuration

Step 1
Allow Domoticz to receive new hardware for 5 minutes (Domoticz GUI > Setup > Settings > Hardware/Devices).
The Domoticz log lists initial entries for the new Power Device:

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

This means Domoticz recognizes the new Power device.

Step 2
Switch to the Domoticz Devices List to check out the new devices
Recognized by Domoticz (Type, SubType, Data, Idx) are these four devices:

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

Initial Devices List

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

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

Utility Tab Widget & Log with a few days

MQTT

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


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


Properties

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

HTTP API Request

Request

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

JSON Response (extract)


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


Properties

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

HTTP API Information

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

Issues

Whilst using this device, encountered several issues.

1) If the RFXtrx433E is more 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 system from the public internet.

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

Notes

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

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

Solution

Overview

The solution uses a Node-RED flow (named Email Control), acting as kind of middleware, to handle Domoticz commands being sent via email (i.e. email client or App).
Required are the Node-RED nodes: node-red-node-email (thanks to the author).

There are two solutions developed based on MQTT or HTTP
The preferred solution is HTTP, because it offers more commands but it is also the underlying Domoticz API.
The MQTT solution is not described 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 nd stripped for the submitted HTTP API Request URL.
5. submits the Domoticz HTTP API Request URL.

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

Notes

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

Email Subject & Body

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

The email body, as plain text, contains a string with commands.
Any command as described in the Domoticz API/JSON URL's manual can be defined in the email body.

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

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

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

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

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

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

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

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

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

Node-RED Flow

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

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

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

Node Check From Email Address (Type switch)
Check the header from email address. This depends on the email header as send by the provider email server. The message header is JSON parsed.
Example:

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

to compare against a dedicated (=hardcoded) email address.
This ensures only emails are handled which are sent by an accepted email address.

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

Copy the output of debug note and insert a JSON string in the Inject node. Analyze the string on the from address. Set the JSON property in the debug node to determine the from address. If to 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:8080/json.htm?" + msg.payload;
// Return the message to be published by the mqtt out node
return msg;
```

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

Node Sent HTTP Request Domoticz (Type http request)
Sent the message URL using GET method and return a UTF-8 string.

Node Email Response (Type email)
Sent the HTTP API Request response as email.
The SMTP server uses port 587 without option “Use secure connection”, but with option “Use TLS?”. This is required for the email server used by the author.

The response from Domoticz is a JSON formatted string as email body.
If the result of the command is OK, then the email body is like:

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

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

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

Node-RED Flow Enabled or Disabled

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

Enabled	Disabled

Control Examples

Domoticz E-Mail Control HTTP - Example Thermostat

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

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

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

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

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

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

E-Mail Response Subject & Body
RESPONSE:Thermostat MakeLab Off

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

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

Domoticz E-Mail Control HTTP - Example Get User Variable

Control Action
Get the status of the user variable TH_ALERTTOEMAIL with IDX=14

E-Mail Subject & Body
DOMOTICZ:Get Status Uservariable TH_ALERTTOEMAIL
type=command&param=getuservariable&idx=14

Domoticz Log Entries
This request is not logged.

The e-mail response should be sufficient.

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

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

E-Mail Response Subject & Body
RESPONSE:Get Status Uservariable TH_ALERTTOEMAIL


```
{
  "result" : [
    {
      "LastUpdate" : "2019-11-07 10:49:35",
      "Name" : "TH_ALERTTOEMAIL",
      "Type" : "0",
      "Value" : "4",
      "idx" : "14"
    }
  ],
  "status" : "OK",
  "title" : "GetUserVariable"
}
```

Android Client App

An example of a simple app to test sending emails with commands (subject & body) to the Domoticz system via Node-Red as the middleware.
The Android app is created with the development tool B4A from Anywhere Software.
B4A offers a simple way to develop native Android apps with the B4X language.

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

The app has uses a B4X customlistview with items related to a function. 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)	The preferences dialog uses the B4XPreferences library with a json configuration file (Files Folder > prefdialog.json).

Event Monitor

Purpose

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

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

Note
Just a few entries as an example.

Domoticz Configuration

Create a text device to display the events.

Domoticz GUI > Setup > Hardware > Hardware VirtualSensors > Press Create Virtual Sensors
Name: Event Monitor
Sensor Type: Text

Domoticz GUI > Setup > Devices > Scroll down to the last list entry.
A new device is listed:

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

Automation Script

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

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

Shared helper function: global_data
Add a function to update the text of the Event Monitor device.

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

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

Event name: event_monitor

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

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

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

Update the dzVents Lua script to loop over the devices.

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

Maintenance

Clear the device log from time-to-time.

Option Domoticz GUI
Tab Utility > Widget Event Monitor > Log > Clear

Option SQLite3 Command
Note
Requires SQLite3 Package to be installed.

Example

1. Open a terminal.
2. Login as user Pi
3. Change to the domoticz database folder
   cd /home/pi/domoticz
4. Start sqlite3 command
   sqlite3
5. Select device log entries to check if there are entries
   select count(*) from lightinglog where devicerowid=161;
6. Delete device log entries
   delete from lightinglog where devicerowid=161;
7. Select device log entries to check if entries are deleted
   select count(*) from lightinglog where devicerowid=161;

```
Login Using username "pi".
cd domoticz
sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
sqlite> select count(*) from lightinglog where devicerowid=161;
46
sqlite> delete from lightinglog where devicerowid=161;
sqlite> select count(*) from lightinglog where devicerowid=161;
0
```

Hardware Monitor Raspberry Pi

Purpose

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

RPi Data Monitored
CPU Usage %, CPU Temperature °C, Discspace Used %, RAM Used %.

RPi Devices (Name)
Domoticz Production (pidopro), Domoticz Development (pidodev), volkszaehler (pivz), homematic (pihm - planned).

The hardware data for all the RPi's is displayed in the Domoticz development system.

Screenshots of the Hardware Monitor Dashboard & Floorplan.

Solution

The data provided by each RPi: CPU Temp, CPU Usage, RAM Usage, Disc Usage.

For the RPi's 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 RPI's, a webserver solution "hwmon" is used to gather the hardware data.

The Domoticz Development system:

- has 4 virtual sensor devices (device name data) for each of the RPi's, i.e. pivz CPU Temp, pivz CPU Usage etc.
- has a roomplan "Hardware Monitor" with all the hardware monitor devices
- has a flooplan "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 system, gathers all used devices data from the Domoticz production system.
The hardware monitor devices (#4) are selected ad used to update the devices on the Domoticz development system.

Automation Script

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

Pseudo Code
Place a HTTP request to the Domoticz production system (pidopro) every minute triggered by a timer.
The request obtains all data from the used devices.
The HTTP request has a unique callback.
Handle the HTTP JSON response by

- selecting a device by its idx
- get the data and update the Domoticz device on the Domoticz development system

Indoor Air Quality Monitor (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
A screenshot of the prototype in action.

Air Quality Bricklet measures the IAQ Index (ppm), IAQ Condition, IAQ Accuracy, Air Pressure (mbar), Humidity (%), Temperature (C).

Referencing Tinkerforge documentation: *_The IAQ index is a measurement for the quality of air. To calculate the IAQ index the Bricklet detects ethane, isoprene (2-methylbuta-1,3-diene), ethanol, acetone and carbon monoxide (often called VOC, volatile organic components) by adsorption. These gas measurements are combined with the measurements of air pressure, humidity and temperature to calculate the final IAQ index.*

- **IAQ Condition Levels** (6) range=condition (color):
  0-50=Good (green), 51-100=Moderate (yellow), 101-150=Unhealthy sensitive groups (orange), 151-200=Unhealthy (red), 201-300=Very Unhealthy (purple), 301-500=Hazardous (maroon).
- **IAQ Index Accuracy Levels** (4): Unreliable, Low, Medium, High.

Ambient Light Bricklet measures the Illuminance (lx).

LCD 20x4 Bricklet displays the IAQ Index ppm, IAQ Condition, Temperature C, Humidity %, Air Pressure mbar, Illuminance lx.

RBG LED Bricklet indicates the IAQ Condition Level Color.

The plugin polls in regular intervals data from the **Indoor Air Quality Station**.

Domoticz Indoor Air Quality Devices created - Name (Type,SubType):

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

Note
Enhanced functionality, like switch LCD backlight, switch lights (depending Lux) are realized with dzVents Lua scripts.

For more information or get the latest version, go here (GitHub).
See also Python Plugin Development.

Note
The dashboard with room Indoor Air Quality Monitor (in mobile mode).

Info Message

Purpose

NOT USED – Replaced by the Alert Message.

To display an Info Message on the Dashboard.
The Info Message is populated by various dzVents events, i.e.

- Raspberry Pi monitor (event name: rpi_monitor)
- Hue lamps living room timed switch (event name: hue_wz_timer)
- More see the dedicated functions

Solution

Add a Virtual Sensor named “Control Message”, Type General, SubType Text.

52	VirtualSensors	00082051	1	Control Message	General	Text	Hue MakeLab Switching OFF 16:55:34

In the various dzVents events, the device is updated.

Control Message	**Hue MakeLab switched OFF 17:18:05**

Automation Script

Event name: info_msg

```
local IDX_HUE_MAKELAB = 118;
local IDX_CONTROLMSG = 52

local function isnowtime(domoticz)
    return domoticz.time.rawTime
end

return {
    active = true,
  on = {
    devices = {
     IDX_HUE_MAKELAB
    }
  },
  execute = function(domoticz, device)
    if (device.state == 'On') then
     device.switchOff()
      domoticz.devices(IDX_CONTROLMSG).updateText(device.name .. ' switched OFF ' ..
isnowtime(domoticz))
    end
  end
}
```

Key Dates

Purpose

To display key dates in a roomplan “Termine” and in the Web App Site Control.

The key dates are also displayed in the Info Tab of the Web App Site Control. Rather handy to quickly check some important (key) dates.

Note
In the mean time many more devices are defined. This is just an example for the key waste dates.

Brief Description
Various key dates are maintained in a comma-separated-values (CSV) text file located in the dzVents script folder. Each line of the CSV file is related to a Domoticz device (VirtualSensor,Type Text). Each line contains the idx and the various dates.
Once per day, a dzVents Lua Automation scripts reads the CSV file line-by-line and updates the text of the device if the days from now till next date is greater or equal 0 (datediff).
If the datediff is 1 (set by a threshold), a notification is sent to alert.

The background for this function was to manage waste calendar dates for locations.
Therefor, the main key dates are waste calendar dates (“Abfall Termine”) for a city (location) and waste type. The waste types are (in German): Rest, Bio, Gelb, Papier, Schad.
The dates are provided by the respective city waste management (“Abfallwirtschaft”) for the cities Pinneberg and Hohenfelde.

The roomplan “Termine” is defined (Domoticz GUI Setup > More Options > Plans > Roomplan) holding all devices as defined in the CSV file.

Domoticz Configuration

The devices are created as Virtual Sensors Type General, SubType Text using the Hardware Type Dummy.

Idx	Name	Enabled	Type
6	VirtualSensors	Yes	Dummy (Does nothing, use for virtual switches only) Create Virtual Sensors

98	VirtualSensors	00082098	1	Biotonne Pi	General	Text	06-09-2018
99	VirtualSensors	00082099	1	Papier Pi	General	Text	21-09-2018
100	VirtualSensors	00082100	1	Restmüll Pi	General	Text	30-08-2018
101	VirtualSensors	00082101	1	Gelber Sack Pi	General	Text	30-08-2018
102	VirtualSensors	00082102	1	Schadstoff Pi	General	Text	08-09-2018
103	VirtualSensors	00082103	1	Biotonne Ho	General	Text	27-08-2018
104	VirtualSensors	00082104	1	Papier Ho	General	Text	17-09-2018
105	VirtualSensors	00082105	1	Restmüll Ho	General	Text	27-08-2018
106	VirtualSensors	00082106	1	Gelber Sack Ho	General	Text	28-08-2018

Concept

A dzVents Lua Automation script event, “keydates_update” executed dzVents time trigger reads a CSV file, which contains per line the Device Idx, Interval, waste dates.
The device is updated based upon the difference in days between now and the waste date.
The script also notifies via email (receipients defined in the Domoticz Settings Email and Notification) if the waste target date is 1 days from actual date.

CSV Input File

The CSV file is read line by line, using per line, the Device Index idx (field 1) to update a date (fields 2 and more depending available dates).

CSV Line Syntax: idx, interval, date1, dateN

idx	NN 98, 99…
date	Format Day-Month-Year dd-mm-YYYY 09-08-2018

Note
If a device is not used, ensure to take out the line in the keydates.csv file
OR
place -1, prior idx.
If the idx is used, remove the -1, entry. Example:
-1;999,23-03-2020,22-06-2020,21-09-2020,16-12-2020
999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

Example CSV file line entry

The example shows for

- Idx 98 (“Biotonne Pi”): Number of dates for the remaining year 2018 and full year 2019 (only first few entries).
- Idx 999: Line starts with -1, means this device is not used.

```
98,13-12-2018,28-12-2018,10-01-2019,24-01-2019,07-02-2019
-1,999,32,23-03-2020,22-06-2020,21-09-2020,16-12-2020
```

CSV File name & location
The CSV file keydates.csv, located in folder /home/pi/domoticz/scripts/dzVents/scripts.

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

Automation Script

The dzVents Lua script event is time triggered once per day at 00:30. See source for details.
Event name: keydates_update

```
…
--[[
    keydates_update.lua
    Read a csv file with keydates per device.
    The keydates file "keydates.csv" is located in the dzVents scripts folder.
    (keydatesfile = '/home/pi/domoticz/scripts/dzVents/scripts/keydates.csv')
    Line entry csv file: idx,date1,dateN.
    Example:
    999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

    Action:
    Update the device text with the date if the first datediff in days >= 0

    Note:
    If a device is not used, ensure to take out the line in the keydates.csv file
    OR place -1, prior idx. If the idx is used, remove the -1, entry. Example:
    -1;999,23-03-2020,22-06-2020,21-09-2020,16-12-2020

    Project: domoticz-homeautomation-workbook
    Interpreter: dzVents
    Author: Robert W.B. Linn
    Version: 20200127
]]--

-- threshold in days to notify
-- if the difference of the current day to a keydates day equals threshold
-- then notify
local TH_KEYDATES = 1

-- Helpers
function string:split( inSplitPattern, outResults )
    if not outResults then
        outResults = { }
    end
    local theStart = 1
    local theSplitStart, theSplitEnd = string.find( self, inSplitPattern, theStart )
    while theSplitStart do
        table.insert( outResults, string.sub( self, theStart, theSplitStart-1 ) )
        theStart = theSplitEnd + 1
        theSplitStart, theSplitEnd = string.find( self, inSplitPattern, theStart )
    end
    table.insert( outResults, string.sub( self, theStart ) )
    return outResults
end

-- the keydatescalendar file is updated once per year
```

```
local keydatesfile = '/home/pi/domoticz/scripts/dzVents/scripts/keydates.csv'
local sep = ","

-- read the csv file
function readandupdate(domoticz)
    -- read the file line by line.
    -- each line contains an array with device[1], dates[2...]
    for line in io.lines(keydatesfile) do
      -- split the line by comma , (sep) and assign to a table
      local keydatestable = line:split(sep)
      local idx = tonumber(keydatestable[1])
      -- flag to check if a keydate has been found
      local datefound = 0
        -- the idx must be > 0 else not used
        if idx > 0 then
            -- check the dates, start at 2 because 1=idx
            for i = 2, #keydatestable do
                -- get the keydatesdate array entry
                local keydate = keydatestable[i]
                -- split the keydate into an array with 3 entries day, month, year (D,M,Y)
                local keydatesplit = keydate:split("-")
                -- calculate the daysdiff between now and the keydate
                local daysdiff = domoticz.helpers.datediffnow(keydatesplit[1], keydatesplit[2],
keydatesplit[3])
                -- get he month which is array entry index 2
                local month = tonumber(keydatesplit[2])
                -- check if daysdiff => 0 and the date is not found
                if daysdiff >= 0 and datefound == 0 then
                    datefound = 1
                    -- update the device text with the new date
                    domoticz.devices(idx).updateText(keydate)
                    -- check number of days left compared to the threshold
                    -- i.e. if 1 day left, then notify using device name & text.
                    -- the device text contains the date
                    -- NOT USED = handled by the dzVents script keydates_calendar_notifier
                    if daysdiff == TH_KEYDATE then
                   -- print ('Device ' .. idx .. ' notifying...')
                        message = '[ACHTUNG] ' .. domoticz.devices(idx).name .. ' ' ..
domoticz.devices(idx).text
                        -- update alert message
                        -- msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
                        domoticz.log(message, domoticz.LOG_INFO)
                        domoticz.notify(message)
                    end
                end

                -- check if for the last entry daysdiff less 0, then no new date available
                -- this is used f.e. for hohenfelde as last keydate is sep or oct
                if i == #keydatestable and daysdiff < 0 then
                    datefound = 1
                    print (idx .. '=' .. keydate .. ', days=' .. daysdiff .. ', KEIN TERMIN')
                    -- update the device text with the info that there is no date (yet) available
                    domoticz.devices(idx).updateText('Kein neuer Termin')
                end

            end -- for

            -- check if a keydate has been found, if not then set Kein termin message
            if datefound == 0 then
                print (idx .. '=' .. 'KEIN TERMIN')
                -- update the device text with the info that there is no date (yet) available
                domoticz.devices(idx).updateText('Kein neuer Termin')
            end

            -- JUST SOME LEARNERS
            -- print the content
            -- for i = 1, #keydatestable do
            --  print( keydatestable[i] )
            -- end
            -- print device idx and length of the csv string
            -- print (keydatestable[1] .. "Length:" .. #keydatestable)
        end -- if idx > 0
    end
end
```

```
return {
  on = {
        timer = {
          -- for tests use every minute
          -- 'every minute'
          'at 00:30'
        }
    },
  execute = function(domoticz)
        readandupdate(domoticz)
  end
}
```

Python CSV File Generation Script

A Python 2 script is used to create the keydates.csv file from an iCal file.

Script

```
# -*- coding: cp1252 -*-
## File:    wastecalicstocsv.py (Python 2)
## Purpuse: Create CSV file with waste dates from ICS iCal file Pinneberg and empty entries Hohenfelde

## Notes:
## The generated csv file wastecal.csv must be placed in /home/pi/domoticz/scripts/dzVents/scripts
## Domoticz Device Idx and Interval
## Biotonne Pi=98,16
## Papier Pi=99,32
## Restmuell Pi=100,16
## Gelber Sack Pi=101,16
## Schadstoff Pi=102,32
## Biotonne Ho=103,16
## Papier Ho =104,32
## Restmuell Ho=105,16
## Gelber Sack Ho=106,16
## The generated file has to be merged with the keydates.csv file located on the Domoticz system:
## /home/pi/domoticz/scripts/dzVents/scripts
import csv, io, sys
import datetime
from datetime import date

wastecalendarics = 'wastecal-pi-2019.ics'

# Define the wastecalender csv file located in the same folder as the python script.
# CSV Format: wastetype, date1, date2 ...
# Date format dd-mm-YYYY, i.e. 09-08-2018
wastecalendarcsv = 'wastecal-2019.csv'

# Check the python version to set the file open arguments
if sys.version_info[0] == 2:  # Not named on 2.6
    icsaccess = 'rb'
    icskwargs = {}
    csvaccess = 'wb'
    csvkwargs = {}
else:
    icsaccess = 'rt'
    icskwargs = {'newline':''}
    csvaccess = 'a'
    csvkwargs = {}
    #csvkwargs = {'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(wastecalendarics, icsaccess, **icskwargs) as icsfile:
    # lines = icsfile.readlines()
    # Loop thru the file line by line
    for line in icsfile:
        # Strip the CRLF from the line
        linet = line.strip()
        # Print the line for tests
        # print linet
        # Check the entries

        # If token event end is found and an event has been 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

    # 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()
```

Music Player

Purpose

- To Remote Control, via Domoticz, music of a Kodi Media Player running on a Raspberry Pi (or other hardware) with LibreELEC.
- To explore setting up Kodi, communicating with Kodi using JSON-RPC API requests, control & monitor from various systems (Remote Browser, Domoticz and other).
- To use the Kodi Media Player, as-is without developing additional add-ons or change the Kodi environment - stricly use JSON-RPC API requests.
- To use Domoticz Virtual Sensors and a single Automation Event dzVents Lua script.
- Although Kodi is a media player, the focus is on playing music only.

Solution

Prior creating the Music Player, explored how to communicate with Kodi using JSON-RPC API requests. Recommend to read first the chapter Kodi Music Player.

The description next uses several screenshots as easier to explain. If hard to read, zoom in the page.

The screenshot below outlines the functionality with the various Domoticz Devices grouped in a Roomplan “Kodi Music Player”.

The remote control is On, playback mode Play and the text devices Playing & Progress are updated every 10 seconds.

Domoticz Dashboard Mobile
Room plan with all devices used by the Music Player.
ACTION: Switch Devices to Control the Player.
Play or pause current music title. Select next or previous title.
Turn the volume On or Off.
Set the volume between 0 – 100%.
Stop the player or shutdown. To start, a push-button connected to the Raspberry Pi GPIO pin #3 is used. Also possible to shutdown the Raspberry Pi.
Use Domoticz to Remote Control. When connected to a player, turn On followed by Play.
Select a player. The player is defined in a user variable which is updated by the switch selection level action. More players can be added by enhancing the switch selector levels.
INFO: Text Devices updated every 10 seconds by Automation Script dzVents Lua.
Current song playing or event status message depending switch selected.
Progress: Percentage & time played; Total title play time.

Installation

System	Action
Kodi	Setup a Kodi Player or use an existing – required is the IP address of the player. More then one player can be used.
Domoticz	Create the devices to remote control the player using the hardware “Dummy” (name, type, subtype, switch type): • Player Playing, General, Text • Player Progress, General, Text • Player Playback, Light/Switch, Selector Switch • Player Volume, Light/Switch, Switch, Dimmer • Player Remote Control, Light/Switch, Switch • Player Selection, Light/Switch, Selector Switch • Player System , Light/Switch, Selector Switch
Domoticz	Create two user variables • Name: URL_PLAYER, Type: string, Value: http://player-ip:8080/jsonrpc • Name: URL_DOMOTICZ, Type: string, Value: http://domoticz-ip:8080/json.htm
Domoticz	Create an event dzVents Lua Music_Player. • Copy the music_player.dzVents. • Set the idx of the devices. • Set event to On and save the event
Kodi	Start the Kodi Player via browser (Chorus) or app like Kore and play album etc.
Domoticz	Select the player > select On > select Play. After 10 seconds the information playing and progress are updated.
Domoticz	Check the log file for errors.
Domoticz	Create a roomplan and /or floorplan (with the roomplan)

Overview Devices & User Variables

Devices

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
123	VirtualDevices	00082123	1	Player Progress	General	Text	1.3% / 00:00:03 / 00:03
117	VirtualDevices	00082117	1	Player Playing	General	Text	Artist - Title
118	VirtualDevices	000140C6	1	Player Playback	Light/Switch	Selector Switch	Set Level: 10 %
125	VirtualDevices	000140CD	1	Player System	Light/Switch	Switch	Off
126	VirtualDevices	000140CE	1	Player Remote Control	Light/Switch	Switch	On
127	VirtualDevices	000140CF	1	Player Selection	Light/Switch	Selector Switch	Set Level: 10 %
124	VirtualDevices	000140CC	1	Player Volume	Light/Switch	Switch	Set Level: 49 %

User Variables

Idx	Variable name	Variable type	Current value
4	URL_DOMOTICZ	String	http://192.168.1.179:8080/json.htm
3	URL_PLAYER	String	http://192.168.1.62:8080/jsonrpc

Device Player Selection (idx=127)

Selector Levels with action to set the user variable URL_PLAYER

Selector Levels:

Level	Level name
0	Off
10	Player A
20	Player B

Level name: Add

Selector actions:

Level	Action
0	
10	http://192.168.1.179:8080/json.htm?type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.62:8080/jsonrpc
20	http://192.168.1.179:8080/json.htm?type=command&param=updateuservariable&vname=URL_PLAYER&vtype=2&vvalue=http://192.168.1.58:8080/jsonrpc

Domoticz is running on a Raspberry Pi with IP address 192.168.1.179.
The music player is running on a Raspberry Pi with IP address 192.168.1.62.
When the sector level changes, the user variable (idx=3, name=URL_PLAYER) is updated.
The Domoticz URL is used to clear logs from the devices to avoid spoiling the database with not needed entries.

Roomplan and Floorplan

Roomplan “Kodi Music Player”	Floorplan “Kodi Music Player”

Remote Control Functions

The next screenshots outline the various Remote Control Functions.

Player A Remote Control On

The remote control is On and asking to press Play to start listening. If the player is not playing, the device Playing states "Not playing".
If the remote control is not On, no other switch can be used.

Player A Playback Play

The remote control is On and Play is selected. Devices Playing Artist & Title and the Progress are updated every 10 seconds.

Player A Playback Pause

The remote control is On and Playback is set to Pause. Device Playing Progress states Pause with last progress information.

Remote Control Off

The remote control is Off. Device Playing is updated and Progress set to 0%. The Playback selection is Off (not visible).

Player B Selected

Switched from Player A to Player B. The remote control and playback are turned Off. To start listening press On and Play.

Player B Not Reachable

Pressed On and Play, but Player B is not turned On. Device Playing displays the error message and the remote control is turned Off.
See also the Domoticz log for more info.

Player A Stopped

Player A stopped. The initial message is Player stopped, which is followed after 10 seconds by message "Not playing" and disabling device System.
The remote control is still On.
Use the Kodi User Interface or any other to start playing again. If playing, this is recognized by the event and the devices Playing & Progress gets updated after max 10 seconds.

Player A Shutdown

If the player is shutdown, the device Playing states "Player shutdown" followed by the message that the server is not reachable, which is followed by turning Off the remote control.

Automation Script

A single dzVents Lua script is used to control the Music Player.
The source code of the script, kodi_music_player.dzvents, is not included as having around 450 lines.
The way the script works in a nutshell – look at the source for details.

There are 3 triggers:
timer = { 'every hour' }
Function
Clear the device logs every hour to avoid spoiling the database with entries not used.

devices = { IDX_PLAYBACK, IDX_SETVOLUME, IDX_SYSTEM, IDX_REMOTECONTROL, IDX_PLAYERSELECTION }
Function
Handle device changes and submit JSON-RPC API commands via method openURL().
These are according to the screenshot shared earlier.

httpResponses = { CALLBACK_GETITEM, CALLBACK_GETPROPERTIES, CALLBACK_PLAYBACK }
Function
Handle the callback response of the JSON-RPC API commands.

Update Devices Playing & Progress
The devices are updated every 10 seconds
Pseudocode

```
DELAYSECONDS = 10
Trigger openURL(GETITEM, CALLBACK_GETITEM)
Handle CALLBACK_GETITEM
  Run openURL(GETPROPERTIES, CALLBACK_GETPROPERTIES)
Handle CALLBACK_GETPROPERTIES
  Run openURL(GETITEM, CALLBACK_GETITEM).afterSec(DELAYSECONDS)
```

Notes

1. The trigger to start device updates is by switching the device “Remote Control” On,
2. The change of the switch device triggers the first openURL for the JSON-RPC API GETITEM,
3. The GETITEM callback runs openURL for the JSON-RPC API GETPROPERTIES,
4. The GETPROPERTIES callback runs GETITEM again, but after a delay, set by the constant DELAYSECONDS to 10 seconds,
5. After the delay, the loop starts again from the GETITEM callback (3).

Notes

Purpose

To create notes with a subject & content and functions to save, copy, notify, export and clear. The background developing this function, was to be able to quickly save scribbles / short information when using various devices (PC, Tablets, Smartphones) in the local network.
The standard Domoticz HTTP API calls are used for the functions save/copy/notify/export/clear.
It is not possible to change a saved note, delete a selected note (see below under Limitations), because of not available as HTTP API calls.

Solution

- Text device “Notes” to store the notes
- Switch device “Notes Export” to export the textlog to a local file
- Custom page "Notes.html" with forms and buttons
- dzVents Automation script “Export_Notes”

The file custom page is stored in folder <path>/domoticz/www/templates, which holds all custom pages.

Form Note

The form contains the elements:

- Input - set the note subject (mandatory).
- Textarea – set the note content (optional).
- Buttons to handle the note subject & content:
  - Save - the note to the assigned Domoticz text device.
  - Copy - the note to the clipboard.
  - Notify - the note to all enabled notification systems.

Form Device History Log

The form contains the elements:

- Select - a note from the device history log and display the subject and content in the form Note fields subject & content.
- Buttons for the textlog:
  - Export the textlog to a local file.
  - Clear the text device history log.

Additional elements:

- Label - show status message after any action.
- Button Info - show about.

Limitations
Because standard API calls are used, a note can not be changed nor deleted. It is only possible to save a note or delete all notes from the database.
This means if f.e. want to change a note, a new note must be created.
It might be possible to change or delete a note via API calls for any future Domoticz releases – but not known so far.
A workaround – but be careful – is direct access the Domoticz database via SQLite Shell.
The notes are stored in table lightinglog.

Remarks

- Dialogs make use of Bootbox (Bootstrap framework) as also used by Domoticz.
- CSS classes are used from the Domoticz CSS styles (see folder domoticz/www/css).
- IMPORTANT: do not place buttons inside form tags - experienced strange behaviour with butons or bootbox.

Format of a Note

The text is stored in the Domoticz database table lightinglog as format: subject|content
To split the note subject and note content, the delimiter | (pipe) is used.
The content is by default stored without linefeeds. A placeholder replaces the textarea content "\r\n" with a placeholder text "CRLF".

When a note is selected from the device history:

- the subject is extracted from the text,
- the delimiter | is removed,
- in the content, the placeholder "CRLF" is replaced by "\r\n" to show proper linefeeds in the textarea.

Example note stored with subject|content delimiter and placeholder CRLF:

```
This is a Test Note|The content of this test note.CRLFThanks.
```

Domoticz HTTP API

The HTTP requests to retrieve or store the data make use of Domoticz HTTP API requests submitted via AJAX calls.

Domoticz Configuration

There are 2 devices required, a text and switch. The switch is used by dzVents automation script Notes_Export to save the log of the text device.
The custom page is located in <path>/domoticz/www/templates/Notes.html

Text Device
Create a text device to hold the note entries.
Domoticz > GUI > Setup > Hardware > Create Virtual Sensor (Hardware: VirtualSensors):
Name: Notes, Sensor type: Text.
After adding the Domoticz > Setup > GUI lists the new device:

```
Idx=240,Hardware=VirtualSensors,Name=Notes,Type=General,SubType=Text
```

Notes
The device idx is required for the custom page "Notes.html".
The device widget is accessible from the Domoticz GUI > Tab Utility.

Example with subject, delimiter | to split the subject from the content,CRLF placeholder for the content.

Switch Device
Create a switch device to export the device log to a text file.
Domoticz > GUI > Setup > Hardware > Create Virtual Sensor (Hardware: VirtualSensors):
Name: Notes Export, Sensor type: Switch.
After adding the Domoticz > Setup > GUI lists the new device:

```
Idx=241,Hardware=VirtualSensors,Name=Notes Export,Type=Light/Switch,SubType=Switch
```

Select the device widget and change the Switch Type from On/Off to Push On Button.

Custom Menu
Enable Domoticz > GUI > Setup > Settings > System > Active Menus: Custom.
The Domoticz GUI shows the tab Custom.
All custom pages fould in the Domoticz folder <path>/domoticz/www/templates are listed.

Screenshot Custom tab and dropdown with only the Notes custom page:

dzVents
Create a new automation script and paste the content of the file *Notes_Export.lua*.
Save the event as Export_Notes.
Set the Idx of the devices created:

```
-- idx of the devices (idx,hardware,name,type,subtype,switchtype)
local IDX_NOTES_SWITCH = 241;    --241,VirtualDevices,Notes Save Log,Light/Switch,Switch,push On Button
local IDX_NOTES_TEXT = 240;      --240,VirtualDevices,Notes,General,Text
```

Enable Domoticz > GUI > Setup > Settings > System > Active Menus: Custom.
The Domoticz GUI shows the tab Custom.
All custom pages found in the Domoticz folder <path>/domoticz/www/templates are listed.

Export Notes

The textlog can be exported to a local file which is triggered by the custom page Export button by submitting a HTTP API request to push switch device on.
The dzVents script listens to changes of the swith device and if on, the textlog is requested via HTTp API request. The response is a JSON string which is conveted to a Lua table. The Lua table is looped to get and built the notes data. The notes data is stored to a local file.

Concept with devices configured on the Domoticz development system

Database Records

Table

The table *LightingLog* of the Domoticz database holds the records for the function Notes. The field *DeviceRowID* is the idx of the device which is used in SQL statements.

Domoticz database operations can be performed using the SQLite Shell.

Example Select Notes

```
$ cd domoticz
$ sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
sqlite> .open domoticz.db
sqlite> .header on
sqlite> select * from lightinglog where devicerowid=240;
DeviceRowID|nValue|sValue|Date|User
240|0|Hello World|2020-03-15 19:04:28|
240|0|This is a test note|The content of the test note.CRLFCRLFThanks.|2020-03-15 19:29:21|
240|0|This is a test note 2|The content of the test note 3CRLFCRLFCRLFThanks.|2020-03-16 06:59:49|
240|0|This is a test note 3|The content of the test note 3CRLF-----------------------------
CRLFThanks.|2020-03-16 09:09:04|
sqlite> .exit
$
```

To change or delete a note, the *Date* field has to be used in the SQL statement WHERE clause together with the *DeviceRowID*.

… but as mentioned earlier, this function intent is just to save notes and be able to copy or notify …

CRUD Operations

To show how to use CRUD operations on the Domoticz database = Create (Insert), Read (Select), Update, Delete. Included comments (//) for the operations.

```
$cd domoticz
$sqlite3
SQLite version 3.27.2 2019-02-25 16:06:06
sqlite> .open domoticz.db
sqlite> .header on

// Select all records for the text device with idx=111 holding the notes
sqlite> select * from lightinglog where devicerowid=111;
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 1|The content of test note 1.CRLFCRLFThanks.CRLF||2020-03-16 16:31:05
111|0|This is test note 2|The content of test note 2.CRLF---------------------------
CRLFThanks.CRLF||2020-03-16 16:31:19

// Select a specific record for a date
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-16 16:31:19";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 2|The content of test note 2.CRLF---------------------------
CRLFThanks.CRLF||2020-03-16 16:31:19
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-16 16:31:19";
count(*)
1

// Insert a new record
sqlite> insert into lightinglog (DeviceRowID,nValue,sValue,User,Date)
   ...> values (111,0,"This is test note 3|The content of test note 3.CRLFThanks.","","2020-03-17
08:09:10");
```

```
// Check if the new record is inserted
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 3|The content of test note 3.CRLFThanks.||2020-03-17 08:09:10

// Update an existing record using where for a specific specific deviceid and date
sqlite> update lightinglog
   ...> set svalue="This is test note 3|The content of test note 3.CRLFUpdate 1CRLFThanks."
   ...> where date="2020-03-17 08:09:10";

// Check if the record is updated
sqlite> select * from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
DeviceRowID|nValue|sValue|User|Date
111|0|This is test note 3|The content of test note 3.CRLFUpdate 1CRLFThanks.||2020-03-17 08:09:10
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
count(*)
1

// Delete a record with where for a specific deviceid and date
sqlite> delete from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";

// Check if the recor is deleted using count()
sqlite> select count(*) from lightinglog where devicerowid=111 and date="2020-03-17 08:09:10";
count(*)
0
sqlite>
```

Philips Hue

Purpose

To control devices connected to the Philips Hue Bridge.

Hue Bridge Configuration

There is one Hue Bridge (Hardware idx=12) connected to the Router.

Note
Initially Groups & Scenes are not used.

Hue Devices

The list of Hue Devices, which are automatically created by Domoticz (Setup > Settings > Hardware allow for 5 minutes), in order of their idx:

Idx	Hardware	ID	Unit	Name	Type	SubType
110	Philips Hue Bridge 1	0000001	1	Hue Haustür	Lighting 2	AC
111	Philips Hue Bridge 1	00000002	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 if OFF again one minute later.
This type of script could be used, to switch

- the Hue Lamps in the Living Room, ON ‘at sunset’ and OFF ‘at 23:00’.
- The Hue Lamp Outside ON 'at sunset' and OFF 'at sunrise'.

Event name: hue_control2

```
-- Idx of the devices
local IDX_HUE_MAKELAB = 118;

return {
    active = true,
    on = {
        timer = {
          'at 15:49',
          'at 15:50'
        }
    },
    execute = function(domoticz, timer)
        if (timer.trigger == 'at 15:49') then
          domoticz.devices(IDX_HUE_MAKELAB).switchOn()
        end

        if (timer.trigger == 'at 15:50') then
            domoticz.devices(IDX_HUE_MAKELAB).switchOff()
        end
    end
}
```

Hue Timed Switch Lights

Switch the Hue lights in the living room (‘WZ’), ON ‘30 minutes before sunset’ and OFF ‘at 23:00’.
Update the Alert Message.

Automation Script
Event name: hue_wz_timer

```
--[[
    hue_wz_timer.lua
    Turn WZ Hue lamps on & off by given time: ON at sunset, OFF at 23:00.
    Update the alert message.
    Ensure the timer and the timer trigger are the same.
]]--

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

-- Idx of the devices
local IDX_HUE_WZABLAGE = 111
local IDX_HUE_WZTV = 112
local IDX_HUE_HAUSTUER = 110

-- Update alert message with alert level green.
local function setalertmsg(domoticz, state)
  local message= 'Hue Lampen ' .. state .. ' ' .. utils.isnowhhmm(domoticz)
    msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
end

local function hueswitchon(domoticz)
  domoticz.devices(IDX_HUE_WZABLAGE).switchOn()
    domoticz.devices(IDX_HUE_WZTV).switchOn()
    domoticz.devices(IDX_HUE_HAUSTUER).switchOn()
    setalertmsg(domoticz, 'an')
end

local function hueswitchoff(domoticz)
    domoticz.devices(IDX_HUE_WZABLAGE).switchOff()
    domoticz.devices(IDX_HUE_WZTV).switchOff()
    domoticz.devices(IDX_HUE_HAUSTUER).switchOff()
    setalertmsg(domoticz, 'aus')
end

return {
    active = true,
    on = {
        timer = {
          '60 minutes before sunset',
           '30 minutes before sunset',
          'at 23:00',
          'at 07:00',
          'at 08:00'
        }
    },
    execute = function(domoticz, timer)

        -- in the evening send message hue lights will be switched on
        if (timer.trigger == '60 minutes before sunset') then
            -- format date & time see www.lua.org/pil/22.1.html. X = time.
            local now=os.time()
            local nowplus30minutes = utils.converttimehhmm(os.date("%X",now+(30*60)))
        local message= 'Hue Lampen gehen an ' .. nowplus30minutes
            msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
        end

        -- switch the hue lights on & off in the evening
        if (timer.trigger == '30 minutes before sunset') then hueswitchon(domoticz) end
        if (timer.trigger == 'at 23:00') then hueswitchoff(domoticz) end

        -- switch the hue lights on & off in the morning
```

```
        -- NOT USED
        -- if (timer.trigger == 'at 07:00') then hueswitchon(domoticz) end
        -- if (timer.trigger == 'at 08:00') then hueswitchoff(domoticz) end

    end
}
```

Domoticz Log 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 browser.
Select a Hue and switch the light ON|Off or set Brightness Level.

Read more about Node-RED Appendix Node-RED.

Dashboard

The dashboard ui contains the tab Hue Light Control with a single group Hue Light.
The dropdown lists the Hue devices to control. If selected the state gets updated first, i.e the slider position, switch and the gauge level are set.
Change the light by moving the slider or set the switch.
If the device gets updated in the browser by the Domoticz GUI, then the Node-RED dashboard is updated.

Flow

Note
When the flow starts, a flow context array “idxarray” is defined with the Hue deviceidx’es, which are used for the dropdown with the Hue lights. An initial Hue is set in the dropdown. The dropdown notifies a change and requests the selected Hue device information via HTTP request. The HTTP response contains the full device information from which only a subset are used. A new JSON message is created with the required properties. These are 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 browser Domoticz GUI), the MQTT listener node recongnized this and the message payload is handled also by the node selecting the Hue with the connected nodes.
So, the “Select Hue by Idx” node has two inputs

1. from a converted HTTP request and
2. from the MQTT listener.

Hue Direct Access

Next some examples on how to direct access (control) a Hue Light without the use of the Domoticz Hardware “Philips Hue Bridge”.
The direct access possibilities are explored more out of curiosty.

To enable direct access with the Philips Hue Bridge, following information is required:

- the IP address of the Philips Hue Bridge
- an Philips Hue authorized username

Recommend to read in depth about Philips Hue Development at MeetHue and Get Started.

Abbreviations used in the next examples

- Hue = Philips Hue Light
- Hue 5 = Hue light number 5, named "Hue white lamp 4", from the list of Hue devices

For the next examples, HTTP commands are used to direct access the Hue 5.
Supported are four HTTP methods (see documentation Philips MeetHue).

- GET: fetch all information about the addressed resource
- PUT: modify an addressed resource
- POST: create a new resource inside the addressed resource
- DELETE: delete the addressed resource

The HTTP response is JSON formatted string.

HTTP - Browser

The communication with the Hue Bridge is direct using HTTP.
The HTTP commands are submitted from a browser.

HTTP GET Request State Hue 5

Request

```
http://hue-bridge-ip/api/username/lights/5
```

Result


```
{
"state":{"on":true,"bri":89,"alert":"select","mode":"homeautomation","reachable":true},
"swupdate":{"state":"noupdates","lastinstall":"2018-12-05T17:48:30"},
"type":"Dimmable light",
"name":"Hue white lamp 4",
"modelid":"LWB010",
"manufacturername":"Philips",
"productname":"Hue white lamp",
"capabilities":{"certified":true,
"control":{"mindimlevel":2000,"maxlumen":806},
"streaming":{"renderer":false,"proxy":false}},
"config":{"archetype":"classicbulb","function":"functional","direction":"omnidirectional",
"startup":{"mode":"safety","configured":true}},
"uniqueid":"00:17:88:01:03:b5:e2:68-0b",
"swversion":"1.46.13_r26312",
"swconfigid":"322BB2EC",
"productid":"Philips-LWB010-1-A19DLv4"
}
```


HTTP PUT Request Switch Hue 5

To change the state of a Hue light, submit an HTTP PUT request with a message body containing the state action as JSON string.

URL

```
http://hue-bridge-ip/api/username/lights/lightnr/state
```

Body Payload JSON

```
{"on":true}
```

or

```
{"on":false}
```

HTTP - curl

The communication with the Hue Bridge is direct using a *curl command* requesting HTTP PUT with JSON Data.
curl is a tool to transfer data from or to a server, using a protocol, i.e. FTP, HTTP and many more.
The *curl command* is submitted on a Raspberry Pi from the CLI (Command Line Interface, i.e. terminal).

curl HTTP PUT Request Switch Hue 5

curl Command Switch Hue 5 ON

```
curl --request PUT --data "{\"on\":true}" http://hue-bridge-ip/api/username/lights/5/state
```

Response

```
 [{"success":{"/lights/5/state/on":true}}]
```

curl Command Switch Hue 5 OFF

```
curl --request PUT --data "{\"on\":false}" http://hue-bridge-ip/api/username/lights/5/state
```

Response

```
 [{"success":{"/lights/5/state/on":false}}]
```

HTTP - dzVents Lua Script

To access Hue 5, two dzVents Automation scripts are shared:

- Get State Hue 5 – using openURL
- Put State Hue 5 – using os.execute curl command

IMPORTANT
State 20200122, Domoticz V4.11637:
The function openURL does not support the HTTP PUT method (only GET, POST).
To change the state of a Hue, PUT is required (see previous info under Hue Direct Access).

dzVents Lua Script State Hue 5 – HTTP

```
-- Hue Light Control
-- Test to obtain the state of a hue devices via direct http rest request (GET method)
-- The request is triggered by a switch with idx=5

local HUEBRIDGEIP = 'hue-bridge-ip'
local HUELIGHT5RES = "huelight5"
local HUEUSERNAME = "********"
local HUELIGHTNR = 5
local HUELIGHT5IDX = 73

return {
  on = {
    devices = { HUELIGHT5IDX },
    httpResponses = { HUELIGHT5RES }
     },
  execute = function(domoticz, item)
    if (item.isDevice) then
      local urlRequest = string.format("http://%s/api/%s/lights/%s",HUEBRIDGEIP,HUEUSERNAME,HUELIGHTNR)
      domoticz.log(urlRequest)
      domoticz.openURL({url = urlRequest,method = 'GET',callback = HUELIGHT5RES,})
    end

    if (item.isHTTPResponse) then
      domoticz.log('HUELIGHT5 Status Code = ' .. item.statusCode .. ', Name=' .. item.callback)
      if (item.statusCode == 200 and item.callback == HUELIGHT5RES) then
        if (item.isJSON) then
          -- This will give the full device status: domoticz.log('Item data:' .. item.data)
          -- Lets pick state and bri
          local result = item.json
          domoticz.log(string.format("Hue Light 5 state:%s, bri:%s",result.state.on, result.state.bri))
         end
         return
      else
        domoticz.log('[ERROR] Handling the request:' .. item.statusText, domoticz.LOG_ERROR)
        -- domoticz.log(item, domoticz.LOG_ERROR)
       return
      end
    end
  end
}
```

This is an example of a JSON string returned, from which attributes can be selected, i.e.

```
local result = item.json
result.state.on
result.state.bri
```


```
{
  ["manufacturername"]="Philips", ["productname"]="Hue white lamp",
  ["state"]=
    {
      ["mode"]="homeautomation", ["on"]=false, ["alert"]="select", ["reachable"]=false, ["bri"]=102},
      ["swupdate"]={["lastinstall"]="2018-12-05T17:48:30", ["state"]="noupdates"
    },
  ["capabilities"]=
    {
      ["certified"]=true, ["control"]={["mindimlevel"]=2000, ["maxlumen"]=806
    },
  ["streaming"]=
    {
      ["proxy"]=false, ["renderer"]=false}
    },
  ["uniqueid"]="00:17:88:01:03:b5:e2:68-0b",
  ["modelid"]="LWB010",
  ["name"]="Hue white lamp 4",
  ["swconfigid"]="322BB2EC",
  ["config"]=
    {
      ["archetype"]="classicbulb",
      ["startup"]=
        {
          ["configured"]=true, ["mode"]="safety"
        },
      ["direction"]="omnidirectional",
      ["function"]="functional"
    },
  ["swversion"]="1.46.13_r26312",
  ["productid"]="Philips-LWB010-1-A19DLv4",
  ["type"]="Dimmable light"
}
```

dzVents Lua Script Switch Hue 5 – curl

```
-- Hue Light Control
-- Test direct switching on off via curl with json command and http put request

local HUEBRIDGEIP = 'hue/bridge/ip'
local HUELIGHT5RES = "huelight5"
local HUEUSERNAME = "***************"
local HUELIGHTNR = 5
local HUELIGHT5IDX = 73

return {
  on = {
    devices = { HUELIGHT5IDX },
     },
  execute = function(domoticz, item)
    if (item.isDevice) then
      -- curl --request PUT --data "{\"on\":true}" http://hue-bridge-ip/api/username/lights/5/state
      local switchState = true
      if (domoticz.devices(HUELIGHT5IDX).state == "Off") then
        switchState = false
      end
      local curlRequest = string.format('curl --request PUT --data "{\\"on\\":%s}"
http://%s/api/%s/lights/%s/state',switchState,HUEBRIDGEIP,HUEUSERNAME,HUELIGHTNR)
      domoticz.log(curlRequest)
      local handle = os.execute(curlRequest)
      domoticz.log(handle)
    end
  end
}
```

HTTP - Node-RED

Example Node-RED flow to switch or set the brightness of Hue 5.

Source: hue_direct_control_example.flow

Switch Hue 5
Flow nodes: inject > function > http request > debug

Inject Node Set Payload – the message body

```
{“on”:true}
OR
{“on”:false}
```

Function Node Set URL Change State

```
// Set the url message
// Example: http://hue-bridge-ip/api/username/lights/lightnr/state
const urlBridge = “http://hue-bridge-ip/api/”;
const 146sernamе = “username”;
const lightNr = 5;
msg.url = urlBridge + 146sernamе + “/lights/” + lightNr + “/state”;
node.warn(msg.url);
return msg;
```

HTTP Request Node
Method PUT
URL is set by the previous function node.

Debug Node
Log the HTTP request result.
Example switch light off

```
[{“success”:{“/lights/5/state/on”:false}}]
```

Node-RED Brightness Hue 5

Basically as previous, but the message body, which is the payload, with JSON properties on = true and bri = 76.

```
{"on":true, "bri":76}
```

with HTTP result

```
[{"success":{"/lights/5/state/on":true}},{"success":{"/lights/5/state/bri":76}}]
```

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 chapter Explore RaspberryMatic.

The hardware used is a *homematicIP Window and Door Contact – optical* (HMIP-SWDO, Product ref: 140733A09) and Metal Postbox with dimensions (W x H x D) 35 x 40 x 10 cm. The postbox is ~40 m distance from the CCU (which is in range of 300 m according HMIP-SWDO datasheet).

Hint
The device HmIP-SWDO is light sensitive, means if its dark then there might be no signal. If that's an issue, use the device HmIP-SWDM = window / door contact which magnet.

Whilst evolving, several solutions have been developed. The third solution is in place.

Solution	**Description**
Alert	If the postbox is opened, set the Alert message and send email notification (no manual intervention required).
Alert + Switch	As previous plus functionality: Turn a switch on, indicating the postbox has been opened. Manual turn the switch to off after checking the postbox. This solution handles the situation where the postbox latch is kept open instead of closing after post has been dropped. If the latch remains open, the homematicIP SWDO device keeps on sending messages. This is avoided by sending a message onl once and manual reset the switch after the postbox has been checked.
Alert + Switch + Voltage **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 system.

Solution Alert

Communication Flow

1. HMIP-SWDO detects opening the hatch of the postbox.
2. HMIP-SWDO triggers a script which submits a HTTP API Request to the Domoticz n system to update Alert Message device to Level 4 with message “Postbox opened”.
3. Domoticz updates the device and sends out Email notification as triggered by comparing Alert Level 4 against user variable TH_ALERTTOEMAIL (which has integer value 4).

Issue
If the hatch remains open, the HMIP-SWDO keeps on sending alert messages. To avoid this, the alert message script checks if an alert is sent again within a short timeframe threshold, but

- in case other alert messages are received, these are not captured and
- also if the threshold is reached, alerts are still sent.

Therefor option two is developed which requires manual intervention.

RaspberryMatic CCU Configuration

Step 1
The device HMIP-SWDO is added to the CCU following the Teach-in devices procedure (used entering the Key and SGTIN data).

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
```

RaspberryMatic Script

The script makes use of a CUxD device (see Addon CUxD) to run system commands.

The system command, which triggers a HTTP API Request, to update the Domoticz Alert Message device (IDX=55) with Alert Level 4 (nvalue) and the message (svalue):

```
wget -q -O - 'http://domoticz-
ip:8080/json.htm?type=command&param=udevice&idx=IDX&nvalue=4&svalue=MESSAGE'
```

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

Script Code

RaspMatic Script: postbox_notifier.script

```
string sFunction = "Postbox Alert";
string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! A domoticz alert ensor is used
string nIdx = 55; ! Type: General; SubType: Alert
string nLevel = 4;  ! Red
string sText = sDate # "%20" # sTime # "%20-%20Postbox%20opened!";
! Build the domoticz HTTP API Request url
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm"; ! Production
string sUrl =
sDomUrl#"?type=command"#cAmp#"param=udevice"#cAmp#"idx="#nIdx#cAmp#"nvalue="#nLevel#cAmp#"svalue="#sTex
t#"'";

! 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 uservarable 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.prevtimenotified) > 60) then
      domoticz.notify(device.text, '' , domoticz.PRIORITY_HIGH)
      domoticz.data.prevtimenotified = os.time()
    end
  end
}
```

Note

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

Solution Alert + Switch

Communication Flow

1. HMIP-SWDO detects opening the hatch of the postbox.
2. HMIP-SWDO triggers script which submits HTTP API Request to Domoticz system to set the state of the switch “Postbox Notifier” (idx: 189) to “On” and sends out an Alert Message
    a. subject “Postbox opened (HH:MM)”
    b. body “Ensure to reset the switch Postbox Notifier.”
3. The switch “Postbox Notifier” is set to “Off” manually.

Example

RaspberryMatic CCU Configuration

Step 1
The device HMIP-SWDO is added to the CCU following the Teach-in devices procedure (used entering the Key and SGTIN data).

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:8080/json.htm?type=command&param=switchlight&idx=IDX&switchcmd=CMD’
```

Note
The switch command “CMD” is case sensitive: On | Off.

The CUxD device used is “CUxD (28) System” with Function Exec – Remote Control with 19 keys = key :1 is used.

Script Code

```
! Function: Postbox Notifier, 20191110 by rwbl
string sFunction = "Postbox Notifier";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
! string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";

! A domoticz alert sensor is used
string nIdx = 189; ! Type: Light/Switch; SubType: On/Off
string sOn = "On";
string sText = "Postbox%20opened%20(" # sDate # "%20" # sTime # ")";

! Build the domoticz HTTP API Request url
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm"; ! Production
string sUrl =
sDomUrl#"?type=command"#cAmp#"param=switchlight"#cAmp#"idx="#nIdx#cAmp#"switchcmd="#sOn#"'";
WriteLine(sUrl);

! OPTION: Run the command without a return result
! res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);

! OPTION: Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: The script running on the CCU waits until the completion - ensure not to execute commands which
take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").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.

Create Virtual Sensor
Name: Postbox Notifier Voltage
Sensor Type: Voltage
OK Cancel

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.

Domoticz Log Entries

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

For tests running automation script every minute

```
2019-12-11 18:24:00.942 Status: dzVents: Info: Handling httpResponse-events for: "RESPOSTBOXNOTIFIER
2019-12-11 18:24:00.942 Status: dzVents: Info: ------ Start internal script: postbox_notifier: HTTPResponse: "RESPOSTBOXNOTIFIER"
2019-12-11 18:24:00.942 Status: dzVents: Info: Postbox Notifier Voltage:Voltage=1.5
2019-12-11 18:24:00.959 Status: dzVents: Info: ------ Finished postbox_notifier
```

RaspMatic HTTP XML-API request operating voltage

```
2019-12-11 18:26:51.854 (VirtualSensors) Light/Switch (Postbox Notifier State)
2019-12-11 18:26:51.841 Status: User: Admin initiated a switch command (189/Postbox Notifier State/On)
2019-12-11 18:26:51.861 Status: Incoming connection from: 192.168.1.225
2019-12-11 18:26:52.061 Notification sent (http) => Success
2019-12-11 18:26:52.032 Status: dzVents: Info: Handling events for: "Postbox Notifier State", value: "On"
2019-12-11 18:26:52.032 Status: dzVents: Info: ------ Start internal script: postbox_notifier: Device: "Postbox Notifier  State (VirtualSensors)", Index: 189
2019-12-11 18:26:52.033 Status: dzVents: Info: Device Postbox Notifier  State changed On = 0
2019-12-11 18:26:52.034 Status: dzVents: Info: POSTBOXNOTIFIER Postbox Notifier notification sent: Postbox opened (18:26) (Postbox Notifier: Ensure to reset switch.)
2019-12-11 18:26:52.034 Status: dzVents: Info: ------ Finished postbox_notifier
```

Postbox opened. RaspMatic scripts sets Domoticz switch to On

```
2019-12-11 18:28:30.907 (VirtualSensors) Light/Switch (Postbox Notifier State)
2019-12-11 18:28:30.903 Status: User: Admin initiated a switch command (189/Postbox Notifier State/Off)
2019-12-11 18:28:31.009 Status: dzVents: Info: Handling events for: "Postbox Notifier State", value: "Off"
2019-12-11 18:28:31.009 Status: dzVents: Info: ------ Start internal script: postbox_notifier: Device: "Postbox Notifier  State (VirtualSensors)", Index: 189
2019-12-11 18:28:31.010 Status: dzVents: Info: Device Postbox Notifier  State changed Off = 1
2019-12-11 18:28:31.011 Status: dzVents: Info: ------ Finished postbox_notifier
```

Domoticz switch reset to off by user

Domoticz Roomplan & Dashboard

Dashboard

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://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
-- datapoint
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()  -- For Linux
local DATAPOINT2543 = JSON:decode('{"name":"Postbox Notifier
Voltage","id":2543,"xpath":"//datapoint[@ise_id=2543]/@value","response":"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 is switch is manually switched Off
   if (domoticz.devices(IDX_POSTBOXNOTIFIER_STATE).state == 'Off') and
      (domoticz.data.pbnnotified == 1) then
      domoticz.data.pbnnotified = 0
     end
   end

   -- check if the item is a device, then request datapoint information
   if (item.isTimer) then
     domoticz.openURL({url=URL_RASPMATIC..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 (RaspberryMatic)

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.

There are two solutions developed:
1. Advanced – Created a Domoticz Python Plugin with various devices
2. Simple – Domoticz thermostat selector switch device with a setpoint device

Solution Plugin

The homematicIP Radiator Thermostats HmIP-eTRV-B & HmIP-eTRV-2 are installed at several radiators in the house.

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

- Various Domoticz device timers are put in place to control the temperature depending the radiator location.
- If an open window is detected by the thermostat, the setpoint is decreased to 12 °C until the window is closed, than the setpoint is back to previous value.
- 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).
- Just as an experiment: a Node-RED Dasboard.

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

Installation

These steps apply for every Radiator Thermostat Device:

HomeMatic WebUI (RaspberryMatic):
Add the thermostat device to RaspberryMatic (Teach-in)

Download the HmIP-eTRV plugin repository from here

On the Domoticz system, create folder: /home/pi/domoticz/plugins/hmip-etrv

Copy the file plugin.py to the folder: /home/pi/domoticz/plugins/hmip-etrv

Restart Domoticz either via command line or GUI:

- sudo service domoticz.sh restart
- GUI > Setup > More Options > Restart System

The plugin requires the parameters:

- CCU IP Address
- Device id of the thermostat device HmIP-eTRV-B or HmIP-eTRV-2
- Datapoint id's for the selected device:
  SET_POINT_TEMPERATURE, ACTUAL_TEMPERATURE, LOW_BAT, LEVEL
  as comma separated list in this order

Note
Obtain the device & datapoint id's by requesting the device state list from the CCU via URL: http://ccu-ip/addons/xmlapi/statelist.cgi

Get the devices state list: http://ccu-ip-address/addons/xmlapi/statelist.cgi with XML tree result.

1. Select Device HMIP-eTRV-B with name "Thermostat Dining Room"
2. Get the Device ID **3637**
3. Get the Datapoints IDs for the properties:
   SET_POINT_TEMPERATURE=3680,ACTUAL_TEMPERATURE=3663,LOW_BAT=3645,LEVEL=3672

The ids are used to

1. get the actual temperature, battery low voltage status & level
2. set the thermostat setpoint.


```
device config_pending="false" unreach="false" ise_id="3637" name="Thermostat Dining Room">
  - channel ise_id="3638" name="Thermostat Dining Room:0" operate="true" visible="true" index="0">
      datapoint ise_id="3639" name="HmIP-RF.000A1A49A0D878:0.CONFIG_PENDING" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
      datapoint ise_id="3643" name="HmIP-RF.000A1A49A0D878:0.DUTY_CYCLE" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
      datapoint ise_id="3645" name="HmIP-RF.000A1A49A0D878:0.LOW_BAT" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="LOW_BAT"/>
      datapoint ise_id="3649" name="HmIP-RF.000A1A49A0D878:0.OPERATING_VOLTAGE" operations="5" timestamp="1577009479" valueunit="" valuetype="4" value="3.000000" type="OPERATING_VOLTAGE"/>
      datapoint ise_id="3650" name="HmIP-RF.000A1A49A0D878:0.OPERATING_VOLTAGE_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
      datapoint ise_id="3651" name="HmIP-RF.000A1A49A0D878:0.RSSI_DEVICE" operations="5" timestamp="1577009479" valueunit="" valuetype="8" value="206" type="RSSI_DEVICE"/>
      datapoint ise_id="3652" name="HmIP-RF.000A1A49A0D878:0.RSSI_PEER" operations="5" timestamp="1576990804" valueunit="" valuetype="8" value="200" type="RSSI_PEER"/>
      datapoint ise_id="3653" name="HmIP-RF.000A1A49A0D878:0.UNREACH" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="UNREACH"/>
      datapoint ise_id="3657" name="HmIP-RF.000A1A49A0D878:0.UPDATE_PENDING" operations="5" timestamp="1576940478" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
    </channel>
  - channel ise_id="3661" name="HmIP-eTRV-2 000A1A49A0D878:1" operate="true" visible="true" index="1">
      datapoint ise_id="3662" name="HmIP-RF.000A1A49A0D878:1.ACTIVE_PROFILE" operations="7" timestamp="1577009479" valueunit="" valuetype="16" value="1" type="ACTIVE_PROFILE"/>
      datapoint ise_id="3663" name="HmIP-RF.000A1A49A0D878:1.ACTUAL_TEMPERATURE" operations="5" timestamp="1577009479" valueunit="" valuetype="4" value="22.700000" type="ACTUAL_TEMPERATURE"/>
      datapoint ise_id="3664" name="HmIP-RF.000A1A49A0D878:1.ACTUAL_TEMPERATURE_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
      datapoint ise_id="3665" name="HmIP-RF.000A1A49A0D878:1.BOOST_MODE" operations="6" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="BOOST_MODE"/>
      datapoint ise_id="3666" name="HmIP-RF.000A1A49A0D878:1.BOOST_TIME" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="BOOST_TIME"/>
      datapoint ise_id="3667" name="HmIP-RF.000A1A49A0D878:1.CONTROL_DIFFERENTIAL_TEMPERATURE" operations="2" timestamp="0" valueunit="" valuetype="4" value="" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
      datapoint ise_id="3668" name="HmIP-RF.000A1A49A0D878:1.CONTROL_MODE" operations="2" timestamp="0" valueunit="" valuetype="16" value="" type="CONTROL_MODE"/>
      datapoint ise_id="3669" name="HmIP-RF.000A1A49A0D878:1.DURATION_UNIT" operations="2" timestamp="0" valueunit="" valuetype="16" value="" type="DURATION_UNIT"/>
      datapoint ise_id="3670" name="HmIP-RF.000A1A49A0D878:1.DURATION_VALUE" operations="2" timestamp="0" valueunit="" valuetype="16" value="" type="DURATION_VALUE"/>
      datapoint ise_id="3671" name="HmIP-RF.000A1A49A0D878:1.FROST_PROTECTION" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="FROST_PROTECTION"/>
      datapoint ise_id="3672" name="HmIP-RF.000A1A49A0D878:1.LEVEL" operations="7" timestamp="1577009479" valueunit="" valuetype="4" value="0.120000" type="LEVEL"/>
      datapoint ise_id="3673" name="HmIP-RF.000A1A49A0D878:1.LEVEL_STATUS" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="LEVEL_STATUS"/>
      datapoint ise_id="3674" name="HmIP-RF.000A1A49A0D878:1.PARTY_MODE" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="PARTY_MODE"/>
      datapoint ise_id="3675" name="HmIP-RF.000A1A49A0D878:1.PARTY_SET_POINT_TEMPERATURE" operations="5" timestamp="0" valueunit="" valuetype="4" value="0.000000" type="PARTY_SET_POINT_TEMPERATURE"/>
      datapoint ise_id="3676" name="HmIP-RF.000A1A49A0D878:1.PARTY_TIME_END" operations="7" timestamp="0" valueunit="" valuetype="20" value="" type="PARTY_TIME_END"/>
      datapoint ise_id="3677" name="HmIP-RF.000A1A49A0D878:1.PARTY_TIME_START" operations="7" timestamp="0" valueunit="" valuetype="20" value="" type="PARTY_TIME_START"/>
      datapoint ise_id="3678" name="HmIP-RF.000A1A49A0D878:1.QUICK_VETO_TIME" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="QUICK_VETO_TIME"/>
      datapoint ise_id="3679" name="HmIP-RF.000A1A49A0D878:1.SET_POINT_MODE" operations="7" timestamp="1577009479" valueunit="" valuetype="16" value="1" type="SET_POINT_MODE"/>
      datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1577009479" valueunit="°C" valuetype="4" value="21.000000" type="SET_POINT_TEMPERATURE"/>
      datapoint ise_id="3681" name="HmIP-RF.000A1A49A0D878:1.SWITCH_POINT_OCCURED" operations="5" timestamp="1577009479" valueunit="" valuetype="2" value="false" type="SWITCH_POINT_OCCURED"/>
      datapoint ise_id="3682" name="HmIP-RF.000A1A49A0D878:1.VALVE_ADAPTION" operations="7" timestamp="0" valueunit="" valuetype="2" value="false" type="VALVE_ADAPTION"/>
      datapoint ise_id="3683" name="HmIP-RF.000A1A49A0D878:1.VALVE_STATE" operations="5" timestamp="1577009479" valueunit="" valuetype="16" value="4" type="VALVE_STATE"/>
      datapoint ise_id="3684" name="HmIP-RF.000A1A49A0D878:1.WINDOW_STATE" operations="7" timestamp="1577009479" valueunit="" valuetype="16" value="0" type="WINDOW_STATE"/>
    </channel>
    channel ise_id="3685" name="HmIP-eTRV-2 000A1A49A0D878:2" operate="true" visible="true" index="2"/>
    channel ise_id="3686" name="HmIP-eTRV-2 000A1A49A0D878:3" operate="true" visible="true" index="3"/>
    channel ise_id="3687" name="HmIP-eTRV-2 000A1A49A0D878:4" operate="true" visible="true" index="4"/>
    channel ise_id="3688" name="HmIP-eTRV-2 000A1A49A0D878:5" operate="true" visible="true" index="5"/>
    channel ise_id="3689" name="HmIP-eTRV-2 000A1A49A0D878:6" operate="true" visible="true" index="6"/>
    channel ise_id="3690" name="HmIP-eTRV-2 000A1A49A0D878:7" operate="true" visible="true" index="7"/>
  </device>
```


Important
Prior adding new hardware, ensure Domoticz accepts new hardware as devices will be created when adding the plugin.
Check: GUI > Setup > Settings > Hardware
Either mark accept or click Allow for 5 minutes

Goto GUI > Setup > Hardware

Select Type: homematicIP Radiator Thermostat (HMIP-eTRV)

Give a name, i.e. Thermostat Dining

Set initially the parameter: Keep debug to true
Click button Add
Check the Domoticz log if the four devices are created and updated: Thermostat Dining - Setpoint,Thermostat Dining - Temperature,Thermostat Dining - Battery,Thermostat Dining - Valve
Check the device widgets on the GUI tabs.
If all OK, set the hardware parameter Debug to false
Optional steps (see also screenshots below): • Add the devices to a room plan, i.e Dining Room • Add a device timer to ensure the thermostat is turned Off to avoid heating overnight • Add a switch device to turn the thermostat off
Notes If adding device timers for the setpoint, set the thermostat to “Manu mode” in the HomeMatic WebUI (RaspberryMatic) else if “Auto mode” is set, the selected Week profile is used.

Add new Hardware (GUI > Setup > Hardware)

Devices, Roomplan & Dashboard for selected room

Enhancement Push Off Button

Purpose
To define a switch to handle the thermostat device state change action to Off.

How
Define a VirtualSensor device with properties:

- Name, i.e. "MakeLab Thermostat - Off"
- Sensor Type Switch.

After adding the device, select the device widget and change the properties:

- Switch Type:
  Push Off Button
- Off Action:
  http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=0

The datapoint SET_POINT_TEMPERATURE of the device is required, i.e. 1584. Obtain this datapoint id (ise_id), by requesting the state of the thermostat device with id=1541. The HTTP XML response holds various datapoints, select: SET_POINT_TEMPERATURE.

```
http://raspmatic-ip/addons/xmlapi/state.cgi?device_id=1541
```

```
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" timestamp="1576829294"
valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
```

Domoticz Log Example

Performed test by pushing the icon on the widget, which send the HTTP XML-API request to the RaspberryMatic system to set the setpoint of the MakeLab thermoststat to 0.

```
2019-12-15 18:30:06.767 (VirtualDevices) Light/Switch (MakeLab Thermostat - Off)
2019-12-15 18:30:06.758 Status: User: Admin initiated a switch command (72/MakeLab Thermostat -
Off/Off)
```

Timers

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

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

```
SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID;
```

Output Tool SQLiteSpy

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

Examples SQLite Shell

The header (.header on) and mode CSV (.mode csv) are set.
More information, read chapter SQLite Shell.

Full Overview

```
sqlite> SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID ORDER BY
d.name;
```

```
Name,ID,Active,DeviceRowID,Date,Time,Type,Temperature,TimerPlan,Days,Month,MDay,Occurence
"Hzg Bad Sollwert",1,1,207,2019-12-17,06:00,2,21.0,0,128,0,0,0
"Hzg Bad Sollwert",2,1,207,2019-12-15,11:00,2,19.0,0,128,0,0,0
"Hzg Bad Sollwert",3,1,207,2019-12-17,19:00,2,21.0,0,128,0,0,0
"Hzg Bad Sollwert",11,1,207,2019-12-17,23:00,2,17.0,0,128,0,0,0
"Hzg Dusche Sollwert",8,1,198,2019-12-14,21:30,2,21.0,0,128,0,0,0
"Hzg Dusche Sollwert",9,1,198,2019-12-14,23:00,2,0.0,0,128,0,0,0
"Hzg EZ Sollwert",12,1,210,2019-12-21,22:00,2,17.0,0,128,0,0,0
…and more …
```

Brief Overview

```
sqlite> SELECT d.Name, t.Time, t.Temperature FROM DeviceStatus d, SetpointTimers t WHERE
d.ID=t.DeviceRowID ORDER BY d.name,t.Time;
```

```
Name,Time,Temperature
"Hzg Bad Sollwert",06:00,21.0
"Hzg Bad Sollwert",11:00,19.0
… and more
```

Example Device Timer Definition

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.

Solution Selector Switch & Thermostat

This is a simple solution making use of a selector switch to select from the list of thermostats. The setpoint for the selected thermostat is set by using a thermostat device.
Communication between Domoticz and RaspberryMatic using the RaspberryMatic XML-API CCU Addon.

Domoticz Configuration

Selector Style: Select Menu; Hide Off level on.

Select Thermostat & Setpoint change

Thermostat Devices

Get all devices defined in RaspberryMatic CCU by sending an HTTP XML-API request to the CCU with parameter the script “statelist.cgi”.

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

From the HTTP XML response, only the datapoint type SET_POINT_TEMPERATURE is required as used to control the thermostat, gain, via HTTP XML-API requests with script as parameter.
(extract from the statelist request)


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="3435" name="Thermostat Bad" config_pending="false" unreach="false">
<datapoint ise_id="3478" name="HmIP-RF.002018A99D2097:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144602" valueunit="°C" valuetype="4"
value="21.000000"/>
</device>
<device ise_id="3489" name="Thermostat Dusche" config_pending="false" unreach="false">
<datapoint ise_id="3532" name="HmIP-RF.00201A49952CB8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588139539" valueunit="°C" valuetype="4"
value="6.000000"/>
</device>
<device ise_id="3637" name="Thermostat Esszimmer" config_pending="false" unreach="false">
<datapoint ise_id="3680" name="HmIP-RF.000A1A49A0D878:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144916" valueunit="°C" valuetype="4"
value="21.000000"/>
</device>
<device ise_id="3691" name="Thermostat Flur" config_pending="false" unreach="false">
<datapoint ise_id="3734" name="HmIP-RF.000A1A49A0D8A5:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144818" valueunit="°C" valuetype="4"
value="21.000000"/>
</device>
<device ise_id="1541" name="Thermostat MakeLab" config_pending="false" unreach="false">
<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144325" valueunit="°C" valuetype="4"
value="20.000000"/>
</device>
<device ise_id="3543" name="Thermostat Wohnzimmer-1" config_pending="false" unreach="false">
<datapoint ise_id="3586" name="HmIP-RF.00201A49952CB1:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144907" valueunit="°C" valuetype="4"
value="12.000000"/>
</device>
<device ise_id="3375" name="Thermostat Wohnzimmer-2" config_pending="false" unreach="false">
<datapoint ise_id="3418" name="HmIP-RF.00201A499D76F8:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" operations="7" timestamp="1588144908" valueunit="°C" valuetype="4"
value="21.000000"/>
</device>
</stateList>
```


GET the thermostat SET_POINT_TEMPERATURE

```
http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=DATAPOINT_ISE_ID
```

SET the thermostat SET_POINT_TEMPERATURE

```
http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=DATAPOINT_ISE_ID
```

The datapoints are defined in a dzVents Lua Automation Script.

Automation Script

For the dzVents Lua automation script, the RaspberryMatic Thermostat datapoint ise_id for the type SET_POINT_TEMPERATURE, is required.
The datapoints are used to get or set the setpoint.
In dzVents an array is defined according to the thermostats defined in the selector switch:

```
[level]=ise_id SET_POINT_TEMPERATURE
```

```
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418 , [40]=3680, [50]=3478, [60]=3734,
[70]=3532};
```

Name	Ise_id	Level	Level Name	
Thermostat MakeLab	1584	10	MakeLab	
Thermostat Wohnzimmer-1	3586	20	Wohn-1	
Thermostat Wohnzimmer-2	3418	30	Wohn-2	
Thermostat Esszimmer	3680	40	Essen	
Thermostat Bad	3478	50	Bad	
Thermostat Flur	3734	60	Flur	
Thermostat Dusche	3532	70	Dusche	

IMPORTANT: if the level order is changed in the device widget, the array must be changed as well!

Event name: thermostat_control_selector_switch.dzvents
(below code is stripped)

```
local IDX_THERMOSTAT_SELECTOR = 114, IDX_THERMOSTAT_SETPOINT = 115;
local ID_THERMOSTAT_DATAPOINTS = {[10]=1584, [20]=3586, [30]=3418 , [40]=3680, [50]=3478, [60]=3734,
[70]=3532};
local datapointSelected;
local URL_RASPMATIC_STATECHANGE = 'http://ccu-ip/config/xmlapi/statechange.cgi?ise_id=‘,
URL_RASPMATIC_STATE = 'http://ccu-ip/config/xmlapi/state.cgi?datapoint_id=';
local RES_RASPMATIC_SETSETPOINT = 'res_raspmatic_setsetpoint’, RES_RASPMATIC_GETSETPOINT =
'res_raspmatic_getsetpoint';
local DECIMALS = 1;
function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end
return {
  on = {devices = {IDX_THERMOSTAT_SELECTOR, IDX_THERMOSTAT_SETPOINT},
        httpResponses = {RES_RASPMATIC_SETSETPOINT,RES_RASPMATIC_GETSETPOINT,}},
        data = {updateraspmatic = { initial = 1 }},
        execute = function(domoticz, item)
          if (item.isDevice) then
            datapointSelected = ID_THERMOSTAT_DATAPOINTS[domoticz.devices( IDX_THERMOSTAT_SELECTOR
).level];
            if (item.idx == IDX_THERMOSTAT_SELECTOR) then
              domoticz.data.updateraspmatic = 0;
              local urlrm = string.format("%s%d", URL_RASPMATIC_STATE, datapointSelected);
              domoticz.openURL({url = urlrm, method = 'POST', callback = RES_RASPMATIC_GETSETPOINT});
            end
            if (item.idx == IDX_THERMOSTAT_SETPOINT) then
              if (domoticz.data.updateraspmatic == 1) then
                local newsetpoint = round(item.setPoint,DECIMALS);
  local urlrm = string.format("%s%d&new_value=%.2f", URL_RASPMATIC_STATECHANGE, datapointSelected,
newsetpoint);
     domoticz.openURL({url = urlrm, method = 'POST', callback = RES_RASPMATIC_SETSETPOINT});
              end
              if (domoticz.data.updateraspmatic == 0) then domoticz.data.updateraspmatic = 1; end
            end
          end
```

```
          if (item.isHTTPResponse) then
            if (item.statusCode == 200) then
              if (item.callback == RES_RASPMATIC_GETSETPOINT) then
                datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level];
                local setpoint = domoticz_applyXPath(item.data,'//datapoint[@ise_id="' ..
datapointSelected .. '"]/@value');
                setpoint = round(setpoint, DECIMALS);
                local urldom = string.format("http://domoticz-
ip:8080/json.htm?type=command&param=setsetpoint&idx=%d&setpoint=%.2f",IDX_THERMOSTAT_SETPOINT,setpoint)
;
                domoticz.openURL({url = urldom, method = 'POST'});
              end
              if (item.callback == RES_RASPMATIC_SETSETPOINT) then datapointSelected =
ID_THERMOSTAT_DATAPOINTS[domoticz.devices(IDX_THERMOSTAT_SELECTOR).level]; end
            else
              domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
            end
        end
    end
}
```

Raspberry Pi Monitor

Purpose

To measure & show specific Raspberry Pi system information and trigger alarm if a value is above a certain threshold.
Monitored are: Memory Usage, Hard Disc Space, Temperature.

Solution

Use the Domoticz Hardware Motherboard, which will create several devices. The devices can be used to trigger events.

Hardware Motherboard

Devices derived from Hardware Motherboard

Idx	Hardware	ID	Unit	Name	Type	SubType	
1	Motherboard	0000044C	1	RPi Memory Usage	General	Percentage	20.93%
2	Motherboard	0000044E	1	HDD /boot	General	Percentage	1.5%
3	Motherboard	0000044F	1	RPi HDD	General	Percentage	32.32%
4	Motherboard	0001	1	RPiTemp	Temp	LaCrosse TX3	44.0 C
5	Motherboard	0000044D	1	CPU_Usage	General	Percentage	2.84%

Note
Not all devices are added, (only) RPi Memory Usage, RPi HDD, RPi Temp.

Widgets (Tabs Utility & Temperature)

Monitoring

To monitor selective devices and notify if above threshold.
The notification is done using the Virtual Device Alert Message (idx=55).

An email notification is 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 added to 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.

Roomplans

Purpose

To define roomplans containing all devices. The roomplans are used to control the rooms via the GUI, but also to organize the devices in a proper way.
To access a device, a roomplan should be selected.

The GUI Dashboard contains (only) a few devices for a quick lookup.

Define Roomplan

The roomplans are defined via the GUI > Setup > More options > Plans.
Example list of roomplan and for a room with devices.

Idx	Name
1	$Hidden Devices
13	Bad
14	Dusche
11	Esszimmer (EZ)
18	Flur
16	Haus
2	Information
17	Keller
15	MakeLab
19	Schlafzimmer (SZ)
4	Termine
12	Wohnzimmer (WZ)
8	Hardware Monitor
5	Volumio

Idx	Name
12	Wohnzimmer (WZ)

Showing 1 to 1 of 1 entries (filtered

Edit Delete

Devices
(Select Plan first to Edit...)

Show 10 entries

Idx	Name
112	Hue WZ TV
185	Hue WZ Stehlampe
113	Hue WZ Rob
40	WZ Temperatur
201	Hzg WZ-1 Sollwert
202	Hzg WZ-1 Temperatur
203	Hzg WZ-1 Batterie
204	Hzg WZ-2 Sollwert
205	Hzg WZ-2 Temperatur
206	Hzg WZ-2 Batterie
12	Markise WZ

Showing 1 to 11 of 11 entries

Notes
The definition of the roomplans is not completed yet.
Still thinking about how to organize in a simple way, do not have too many rooms and the naming convention of the roomplan (which is used for defining the device names = see next).
This also relates for defining floorplans.

Define Device Names

Define naming convention and organise device according roomplans.

Rules

- Device naming convention: Hardware Roomplan Device.
- A roomplan gets an abbreviation, max 3 characters, if the roomplan name is longer than 7 characters.
- Domoticz production system: all device names translated to german.
- Domoticz development system: all device names in english.
- Define roomplans and associate devices.
- All devices must be associated with a roomplan.

Examples_(Hardware, Roomplan, Device)

- Hue Light located in the living room near TV.
  Roomplan: Wohnzimmer (WZ).
  Device name: Hue WZ TV
- Thermostat dining room with three devices setpoint,temperature, battery.
  Roomplan: Esszmmer (EZ).
  Device name: Hzg EZ Sollwert, Hzg EZ Temperatur, Hzg EZ Batterie

Overview Roomplans

There is no complete overview of the roomplans defined with devices associated.

An option to get the list of roomplans and devices, is using the SQLite command-line shell (see SQLite Shell) to get the information from the Domoticz database by direct accessing the tables:

- **Database**
  /home/pi/domoticz/domoticz.db

- **Tables**
  - DeviceStatus – get the device name for a given ID (=Idx)
  - Plans – get the roomplans
  - DeviceToPlansMap – get the roomplans with associated devices

The SQLite command-line shell is opened with

```
cd /home/pi/domoticz
sqlite3
```

Plans

Get the roomplans

The output is in order of appearance.

```
sqlite> .header on
sqlite> .mode column
sqlite> select * from plans order by "Order";
ID          Order       Name             FloorplanID  Area
----------  ----------  ---------------  -----------  ----------
1           1           $Hidden Devices  0
13          4           Bad              0
14          5           Dusche           0
11          7           Esszimmer (EZ)   0
16          8           Haus             0
2           9           Information      1
17          10          Keller           0
15          11          MakeLab          0
4           12          Termine          0
12          13          Wohnzimmer (WZ)  0
8           17          Hardware Monito  0
5           18          Volumio          3            123,118
```

Note
Screenshot of the GUI with the dropdown list showing the roomplans.

Get the roomplans

List the plans with their associated devices. The output format is CSV.

If the device name contains blanks, the name is put between "", i.e. Device Name"

```
cd /home/pi/domoticz
sqlite3

sqlite> .open domoticz.db
sqlite> .header on
sqlite> .mode csv
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan,PlanID,Device,Idx
Bad,13,"Hzg Bad Batterie",209
Bad,13,"Hzg Bad Sollwert",207
Bad,13,"Hzg Bad Temperatur",208
Dusche,14,"Hzg Dusche Batterie",200
Dusche,14,"Hzg Dusche Sollwert",198
Dusche,14,"Hzg Dusche Temperatur",199
"Esszimmer (EZ)",11,"Hzg EZ Batterie",206
"Esszimmer (EZ)",11,"Hzg EZ Sollwert",204
"Esszimmer (EZ)",11,"Hzg EZ Temperatur",205
"Esszimmer (EZ)",11,"Hue EZ",111
… and more …
```

RFXCOM RFXtrx433E

Purpose

The RFXCOM RFXtrx433E Transceiver is a 433Mhz transmitter and receiver (transceiver) to control or respond to 433Mhz devices (sensors).
More Product information here (including successor products).

Prepare RFXtrx433E

Install the utility programs RFXflash and RFXmngr on the Development Device.
Note
For upgrading the firmware, the RFX Flash Programmer is required. Always download the latest version prior flashing.

Flash latest firmware (RFXtrx433_Ext2_Firmware) as described in RFXtrx User Guide.
Note
Use the same procedure for updating the firmware, i.e.

Steps to install/upgrade

- Shutdown the Domoticz system (i.e. Raspberry Pi)
- Connect the RFXCOM device to USB port of development device (used COM7 on a notebook)
- Download and unzip the latest firmware
- Download and start the RFXCOM Flash Programmer
- Select the latest hex file
- Write to the device
- Connect the RFXCOM device to the Domoticz system
- Reboot the Domoticz system
- Open Domoticz in a browser and check the log (Setup > Log) and if the hardware device (Setup > Hardware) RFXtrx433e shows the correct version (in this case Version: Ext2/1025)
- In case issues, restart Domoticz from terminal:
  sudo service domoticz.sh restart

Example screenshot flashing firmware version 1025.

Open the RFXmngr, connect to the RFXtrx433E, obtain status information and log incoming messages for devices found.

Example RFXmngr Log

```
Get Status
-----------------------------------------------
Packettype        = Interface Message
subtype           = Interface Response
Sequence nbr      = 1
response on cmnd  = Get Status
Transceiver type  = 433.92MHz
Firmware version  = 1022
Firmware Type     = Ext2
Transmit power    = 10dBm
Hardware version  = 1.2
...
```

Depending devices found, incoming **messages** are logged, i.e. for a TFA TS34C (Temperature & Humidity sensor)

```
30.12.2017 13:46:01
Packettype    = TEMP_HUM
subtype       = TH7 - Cresta, TFA TS34C
                channel 1
Sequence nbr  = 102
ID            = 280E decimal:10254
Temperature   = 17,3 °C
Humidity      = 54
Status        = Comfortable
Signal level  = 5  -80dBm
Battery       = OK
```

Note
The RFXmngr is not only used to log connected devices but also to manually configure connected devices, like the Somfy RTS Devices.

Domoticz Configuration

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

4	RFXtrx433e	Yes	RFXCOM - RFXtrx433 USB 433.92MHz Transceiver Version: Ext2/1022 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 cmnd = Get Status
2018-08-30 11:27:52.311 Transceiver type = 433.92MHz
2018-08-30 11:27:52.311 Firmware version = 1022
2018-08-30 11:27:52.311 Firmware type = Ext2
2018-08-30 11:27:52.311 Hardware version = 1.2
2018-08-30 11:27:52.311 Undec off
2018-08-30 11:27:52.311 X10 enabled
2018-08-30 11:27:52.311 ARC enabled
2018-08-30 11:27:52.311 AC enabled
2018-08-30 11:27:52.311 HomeEasy EU enabled
2018-08-30 11:27:52.311 Meiantech/Atlantic enabled
2018-08-30 11:27:52.311 Oregon Scientific enabled
2018-08-30 11:27:52.311 ATI/Cartelectronic 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 Monitor (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

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 replacd 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

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
}
```

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 previous described solution “Automation Script”.

Outlined is a Node-RED solution for two XETRA stock quotes.

The stock quote information is requested, every 30 minutes, via an Alpha Vantage API request (URL), which returns a JSON string with price information etc. The symbol and price are extracted and send to the respective Domoticz device.

Example API Request & Response

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

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	Propery 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.

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

Notifications

Optional feature to notify incase a stock price is above or below a threshold.

Option Widget

Notify by using the device configuration to set the threshold for a stock and notify via email.

Type	When	Active Systems	Custom Message
Usage	Greater 28	email	Stock Name 28 EUR!

To change the threshold, modify the rule.

Option dzVents

Notify by using User Variables and a dzVentz 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).

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			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

Time Control

Purpose

- To track & control the time spent in hours, per activity block (with start- & end-time) and total/day, on a generic task.
- To set a threshold and notify if the time spent reached daily limit.

Note
The author wanted to get a grip on Make activities time spent with a limit of 2.5 to 4 hours/day.

Screenshots devices, dashboard and a simple graph

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
106	VirtualDevices	00082106	1	Time Control Status	General	Text	18:21 Update: Block=0.32, Day=0.60, Limit=2.50 hrs
102	VirtualDevices	000140B6	1	Time Control Today	Light/Switch	Switch	On
101	VirtualDevices	000140B5	1	Time Control	Light/Switch	Switch	On

Time Control running (On) with status update.

Time Control stopped (Off) with status showing all todays activity blocks with summary. Triggered by switch Time Control Today.

User Variable threshold max time spent/day

Idx	Variable name	Variable type	Current value
2	TH_TIMECONTROL	Float	2.5

Solution

This solution uses a

- **Switch device** ("Time Control") - start & stop time tracking
- **Text device** ("Time Control Status") - provide information on time spent per activity block and day
- **Switch device** ("Time Control Today") - enables to show todays activity blocks in the text device "Time Control Status".
- **User Variable** ("TH_TIMECONTROL") – set the threshold of max hours/day (float) which is checked in the dzVents Lua Automation Script "time_control.lua", function "checkLimit".
- **dzVents Lua Automation Script**: time_control.lua – start, update, stop with status update.
- **dzVents Lua Automation Script**: time_control_today.lua – display daily summary.
- **SQL-Command-File** – select & report the daily activity blocks with summary.

The time between tracking start & end is defined as a block. A day has one or more blocks (or empty if no action).
The switch device bar chart log provides a nice overview on the time spent.
A dzVents Lua Automation Script runs every minute and updates the text device with the time spent on the current block, day total and daily limit in hours.
The daiy limit can be set via a user variable. If not defined or value=0, the daily limit is not used.

Domoticz Configuration

Devices

- Type: Light/Switch, SubType:Switch,SwitchType: On/Off, Name: Time Control, Idx: 101, Add to dashboard.
- Type: General, SubType:Text, Name: Time Control Status, Idx: 106, Add to dashboard.
- Type: Light/Switch, SubType:Switch,SwitchType: On/Off, Name: Time Control Today, Idx: 102, Add to dashboard.

User Variable

- Variable name: TH_TIMECONTROL, Variable type: Float, Variable value: 2.5

Dashboard & Log

The three devices are added to the Domoticz dashboard or to a dedicated roomplan.
The switch "Time Control" is used to start & stop time tracking and the text device provides information when the actitity starts, updates or stops. The update interval is every minute.

Example information:

```
10:52 Stopped: Block=1.43, Day=1.43, Limit=2.50 hrs
11:03 Started: Block=0.00, Day=1.43, Limit=2.50 hrs
11:05 Update: Block=0.03, Day=1.47, Limit=2.50 hrs
```

Note
There are 2 digits used to display the values. This can be changed in the dzVents Lua script "time_control.lua".

To view the time spent, select GUI > Switches > Widget "Time Control" and click Log.

- "Today" shows, a bar chart, state On or Off (Y-axis) and time On or Off (X-axis).
- "Text log" lists the state (Data) with time (Date) and the User.

Automation Events

There are two dzVents Lua automation events:

- **Time Control**: runs every minute and updates the text device "Timer Control Status". At midnight, the daily information is resetted.
  (time_control.lua)
- **Time Control Today**: runs when triggered by switch "Time Control Today" and updates the text device "Timer Control Status" with daily summary.
  The data is selected using sqlite command-line with an external file "time-control-today.sql" (located in folder "/home/pi/domoticz/scripts/dzVents/scripts/" containing the SQL statementss.
  (time_control_today.lua)

Source: time_control.lua

```
-- Idx of the devices used
IDXTRACK = 101          -- switch on/off
IDXSTATUS = 106         -- text status
IDXTHTIMECONTROL = 2    -- user var threshold time control (TH_TIMECONTROL)

-- The value to increment on every timer update: minutes/60 to get an hour.
-- Examples: timer every minute: 1/60, every 5 minutes = 5/60
INCREMENTVALUE = (1/60)

function roundNumber(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

function roundsValue(svalue)
    return roundNumber(tonumber(svalue),2)
end

function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

function isnowhhmm(domoticz)
    local timearray = splitstring(domoticz.time.rawTime, ':')
    return timearray[1] .. ':' .. timearray[2]
end

function checkLimit(domoticz)
    local msg
    -- get the latest value of the user var because it might have changed
    domoticz.data.daylimit = domoticz.variables(IDXTHTIMECONTROL).value
  if (domoticz.data.daylimit == 0) or
     (domoticz.variables(IDXTHTIMECONTROL).value == nil) then
      return
    end
    -- check if dayvalue beyond daily limit and if not already notified
```

```
  if (domoticz.data.dayvalue > domoticz.data.daylimit) and
     (domoticz.data.notified == 0) then
      msg = string.format('%s: Day=%.4f, Daily threshold reached %.2f hrs',
            isnowhhmm(domoticz),
          domoticz.data.dayvalue,
          domoticz.data.daylimit)
      domoticz.notify('Time Control', msg, domoticz.PRIORITY_LOW)
      domoticz.log(msg, domoticz.LOG_INFO)
      domoticz.data.notified = 1
      return
  end
    -- check daylimit increased, i.e made a change whilst already notified. if so, reset notified
  if (domoticz.data.dayvalue < domoticz.data.daylimit) and
     (domoticz.data.notified == 1) then
      domoticz.data.notified = 0
      msg = string.format('%s: Day=%.4f, Daily threshold reset %.2f hrs',
            isnowhhmm(domoticz),
          domoticz.data.dayvalue,
          domoticz.data.daylimit)
      domoticz.log(msg, domoticz.LOG_INFO)
  end
end

function updateStatus(domoticz, action)
    checkLimit(domoticz)
  local msg = string.format('%s %s: Block=%.2f, Day=%.2f, Limit=%.2f hrs',
          isnowhhmm(domoticz),
          action,
          domoticz.data.blockvalue,
          domoticz.data.dayvalue,
          domoticz.data.daylimit)
    domoticz.devices(IDXSTATUS).updateText(msg)
  domoticz.log(msg, domoticz.LOG_INFO)
end

function updateData(domoticz)
  domoticz.data.blockvalue = domoticz.data.blockvalue + INCREMENTVALUE
  domoticz.data.dayvalue = domoticz.data.dayvalue + INCREMENTVALUE
  updateStatus(domoticz, 'Update')
end

function setTrack(domoticz)
    if (domoticz.devices(IDXTRACK).state == 'Off') then
        updateStatus(domoticz, 'Stopped')
    else
      domoticz.data.blockvalue = 0
        updateStatus(domoticz, 'Started')
    end
end

function resetDay(domoticz)
  domoticz.data.blockvalue = 0
  updateStatus(domoticz, 'Day Reset')
  domoticz.data.dayvalue = 0
  domoticz.data.notified = 0
end

return {
    -- handle timer and swith device change on or off
    on = {
        timer = { 'every minute', 'at 00:15' },
        devices = { IDXTRACK }
    },

    data = {
        blockvalue = { initial = 0 },
        dayvalue = { initial = 0 },
        daylimit = { initial = 0 },
        notified = { initial = 0 }
    },

  execute = function(domoticz, item)
     if (item.isTimer) and
         (domoticz.time.matchesRule('every minute')) and
         (domoticz.devices(IDXTRACK).state == 'On') then
```

```
            updateData(domoticz)
            return
      end
      if (item.isTimer) and
            (domoticz.time.matchesRule('at 00:15')) then
            resetDay(domoticz)
            return
      end
      if (item.isDevice) and
            (item.idx == IDXTRACK) then
            setTrack(domoticz)
            return
      end
     end
}
```

Source: time_control_today.lua

```
IDXSWITCH = 102
IDXSTATUS = 106
SQLCMD = "sqlite3 /home/pi/domoticz/domoticz.db '.read /home/pi/domoticz/scripts/dzVents/scripts/time-
control-today.sql'"

function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

local function osExecute(cmd)
    local fileHandle     = assert(io.popen(cmd, 'r'))
    local commandOutput  = assert(fileHandle:read('*a'))
    local returnTable    = {fileHandle:close()}
    -- rc[3] contains returnCode
    return commandOutput,returnTable[3]
end

return {
  on = {devices = { IDXSWITCH },},
  execute = function(domoticz, item)
      if (item.isDevice) then
        if (domoticz.devices(IDXSWITCH).state == "On") then
            output,rt = osExecute(SQLCMD)
                 -- local handle = os.execute(sqlcmd)
                 rawdate = domoticz.time.rawDate .. " "
                 -- add the date to the output
                 output = string.format("%s\n%s",rawdate,output)
                 domoticz.log(output, domoticz.LOG_INFO)
                 -- update the status device
                  domoticz.devices(IDXSTATUS).updateText(output)
                 -- Returncode: 0=OK
                 domoticz.log(tostring(rt), domoticz.LOG_INFO)
                 -- convert the output to a table. Use #outputtable to get the number of entries.
                 outputtable = splitstring(output, "\n")
                 domoticz.log(outputtable, domoticz.LOG_INFO)
             end
    end
     end
}
```

Source: time_control_today.sql

```
.mode list
.separator " - "
.headers on
-- List the blocks
WITH rows AS
  ( SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
    FROM [lightinglog] WHERE [devicerowid] = 101 AND [date] >= date('now'))
SELECT strftime('%H:%M',startdata.date) AS Start, strftime('%H:%M',enddata.date) AS End,
    ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
FROM rows startdata
JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1
```

```
WHERE  startdata.nValue = 1;

-- Show the total
WITH rows AS
  (SELECT *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
   FROM [lightinglog] WHERE [devicerowid] = 101 AND [date] >= date('now'))
SELECT
  ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),3 ) AS Total
FROM rows startdata
JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1
WHERE  startdata.nValue = 1;
```

Various

Cleanup (Interactive)

Delete all time control device entries from the tables LightingLog usig SQLite3 command line.

```
cd /home/pi/domoticz
sudo service domoticz.sh stop
sqlite3
sqlite> .open domoticz.db
sqlite> delete from [LightingLog] where [DeviceRowID] IN (101,102,106);
sqlite> select * from [Percentage] where [DeviceRowID] IN (101,102,106);
sqlite> .quit
```

Clear Device Log (Lua)

To clear a device log via dzVents Lua Script, use

```
IDXSTATUS = 106         -- text status
domoticz.openURL('http://localhost:8080/json.htm?type=command&param=clearlightlog&idx=' .. IDXSTATUS)
```

Clear Device Time Control Status (HTTP API Request)

To clear the text, set the property svalue empty.

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=106&nvalue=0&svalue=

{"status" : "OK","title" : "Update Device"}
```

Timers

Purpose

To perform, in regular intervals, a task or tasks.
Timers can be defined as an Automation Event (all devices) or as a Device Timer (which is only possible for the device type Switches).

Note
Started off by defining timers for all devices by using dzVents.
In the mean time, more and more all kind of Swiches have been added which perform “On Time” tasks.

Rule

Dediced as a rule to use

- **Device Timer** type “On Time” – for a single action, i.e. switch a device On/Off, change setpoint.
- **Automation Events** dzVents - for complex tasks with multiple actions.

Device Timers

The device timers are device specific for Switches only.
These are created via the GUI > Device widget > option Timers.
Enables to perform simple tasks i.e. On | Off, change setpoint value etc.

Types supported (Type,SubType,SwitchType)
(not the full list, these are the ones used for this workbook):

- Thermostat,SetPoint
- Light/Switch,Switch,Dimmer
- Light/Switch,Switch,Push On Button
- Light/Switch,Switch,Push Off Button
- RFY,RFY,Blinds
- Lighting 1, ELRO AB400,On/Off
- Lighting 2, AC,On/Off
- Lighting 4, PT2262,Contact
- Color Switch,WW,Dimmer
- More…

Setpoint Timers

Each radiator thermostat device (type: Thermostat, SubType: SetPoint) has one or more timers to control the temperature.

Setpoint Timers Example

Device

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
207	Bath Thermostat	00160001	1	Hzg Bad Sollwert	Thermostat	SetPoint	20.0

Timers

Name: Hzg Bad Sollwert

2019-12-15 19:27:34 ▲08:31 ▼16:01

Show 25 entries Search:

Active	Type	Date	Time	Command	Days
Yes	On Time		06:00	Temperature, 20	Everyday
Yes	On Time		11:00	Temperature, 19	Everyday
Yes	On Time		23:00	Temperature, 17	Everyday

Setpoint Timers Overview

To get an overview of the setpoint device timers defined, perform an SQL SELECT query on the Domoticz Database. The timers are defined in the table SetpointTimers. The device name is taken from field Name of table DeviceStatus

Example

SQL Select statement to obtain all setpoint timers from the Domoticz Production database.
The device names are in German.
The tables used are DeviceStatus to get the device Name and SetPointTimers to get the timer settings for the device.
The field DeviceRowID is the idx of the device.

```
SELECT d.Name, t.*
FROM DeviceStatus d, SetpointTimers t
WHERE d.ID=t.DeviceRowID
```

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

Switch Timers

Example Switch Device (idx=30). Type: Lighting 2, AC, On/Off

Active	Type	Date	Time	Randomness	Command	Days
No	On Time		00:00	No	On	Everyday
No	On Time		00:05	No	Off	Everyday

SQL SELECT on table Timers

```
SELECT d.Name, t.*
FROM DeviceStatus d, Timers t
WHERE d.ID=t.DeviceRowID
```

Name	ID	Active	DeviceRowID	Date	Time	Type	Cmd	Level	Hue	UseRandomness	TimerPlan	Days	Month	MDay	Occurence	Color
TimerSwitch	1	0	30	2018-09-04	00:00	2	0	100	0	0	0	128	0	0	0	
TimerSwitch	2	0	30	2018-09-04	00:05	2	1	100	0	0	0	128	0	0	0	

Note
The timers are not active in this case.

<TODO>Clarify why field Date has content although the timer definitions do not set a Date but Time only.

<TODO>Add more device timers. Replace Automation scripts which use only one device.

<TODO>Explore if there are any other tables where times are defined.

Timerplans

In the Domoticz settings, several timer plans can be defined.
Initially there is one timer plan: default (with ID=0). This plan is assigned to the timers defined. See previous SQL SELECT query result, field TimerPlan, which has value 0.

In the GUI > Setup > Settings > Other: the timer plan can be set.

The selected timer plan ID, can be found in the Domoticz database, table Preferences , field Key = ActiveTimerPlan, nValue = 0, sValue = “” (empty).

The timer plans are defined in GUI > Setup > More Options > Plans > Timerplan.

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

```
--[[
  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://ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
-- callback of the url request - must be unique across all dzevents
local RES_CCU = 'thermostatmakelabinfo';
-- helper to round a number to n decimals
local DECIMALS = 1;

return {
  on = { timer = { 'every minute'}, httpResponses = { RES_CCU } },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({ url = URL_CCU, method = 'GET', callback = RES_CCU })
    end
    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        -- SETPOINT °C
        local setpointvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
        -- TEMPERATURE °C
        local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')
        -- log
        domoticz.log('Setpoint: ' .. setpointvalue)
        domoticz.log('Temperature: ' .. temperaturevalue)
        local msg = 'SP: ' .. tostring(domoticz.helpers.roundnumber(setpointvalue, DECIMALS)) .. ', T: '
.. tostring(domoticz.helpers.roundnumber(temperaturevalue, DECIMALS))
        -- update the thermostat info device
        domoticz.devices(IDX_THERMOSTAT_MAKELAB_INFO).updateText(msg);
      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

Web App Site Control

Purpose

To control the Domoticz Home Automation System, running on a Raspberry Pi 3B+, via a browser-based application (WebUI, built with Node-RED).

- Easy-to-use User Interface accesible from any device capable running a browser (smartphones, tablets, PC, SBC, TV)
- Control heating (HomematicIP) & lights (Philips Hue)
- Information only for weather data, key dates and status postbox
- … and more to follow …

✓ The goal has been to develop a prototype first, i.e. workout the concept with initial functionality, to be able to enhance or modify the application further.
✓ This solution is also meant as a trigger for ideas or as a suggestion building an alternate User Interface to access this Home Automation & Information System.
✓ The User Interface is translated to the German language.
✓ In the mean time, this app has become main User Interface (using the dark theme).
✓ **Detailed information** about this app can be found on [GitHub.]

Screenshot - example application accessed from a browser on an Android phone. (Dashboard theme: Light (default)

Select thermostat device. The actual temperature is shown as donut. The setpoint is selected from a dropdown with fixed values. Info about the valve and the battery is also given. The data is refreshed every 30 seconds.

Select a light device from a dropdown. The intensity is set by a slider with indicator. The light can be switched on / off.

Weather information in donuts: Temperature, Humidity, Pressure, Wind. The data is refresh every 5 minutes or via the refresh button.

Check the status of the postbox. Info is the flap has been opened. The state can be reset via button or refreshed. The battery voltage is monitored.

Various information starting with daylight, followed by important dates (from a roomplan) and lastly the version of this application.

Screenshot – example application accessed from browser running Windows 10 device. (Dashboard theme: Dark, two function selected)

Web Frontend Customized

Purpose

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

Web Front End Sample

Display weather data from three 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 system via URL.

```
URL: domoticzurl:8080/json.htm?type=devices&used=true&filter=all&favorite=1
```

From the request result (JSON string), selective data for the devices with idx=14,116,117 is obtained.

JSON Result Snippet

The URL returns the data in JSON format.

This snippet shows some of the properties for the devices.
The key result[] is an array with the device information.

In this example the results array (made bold) has a length of 3 for the devices with idx,14,116,117. For each of the devices, the Keys “Data” and “Idx” are used.

Using JavaScript, the result array is parsed (see below).


```
{
   "ActTime" : 1536051617,
  ...
   "app_version" : "4.9980",
   "result" : [
      {
         "Data" : "20.9 C, 79 %",
         "idx" : "14"
      },
      {
         "Data" : "1025 hPa",
         "idx" : "116"
      },
      {
         "Data" : "89;E;14;21;?;?",
         "idx" : "117"
      }
   ],
   "status" : "OK",
   "title" : "Devices"
}
```


JavaScript Snippet

```
// Define the idx to of the devices display
var arrayIdx = [14,116,117];
url = domoticzURL + "json.htm?type=devices&used=true&filter=all&favorite=1"
$.getJSON(url, function(result){
  var idx = 0;
  var sensorInfo = "";
  for (i = 0; i < result.result.length; i++) {
    idx = parseInt(result.result[i].idx);
    if (arrayIdx.indexOf(idx) > -1) {
      sensorInfo += setSensorInfo(result,i);
      console.log("Found ixd=" + 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>
          <img id="linn_logo" src="linn_logo.png">
          <div id="clock">
          </div>
        </div>
      </div>
      <div class="row" id="sensor-container">
      </div>
    </div>

    <script>
      $(document).ready(function(){
        var domoticzURL = "http://vz-ip:8080/";
        //var domoticzURL = "http://localhost:8080/";
        // Define the idx to of the devices display
        var arrayIdx = [14,116,117];
        // Timer interval polling data (ms)
        var intervalTimer = 5000;

        // Update the sensorinfo
        $(function() {
          updateClock();
          updateSensors();
        });

        // Returns the image of the sensor that is stored in the image folder on the Domoticz system
        // /home/pi/domoticz/www/images
        function getSensorImage(TypeImg){
          var sensorURL = "";
```

```
        switch(TypeImg) {
          case "dimmer":
            return domoticzURL + "images/Light48_On.png";
            break;
          case "temperature":
            return domoticzURL + "images/temp-20-25.png";
            break;
          case "wind":
            return domoticzURL + "images/wind48.png";
            break;
          case "gauge":
            return domoticzURL + "images/baro48.png";
            break;
          default:
            return domoticzURL + "images/Light48on.png";
        }
      }

      function setSensorInfo(result,i){
        var sensorInfo = "";
        sensorInfo += ''+
          '<div class="col-sm-6 col-md-4 col-lg-3 sensor-outer-box">'+
            '<div class="sensor-box">'+
              '<img class="sensor-icon" src="' + getSensorImage(result.result[i].TypeImg) +'"></img>'+
              '<div class="sensor-data">'+
                '<h4>' + result.result[i].Name + '</h4>'+
                '<h5>' + result.result[i].Data +'</h5>'+
                '<h6>'  + result.result[i].LastUpdate+'</h6>'+
              '</div>'+
            '</div>'+
          '</div>';
        return sensorInfo;
      }

      function updateClock() {
        var d = new Date();
        var hours = d.getHours();
        if (hours < 10) { hours = '0' + hours;}
        var minutes = d.getMinutes();
        if (minutes < 10) { minutes = '0' + minutes;}
        var seconds = d.getSeconds();
        if (seconds < 10) { seconds = '0' + seconds;}
        document.getElementById("clock").innerHTML = hours + ":" + minutes + ":" + seconds;
        setTimeout(updateClock, 1000);
      }


      function updateSensors() {
        // Get data for the favorite (=Dashboard) devices and select the idx=14,116,117
        // URL: domoticzurl:8080/json.htm?type=devices&used=true&filter=all&favorite=1
        url = domoticzURL + "json.htm?type=devices&used=true&filter=all&favorite=1"
        $.getJSON(url, function(result){
          var idx = 0;
          var sensorInfo = "";
          for (i = 0; i < result.result.length; i++) {
            idx = parseInt(result.result[i].idx);
            if (arrayIdx.indexOf(idx) > -1) {
              sensorInfo += setSensorInfo(result,i);
            }
          }
          document.getElementById("sensor-container").innerHTML = sensorInfo;
          //Finally set a timer to repeatly poll the status of the sensors
          setTimeout(updateSensors,intervalTimer);
        });
      }
    });
  </script>
 </body>
</html>
```

Web Radio Volumio

Purpose

To listen to & control, via Domoticz Dashboard, Web Radio provided by Volumio - the Open Audiophile Music Player.

Features to control Volumio Web Radio

- Set Volumio status Play or Stop.
- Show the title of the current song, updated real time or refresh via switch.
- Set Volume 0 – 100 or switch On | Off.
- Bookmark the current song to the favorites list.
- Indicate status OFF, PLAY, STOP.
- Control via Domoticz Dashboard with dedicated Room Plan.
- Handle Volumio offline state without spoiling logs with error messages.
- Solution must be open for enhancements.

Of course, many options to enhance, but the intention is not to replace the Volumio Web UI.

Screenshot Domoticz Dashboard with Room Volumio

Volumio Current Song List sample	Volumio Favorites List sample
Volumio Status PLAY = switch Volumio Play set to On	
Volumio Status STOP = if the server is started or if switch Volumio Play is set to Off	
Volumio Status OFF = server not reachable (network) or server turned off (shutdown)	

Volumio Setup

A Volumio server has been setup on a Pine A64 (one of the first devices shipped back 2016).

Download the Volumio 2 Pine A64 SD Image from here (see additional info).
Image version: Volumio Digital Audio Player [2.383-2018-03-17].

The Pine A64 is connected to an external amplifier and not using a DAC (<TODO> planned for future).

Get Started

To get started, read the Volumio manual first.

- Connect to Volumio from browser with URL **volumio.local**.
  If not working, determine the server IP address via f.e. the connected router interface.
- Configuration done via the Volumio Web UI Settings.
- Set fixed IP addresses for WLAN and Ethernet.
- Enable SSH (required to access the server via WinSCP or Putty).
- Created an USB with loads of MP3 and plugin to the USB port of the PineA64.
- Do shutdown the server via the Volumio menu Settings > Shutdown.

Solution Options

Experimented with solutions Domoticz dzVents and Node-RED using the Volumio API or the MPD (Music Player Deamon) commands.

My **preferred solution** is to go with Domoticz **dzVents Lua scripting** (reference).

Automation Scripts

Identified two challenges to resolve:

1. The dzVents script event timer updates every 1 minute which means that if a new song is playing, the displayed current song is still the previous and could also result in bookmarking the wrong song.
   Resolved: Adding a switch enbling 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 whats 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 *Switch Type:* 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 *Switch Type:* Dimmer	Set the volume between 0-100
149	Volumio Play	Light/Switch	Switch *Switch Type:* On/Off	Set Play (switch On) or Stop (switch Off)
150	*Volumio Shutdown*	*Light/Switch*	*Switch*	*NOT USED* *<TODO>Find solution how to shutdown.*
151	Volumio Status	General	Alert	Show the status of Volumio: OFF (Red) STOP (Yellow) PLAY (Green)
155	Volumio Refresh	Light/Switch	Switch *Switch Type:* 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	Dasboard with Roomplan Volumio

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

Volumio API Commands

The commands are used in nodes as payload, either incoming or outgoing. To learn which commands are available and how to use, check out the Reference documentation.

Volumio
(Reference)

- volumio.local/api/v1/getstate
- volumio.local/api/v1/commands/?cmd=play
- volumio.local/api/v1/commands/?cmd=stop
- volumio.local/api/v1/commands/?cmd=volume&volume=NN

The getstate command is triggered every 5 seconds by an *Inject Node*.
The response is either a string in case the Volumio server is not reachable (which is converted to a JSON string with status error) or a JSON string containing current song information.

Domoticz
(Reference)

- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":4, "svalue": "OFF"};
- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":1, "svalue": "PLAY"};
- {"command": "udevice", "idx": idxvolumiostatus, "nvalue":2, "svalue": "STOP"};
- {"command": "udevice", "idx": idxvolumiocurrentsong, "svalue": currentsong};
- {"command": "switchlight", "idx": idxvolumiobookmark, "switchcmd": "Off"};
- {"command": "udevice", "idx": idxvolumiofavorites, "nvalue":0, "svalue": currentsong};

The listed message payloads are created in *Function Nodes* with variables, like the idx names and current song (i.e. var idxvolumiostatus = 151;).
The output of the *Function Node* goes to a *MQTT Out Node*, which connects to the Domoticz MQTT broker (localhost:1883) and publishes the messages.

Node-RED Flows

Volumio Control

The comprehensive Volumio Control flow with subflows:

- Volumio to update Domoticz "Volumio Current Song" (idx=144)
- Domoticz Switch "Volumio Play" (idx=149) to set “Volumio Status” (idx=151) PLAY or STOP
- Domoticz Dimmer "Volumio Volume" (idx=148) to set Volumio Volume
- Domoticz Switch "Volumio Bookmark" (idx=145) to set “Volumio Favorites” (idx=146)
- If Volumio shutdown or not reachable set “Volumio Status” (idx=151) OFF

Notes

- Tried to set meaningful nodes names for easier understanding of the flows.
- The comments nodes and the function nodes contain additional information.
- For testing commands used *Inject nodes*, which are removed from the final flow.
- Flow development:
  Opened in a browser following windows (tabs):
  - Domoticz Dashboard
  - Domoticz Log
    (Domoticz Setup > Log)
  - Node-RED Development
  - Optional PuTTy session showin the Node-RED log
    (console command node-red-log)
- Flow development:
  Used Debug nodes to check results or learn message content (esp. MQTT).
  Give the Debug nodes meaning full names to easier read the log (Node-RED Debug tab).
- To check if the Volumio server is reachable, the Ping Node is used. Depending response, the Volumio REST request to get the status is “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 browser).

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

Displayed in browser with http://domoticz-ip:1880/ui

Description (Dasboard UI Node Types)

- Display the current song playing (ui_text)
- Button to bookmark the current song (ui_button)
- Show the last bookmarked song (ui_text)
- Set the volume 0-100 (ui_slider)
- Show volume level (ui_text)
- Set status play or stop (ui_switch)
- Show the status OFF,PLAY,STOP (ui_text)

Dashboard Properties
Tab: Volumio Control, Groups: Current Song & Bookmark, Volume, Status

Used the flow *Volumio Control* to create dedicated MQTT messages (prefix volumiocontrol), which are used by the flow *Volumio Control Dashboard* to set properties.

MQTT Message	**Properties**
Topic	**volumiocontrol/status**
Payload Sample	{"command":"udevice","idx":151,"nvalue":1,"svalue":"PLAY"}
Usage	To set the status of Volumio in a Dashbaord 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.

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.
Source: user_variables_list

```
...
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
File:       user_variables_list.py
Project:    Domoticz-Home-Automation-Workbook
Purpose:    List the user variables
Run:        python3 /home/pi/domoticz/scripts/python/user_variables_list.py
Result:
Name;Idx;Value
TH_AMBIENTLIGHT;1;100
TH_TIMECONTROL;2;2.5
Version:    20200227 by rwbl
"""

import urllib.request
from urllib.error import HTTPError,URLError
import sys
import json
"""
HTTP API Request (running locally on the Domoticz development device)
URL = 'http://127.0.0.1:8080/json.htm?type=command&param=getuservariablesx'
HTTP Response (JSON decodes)
{'result': [
  {'LastUpdate':'2020-02-06 11:47:03','Name':'TH_AMBIENTLIGHT','Type':'0','Value':'100','idx': '1'},
  {'LastUpdate':'2020-02-20 10:32:18','Name':'TH_TIMECONTROL','Type':'1','Value':'2.5', 'idx': '2'}],
  'status': 'OK', 'title': 'GetUserVariables'}
"""
URL = 'http://127.0.0.1:8080/json.htm?type=command&param=getuservariables'
if __name__ == "__main__":
    sys.tracebacklimit = 0
    try:
        req = urllib.request.Request(URL)
    except urllib.request.RequestException as e:
        print("Request not valid:{}".format(e.code))
        raise
    try:
        # parsing response
        response = urllib.request.urlopen(req).read()
        content = json.loads(response.decode('utf-8'))
        # print(content)
        # get the status
        if content['status'] == "OK":
            # get the result array
            result_array = content['result']
            # loop over the array and fetch the device with idx=4
            print('Name','Idx','Value',sep=';')
            for item in result_array:
                print(item['Name'],item['idx'],item['Value'],sep=';')
        else:
            print("Response Error:Check the HTTP API request URL:{}".format(URL))
    except urllib.error.HTTPError as e:
        print("HTTP Error code:{}".format(e.code))
    except urllib.error.URLError as e:
        print("URL Error Reason:{}".format(e.reason))
```

Run

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

Result

```
Name;Idx;Value
TH_AMBIENTLIGHT;1;100
TH_TIMECONTROL;2;2.5
```

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/json.htm?type=command&param=getuservariable&idx=8
```

Response


```
{
   "result" : [
      {
         "LastUpdate" : "2019-02-15 14:29:34",
         "Name" : "TH_STOCK_STOCKA",
         "Type" : "1",
         "Value" : "29",
         "idx" : "8"
      }
   ],
   "status" : "OK",
   "title" : "GetUserVariable"
}
```


Set Value
Set a new value for the threshold of a monitored stock. The value is type 1 (float).

```
http://rpi-ip/json.htm?type=command&param=updateuservariable&vname=TH_STOCK_STOCKA&vtype=1&vvalue=29
```

Response


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

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 the browser request.

Use a JSON node to convert the output to JSON Object used by the Change Node.

Two rules, first move the property Value from the JSON response to the msg.payload, then use that string to get converted to a umber using JSONata function.

Set Value
Set a new value for the threshold of a monitored stock. The value is type 1 (float).

Set the property msg.url using JSONata function join (JSON/API + Value)

The request uses the previous defined msg.url to send to Domoticz

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.

Value 89 obtained via Refresh

Idx	Variable name	Variable type	Current value
10	TH_AMBIENT_LIGHT	Integer	89

Domoticz User Variable List (extract for idx=10)

Changed to 111 and pressed enter

Idx	Variable name	Variable type	Current value
10	TH_AMBIENT_LIGHT	Integer	111

Refresh the page in the browser, shows the updated value for the User Variable

Node-RED Flow

Overview API Requests

Defined User Variable

```
Variable Name: TH_AMBIENT_LIGHT, Type: Integer, Idx: 10
```

HTTP Request Get User Variable

```
http://localhost:8080/json.htm?type=command&param=getuservariable&idx=10
```

HTTP Response (JSON Format)


```
{
   "result" : [
      {
         "LastUpdate" : "2019-02-27 14:37:28",
         "Name" : "TH_AMBIENT_LIGHT",
         "Type" : "0",
         "Value" : "89",
         "idx" : "10"
      }
   ],
   "status" : "OK",
   "title" : "GetUserVariable"
}
```


HTTP Request Update User Variable

```
http://localhost:8080/json.htm?type=command&param=updateuservariable&vname=TH_AMBIENT_LIGHT&vtype=0&vva
lue=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: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 Imafe=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:8080/json.htm?type=custom_light_icons

{
```

```
   "result" : [
  ...
      {
         "description" : "Air Quality",
         "idx" : 101,
         "imageSrc" : "Airquality",
         "text" : "Airquality"
      },
  ...
   ],
   "status" : "OK"
}
```

Folder

The images are copied from the *CustomIcons.zip* to the folder

```
/home/pi/domoticz/www/images/
```

```
ls /home/pi/domoticz/www/images/Airquality*.*
/home/pi/domoticz/www/images/Airquality48_Off.png
/home/pi/domoticz/www/images/Airquality48_On.png
/home/pi/domoticz/www/images/Airquality.png
```

SQL statement to set a Custom Icon

Setting a custom icon via SQL example.

Example: Assign the custom icon “Air Quality” which has idx=101, to the device with idx=41.

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

Explore

This chapter is used to explore areas, like Domoticz functions, MQTT, SQL, Node-RED and more.

API Interaction

Purpose

To explore how to interact using Domoticz API commands and handle response accordingly. Ensure also to lookup the reference Domoticz API/JSON Url's.

This chapter does not go into the depth – just a small example on setting the brightness of a Hue Light via a remote-control device, which could be a micro-controller like an ESP8266 or ESP32 or a Raspberry Pi Zero W.
The remote-control device requires communication with the Domoticz system which is integrated in the SMART Home network.

Option HTTP

Domoticz allows you to interact with all devices using HTTP API Commands generating JSON returns.
This means, it is possible to sent commands to the Domoticz system to act upon and receive a response back.
Sending a command and view the server response can be easily done via a browser.

Note
Under the hood, Domoticz uses HTTP API Commands, which is the preferred way to communicate with the devices.

Example HTTP API Request via Browser
Goal
Set the brightness to 50% for Hue Device with Domoticz idx=118.

HTTP API Request URL

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

HTTP JSON Response

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

Option MQTT

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol.
MQTT provides a publish/subscribe message pattern to provide one-to-many message distribution and decoupling of applications.
Read more in chapter MQTT.

To be able to use MQTT in Domoticz, a MQTT Broker (like mosquito) and a MQTT Client (Hardware MQTT Client Gateway with LAN interface) must be installed (read).

Communication from and to Domoticz works via JSON.
Default MQTT topics of the Domoticz for incoming (domoticz/in) and outcoming (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://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=50
```

HTTP JSON Response

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

MQTT Command

```
{
"command": "switchlight", "idx": 118, "switchcmd":"Set Level", "level": 50
}
```

MQTT JSON Response

```
{
"Battery" : 255,"Level" : 50,"RSSI" : 12,"description" : "","dtype" : "Light/Switch","id" :
"00000005","idx" : 118,"name" : "Hue MakeLab","nvalue" : 2,   "stype" : "Switch",   "svalue1" : "50",
"switchType" : "Dimmer","unit" : 1
}
```

Summary

Comparison HTTP parameters vs MQTT Key: Value pairs

HTTP	MQTT Key: Value
param=switchlight	"command": "switchlight"
idx=118	"idx": 118
switchcmd=Set%20Level	"switchcmd":"Set Level"
level=50	"level": 50

Example Set Virtual Text Sensor

HTTP API Request

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=52&nvalue=0&svalue=Hello%20World
```

HTTP JSON Response


```
{
"status": "OK","title": "Update Device"
}
```


MQTT

```
payload='{"command": "udevice", "idx": 52, "nvalue": 0, "svalue": "Hello World"}
```

MQTT JSON Response


```
{
"Battery" : 255,"RSSI" : 12,"description" : "Display control message from several events.",   "dtype" :
"General",   "id" : "00082051",   "idx" : 52,   "name" : "Control Message",   "nvalue" : 0,   "stype" :
"Text",   "svalue1" : "Hello World",   "unit" : 1
}
```


Domoticz Log 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”

Custom Pages

Purpose

To explore how to create & use custom pages accessible from the Domiticz GUI > Custom.

Examples

Various examples developed which either set or get device data.
The HTML pages are rather simple with some limited CSS styling. The goal is more explore the functionality in using custom pages rather then sophisticated solutions.
The communication between a custom page and Domoticz is via HTTP API requests.

Note
The latest source of the example HTML file can be found in the GitHub repository.

Alert Message Set the level & text of an Alert Device Source: Alert_Message.html	
Quick Message Set the level & text of an Alert Device by using Bootbox custom dialog Source: Quick_Message.html	
RPi Monitor Get in regular intervals the Raspberry Pi CPU & RAM usage Source: Rpi_Monitor.html	
RPi Quick Check Get Raspberry Pi data using Bootbox alert dialog Source: Quick_Check.html	

Switch Hue Light Set the state of a Hue light to on or off Source: Switch_Hue_Light.html	

Prepare

Setup

A custom page is accessable from the Domoticz GUI > Custom.
This requires activation of Domoticz GUI > Setup > Settings > Active Menus > Custom.

The custom pages, files with the extention .html, are stored in the folder

```
/home/pi/domoticz/www/templates
```

on the Domoticz System running on the Raspberry Pi (or other hardware).

Development Hints

Constants& Variables

- Ensure to use unique script wide constants & variables for a page, to avoid the redeclare error message.
- Better to use local function variables instead of script wide vars.
- Use functions with parameters if using script wide contants.
- Some constants in the examples which strives to define only the Domoticz system URL and the idx of the devices used.

Example Constants

```
// Alert Message
const AM_URL_DOMOTICZ = "http://domoticz-ip:8080";
const AM_IDX_ALERTMESSAGE = 110;
```

```
// Notes
const NOTES_ABOUT = "Domoticz Explore Custom Pages.";
const NOTES_URL_DOMOTICZ = "http:// domoticz-ip:8080";
const NOTES_IDX_NOTES = 111;
```

Style

CSS: Domoticz style classes can be used.
Recommend to lookup folder domoticz/www/css to checkout what is defined.
Examples:

`<button id="buttonsave" class="btn btn-success">Save</button>` `<button id="buttoncopy" class="btn btn-warning">Copy</button>`	

`<button id="buttoninfo" class="btn btn-info">Info</button>`	
`<button id="buttonclear" class="btn btn-small btn-danger">Clear</button>`	Clear
`<form action="">` `  <label for="notesubject" id="notesubjectlabel">Subject:</label> ` `  <input type="text" id="notesubject" placeholder="Subject" title="Set the note subject" style="width: 300px !important; min-width: 100px; max-width: 500px;" class="text ui-widget-content ui-corner-all"> ` `  <label for="notecontent" id="notecontentlabel">Content:</label> ` `  <textarea id="notecontent" rows="6" cols="50" placeholder="Add content ..." class="textarea ui-widget-content ui-corner-all"></textarea>` `</form>`	

Standalone Page

Develop the page as a standalone page first using the tags <html<body> … </body</html> and run direct from browser, i.e.

```
file:///<path>/Switch_Hue_Light_Test.html.
```

HTML API Requests

Test the HTML API requests first via browser url, i.e.

```
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=On
```

Debug

Use the browser debug (F12) to check the console output in tab Console.
This is important to ensure proper behaviour of the page.
In JavaScript use console.log() to write info.

```
console.log("Console Message");
```

Example:

Explore Domoticz Source

Explore the Domoticz source code folder www (with subfolders) to learn how Domoticz handles web pages.

As the custom page is embedded in the Domoticz GUI, it is possible to use defined functions also used by Domoticz, i.e.

- Bootstrap framework – Bootbox dialogs, i.e. alert, confirm, custom dialogs
- … explore more on whats possible …

Example Bootbox Alert in a JavaScript Function

HTML

```
<p id="status">Status set by function setStatus</p>
```

JavaScript

```
// update the selector with id status
  function setStatus(msg,showdialog) {
    // Get the current time
    var dateToday = new Date();
    var strTime = dateToday.format("HH:MM:ss");
    msg = strTime + ": " + msg;
    $("#status").html(msg);
    if (showdialog){
      bootbox.alert(msg);
    }
    // console.log(msg);
  }
```

Page Updates

After making page updates, refresh the GUI (F5) or clear the browser cache or restart Domoticz in case non expected behavior.

Custom page Filename

Custom menu: Domoticz sets the first character of the filename to uppercase, therefor used friendly name for the HTML file (i.e. Switch_Hue_Light).
Custom menu: It is also possible to define file names with spaces instead of underscores used previous, i.e. Switch Hue Light which are then showed as the filename defined.

HTML Tags

Important
For the final solution, do not use the tags html, head, title, body in the HTML file - except for the standalone tests.

Page Access

Access a custom page via Domoticz GUI > Custom > filename
OR
Access a custom page via HTTP URL: http://domoticz-ip:8080/#/Custom/pagename

HTTP API Requests

Domoticz HTTP API request are used to retrieve and update data from the Domoticz database.

Tested two options to handle HTTP API requests

- jQuery.getJSON()
- AJAX (**A**synchronous **J**avaScript **A**nd **X**ML)

In exploring the Domoticz source code, AJAX requests seems to be used most.

AJAX uses a XMLHttpRequest object (to request data from the web server) and JavaScript & HTML DOM (to display an/or use the data).

The data returned (HTTP response) from the HTTP API request is in JSON format. Test the response using browser what key:value pairs are returned, to use those in JavaScript.

The response depends on the HTTP API command – lookup the Domoticz HTTP API documentation.
Examples:

Device Update (setter)	
HTTP API Command & Param	type=command&param=udevice&idx=110
HTTP Response	{"status" : "OK","title" : "Update Device"}
JavaScript function (data)	data.status data.title
Device Properties (getter)	
HTTP API Command & Param	type=devices&rid=39

HTTP Response (extract)	{
	"ActTime" : 1583397289,
	"ServerTime" : "2020-03-05 09:34:49",
	"app_version" : "4.11775",
	"result" :
	[
	{
	"Data" : "0.7%",
	"Name" : "CPU Usage",
	"idx" : "39"
	}
	],
	"status" : "OK",
	"title" : "Devices"
	}

JavaScript function (data)	data.ServerTime data.result[0].Data data.status

jQuery.getJSON()

This example function uses jQuery.getJSON() to load JSON-encoded data from the server using a GET HTTP request.

```
var urlDomoticz = "http://domoticz-ip:8080";
var requestUrl = urlDomoticz;
var idxAlertMessage = 110;
var AlertLevel = 2;
var AlertText = "";

// get the alert message text from the full device properties
// update the text of the selector alertmessage
function init_alert_message(){
  requestUrl = urlDomoticz + "/json.htm?type=devices&rid=" + idxAlertMessage;
  $.getJSON(requestUrl,{format: "json"}, function(data){
    // Get the alert level & set the selector alertlevels
    AlertLevel = data.result[0].Level;
```

```
    $("#alertlevel").val(AlertLevel).trigger("chosen:updated");
    // Get the alert message & set selector alertmessage
    AlertText = data.result[0].Data;
    console.log(AlertText);
    $("#alertmessage").val(AlertText);
  });
}
```

AJAX

```
var urlDomoticz = "http://domoticz-ip:8080";
var serverTime = ""
var idxRAMUsage = 3;
var idxCPUUsage = 39;

// get a device property from the array result with 1 entry (index 0)
// example: get_device_property(idxCPUUsage, "Data","#cpuusage");
function get_device_property(idx,property,selector) {
  $.ajax({
    url: urlDomoticz + "/json.htm?type=devices&rid=" + idx,
    async: false,
    dataType: 'json',
    success: function (data) {
      propertyValue = data.result[0][property];
      console.log(propertyValue);
      $(selector).html(propertyValue);
      serverTime = data.ServerTime
      // bootbox.alert(propertyValue + "<br>" + serverTime);
    },
    error: function () {
      bootbox.alert("Error communicating to server!");
      return false;
    }
  });
  return false;
}
```

Standalone Test Template

jQuery.getJSON()

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="http://code.jquery.com/jquery.js"></script>
</head>
<body>
<!-- REMOVE PREVIOUS FOR FINAL SOLUTION -->

<!-- Define the html code -->

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">
  var urlDomoticz = "http://domoticz-ip:8080";
  $(document).ready(function(){
      var requestUrl = urlDomoticz + "/json.htm?type=...";
      $.getJSON(requestUrl,{format: "json"}, function(data){
        console.log(data);
      });
    });
  });
</script>
<!-- REMOVE TAGS BELOW FOR FINAL SOLUTION -->
</body>
</html>
```

AJAX

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="http://code.jquery.com/jquery.js"></script>
</head>
<body>
<!-- REMOVE PREVIOUS FOR FINAL SOLUTION -->
<!-- Define the html code -->

<!-- JavaScript -->
<script type="text/javascript" charset="utf-8">
  var urlDomoticz = "http://domoticz-ip:8080";
  var ASYNCFLAG = false;
  $(document).ready(function(){
      var requestUrl = urlDomoticz + "/json.htm?type=...";
      $.ajax({
        url: requestUrl,
        async: ASYNCFLAG,
        dataType: 'json',
        success: function (data) {
          console.log(data);
      },
      error: function () {
        // define action
        return false;
      }
      });
    });
  });
</script>
<!-- REMOVE TAGS BELOW FOR FINAL SOLUTION -->
</body>
</html>
```

Examples

Note
The latest source can be found in the repository src.

Alert Message

```
<!--
Domoticz custom page: Alert_Message.html (/home/pi/domoticz/www/templates)
Purpose: To set the level & text of an alert device.
Device(s) used: Idx=110,Hardware=VirtualDevices,Name=Alert Message,Type=General,SubType=Alert

The HTML form contains:
* input field to set the alert message
* select element as drop-down list for the alert levels (0=gray, 1=green, 2=yellow, 3=orange, 4=red)
* button to update the Domoticz Alert Device

When the page is loaded, the actual alert message and alert level are set.
Clicking on the button, after setting the alert message an alert level updates the Domoticz device.
The Domoticz HTTP API request is used to retrieve and update data.
This function uses jQuery.getJSON() to load JSON-encoded data from the server using a GET HTTP request.

HTTP API Request Test to set the alert message & level to 3 (yellow):
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=110&nvalue=3&svalue=Hello%20World!
-->
<h3>Explore Domoticz Custom Pages - Alert Message</h3>
<p>20200315 rwbl</p>
<hr>

<form action="">
  <label for="alertmessage" id="alertmessagelabel">Alert Message: </label><br>
  <input type="text" id="alertmessage" title="Set the alert message"><br>
  <label for="alertlevel">Select Alert Level: </label><br>
  <select id="alertlevel">
    <option value="0">Gray</option>
    <option value="1">Green</option>
    <option value="2">Yellow</option>
    <option value="3">Orange</option>
    <option value="4">Red</option>
  </select><br>
</form>
<hr>
<button id="setalertmessage">Submit</button>
<!-- Test Domoticz Result -->
<p id="status">Press submit to set the alert message.</p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  const AM_URL_DOMOTICZ = "http://domoticz-ip:8080";
  // set the idx of the device
  const AM_IDX_ALERTMESSAGE = 110;

  // get the alert message text from the full device properties
  // update the text of the selector alertmessage
  function init_alert_message(url, idx){
    var requestUrl = url + "/json.htm?type=devices&rid=" + idx;
    console.log(requestUrl);
    $.getJSON(requestUrl,{format: "json"}, function(data){
      console.log(data);
      // Get the alert level & set the selector alertlevels
      var alertLevel = data.result[0].Level;
      console.log(alertLevel);
      $("#alertlevel").val(alertLevel).trigger("chosen:updated");
      // $("#alertlevel").text(data.result[0].Data);
      // Get the alert message & set selector alertmessage
      var alertText = data.result[0].Data;
      console.log(alertText);
      $("#alertmessage").val(alertText);
    });
```

```
  }

  function set_alert_message(url, idx){
    var dateToday = new Date();
    var strTime = dateToday.format("HH:mm:ss");
    // Define the request url to submit to the domoticz server
    var alertLevel = $("#alertlevel").children("option:selected").val();
    var alertText = $("#alertmessage").val();
    var requestUrl = url + "/json.htm?type=command&param=udevice&idx=" + idx + "&nvalue=" + alertLevel
+ "&svalue=" + alertText;
    console.log(requestUrl);
    // Submit the request and check the json result. example:
    // {"status" : "OK","title" : "Update Device"}{
    // Load JSON-encoded data from the server using a GET HTTP request.
    // The response JSON keys are accessed using parameter data.status, data.status
    $.getJSON(requestUrl,{format: "json"}, function(data){
      // update the selector with id huestatus
      $("#status").text(strTime + ": Alert Message (idx=" + idx + ") updated. Text: " + alertText + ",
Level: " + alertLevel + ", Server response status: " + data.status + ", title: " + data.title);
    });
  }

  $(document).ready(function(){
    init_alert_message(AM_URL_DOMOTICZ, AM_IDX_ALERTMESSAGE);
    // assign the click function to the hue input checkbox with id hueswitch
    $("#setalertmessage").click(function(){
      set_alert_message(AM_URL_DOMOTICZ, AM_IDX_ALERTMESSAGE)
    });
  });
</script>
```

Quick Message (Bootbox)

```
<!--
Domoticz custom page: Quick_Message.html (/home/pi/domoticz/www/templates)
Purpose: To set the text & level of an alert device by using bootbox dialog.
Device(s) used: Idx=110,Hardware=VirtualDevices,Name=Alert Message,Type=General,SubType=Alert
20200315 by Rwbl
-->

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  const QM_URL_DOMOTICZ = "http://domoticz-ip:8080";
  const QM_IDX_ALERT_MESSAGE = 110;
  const QM_ALERT_TITLE = "Set the Alert Message";

  const formDefinition = '<form>'+
    '<label for="message" id="messagelabel">Message:</label><br>'+
    '<input type="text" id="message" placeholder="Message" title="Set the message" style="width:
100%;max-width: 86%;"><br><br>'+
    '<label for="level" id="levellabel">Level (0=gray, 1=green, 2=yellow, 3=orange,
4=red):</label><br>'+
    '<input type="number" id="level" min="0" max="4" placeholder="1" value="1" title="Set the message
level"><br>'+
    '</form>';

  function setMessage(title,url,idx) {
    var form = $(formDefinition);
    bootbox.confirm({
      title: title,
      message: form,
      callback: function (result) {
        console.log('This was logged in the callback: ' + result);
        if (result === true) {
          var alertMessage = form.find('input[id=message]').val();
          var alertLevel = form.find('input[id=level]').val();
          var requestUrl = url + "/json.htm?type=command&param=udevice&idx=" + idx + "&nvalue=" +
alertLevel + "&svalue=" + alertMessage;
          console.log(requestUrl);
          $.ajax({
```

```
          url: requestUrl,
          async: false,
          dataType: 'json',
          success: function (data) {
            console.log("Set alert message successfully (status: " + data.status + ", title: " +
data.title + ").");
            /* Console log example:
              This was logged in the callback: true
              http://dom/ip:8080/json.htm?type=command&param=udevice&idx=110&nvalue=1&svalue=This is a
new message
              Set alert message successfully (status: OK, title: Update Device).
            */
          },
          error: function () {
            console.log("[ERROR] Set alert message. Can not communicate to the Domoticz server!");
            return false;
          }
        });
      }
      else {
        console.log("Set alert message cancelled.");
      }
    }
  });
}

$(document).ready(function(){
  setMessage(QM_ALERT_TITLE, QM_URL_DOMOTICZ, QM_IDX_ALERT_MESSAGE)
});

</script>
```

Raspberry Pi Monitor

```
<!--
Domoticz custom page: RPi_Monitor.html (/home/pi/domoticz/www/templates)
Purpose: To obtain Raspberry Pi CPU & RAM Usage
Device(s) used (Idx,Hardware,Name,Type,SubType,SwitchType):
39,Raspberry Pi,CPU Usage,General,Percentage
3,,Raspberry Pi,RAM Usage,General,Percentage

The HTML form contains:
* Labels for CPU & RAM Usage

The Domoticz HTTP API request is used to retrieve and update data.
This function uses ajax request to load JSON-encoded data from the server using a GET HTTP request.

HTTP API Request Test for single device to obtain properties in json format:
http://domoticz-ip:8080/json.htm?type=devices&rid=9

TODO:
* Instead setting hardcoded label with text use the device property Name (data.result[0].name).
* Check message (F12): Use of XMLHttpRequest with the synchronous flag set to true is deprecated due to
its impact on user-perceived site performance.
-->

<h3>Explore Domoticz Custom Webpages - RPi Monitor (updates every 10 seconds)</h3>
<p>20200315 rwbl</p>
<hr>

<form action="">
  <label for="cpuusage" id="cpuusagelabel">CPU Usage: </label>
  <label id="cpuusage"></label><br>
  <br>
  <label for="ramusage" id="ramusagelabel">RAM Usage: </label>
  <label id="ramusage"></label><br>
</form>
<hr>
<p id="status"></p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
```

```
  // set the url of the domoticz server
  var RM_URL_DOMOTICZ = "http://domoticz-ip:8080";
  // set the idx of the devices
  var RM_IDX_RAM_USAGE = 3;
  var RM_IDX_CPU_USAGE = 39;

  // update the selector with id status
  function setStatus(msg) {
    $("#status").html(msg);
    // console.log(msg);
  }

  // get a device property from the array result with 1 entry (index 0)
  // example: get_device_property(idxCPUUsage, "Data","#cpuusage");
  function get_device_property(url, idx,property,selector) {
    $.ajax({
      url: url + "/json.htm?type=devices&rid=" + idx,
      async: false,
      dataType: 'json',
      success: function (data) {
        propertyValue = data.result[0][property];
        console.log(propertyValue);
        $(selector).html(propertyValue);
        serverTime = data.ServerTime
        // bootbox.alert(propertyValue + "<br>" + serverTime);
      },
      error: function () {
        bootbox.alert("Error communicating to server!");
        return false;
      }
    });
    return true;
  }

  // get the device properties and update the html selector
  function refresh_data(){
    clearInterval($.refreshTimer);
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_CPU_USAGE, "Data","#cpuusage");
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_RAM_USAGE, "Data","#ramusage");
    setStatus("Last update: " + serverTime);
    $.refreshTimer = setInterval(refresh_data, 10000);
  }

  $(document).ready(function(){
    // Testing without refresh_data
    /*
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_CPU_USAGE, "Data","#cpuusage");
    get_device_property(RM_URL_DOMOTICZ, RM_IDX_RAM_USAGE, "Data","#ramusage");
    setStatus("Last update: " + serverTime);
    */

    // Regular updates
    refresh_data();
  });
</script>
```

Raspberry Pi Quick Check (Bootbox)

```
<!--
Domoticz custom page: Quick_Check.html (/home/pi/domoticz/www/templates)
Purpose: To quickly check some of the device data by using bootbox dialog.
Device(s) used: Idx and data properties
20200315 by Rwbl
-->

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  const QC_URL_DOMOTICZ = "http:// domoticz-ip:8080";
  const QC_DIALOG_TITLE = "Quick Check Raspberry Pi Data";

  function getData(title, url) {
```

```
    var searchDevices = [2,3,7,8,39];
    console.log(searchDevices);
    var requestUrl = url + "/json.htm?type=devices&used=true&displayhidden=0";
    console.log(requestUrl);
    $.ajax({
      url: requestUrl,
      async: false,
      dataType: 'json',
      success: function (data) {
        // check there are entries in the history log
        if(typeof data.result == "undefined"){
          // console.log(data);
          console.log("Status all devices - no devices found (status: " + data.status + ", title: " +
data.title + ")");
          return false;
        }
        console.log("Status all devices - Succesfully retreived (status: " + data.status + ", title: " +
data.title + ")");
        statusDevices = [];
        var listDevices = "<ul>";
        data.result.forEach(function(entry) {
          // console.log(entry.idx + ", " + entry.Name + ", " + entry.Data);
          if (searchDevices.includes(Number(entry.idx))) {
            listDevices += "<li>" + entry.Name + " = " + entry.Data + " (idx=" + entry.idx + ")</li>";
            console.log("Pushing " + entry.idx + ", " + entry.Name + ", " + entry.Data);
            statusDevices.push(entry);
          }
        });
        listDevices += "</ul>";
        console.log("Status all devices - Done");
        setMessage(title, listDevices);
      },
      error: function () {
        console.log("[ERROR] Status all devices. Can not communicate to the Domoticz server!");
        return false;
      }
    });
  }

  function setMessage(title, msg) {
    bootbox.alert({
      title: title,
      message: msg,
      callback: function (result) {
        console.log('This was logged in the callback: ' + result);
      }
    });
  }

  $(document).ready(function(){
    getData(QC_DIALOG_TITLE, QC_URL_DOMOTICZ);
  });

</script>
```

Switch Hue Light

```
<!--
Domoticz custom page: Switch_Hue_Light.html (/home/pi/domoticz/www/templates)
Purpose: To switch a Philips Hue light on / off.
Device used: Idx=118,Hardware=Philips Hue Bridge 1,Name=Hue
MakeLab,Type=Light/Switch,SubType=Switch,SwitchType=On/Off.
The switch (input type checkbox) is triggered by a Domoticz HTTP API request, which is defined &
submitted in the JavaScript function click.
This function uses jQuery.getJSON() to load JSON-encoded data from the server using a GET HTTP request.
Request URL to switch the light On/Off (case sensitive):
http://domoticz-ip:8080/json.htm?type=command&param=switchlight&idx=118&switchcmd=On | Off
-->

<h3>Explore Domoticz Custom Pages - Switch Hue Light</h3>
<p>20200315 rwbl</p>
<hr>

<form action="">
```

```
  <input type="checkbox" id="hueswitch" title="Select the checkbox to turn the light on">
  <label for="hueswitch" id="hueswitchlabel"></label><br>
</form>
<hr>
<p id="status">Select the checkbox to turn the light on.</p>

<!--
  JavaScript
-->
<script type="text/javascript" charset="utf-8">
  // set the url of the domoticz server
  const SH_URL_DOMOTICZ = "http:// domoticz-ip:8080";
  // set the idx of the switch (=hue light)
  // var IDXSWITCH = 113;        // Hue WZ Rob
  const SH_IDX_SWITCH = 118;     // Hue MakeLab

  // get the status of the hue light on or off from the full device properties
  // update the selector huestatus
  function hue_status(url, idx){
    var requestUrl = url + "/json.htm?type=devices&rid=" + idx;
    // console.log(requestUrl);
    $.getJSON(requestUrl,{format: "json"}, function(data){
      // console.log(data);
      // set the checkbox depending light state = important to use off as data could contain "Off","Set
Level..."
      var ischecked = (data.result[0].Data == "Off") ? false : true;
      $("#hueswitch").prop('checked', ischecked);
      // set the switch name as label for the checkbox
      $("#hueswitchlabel").text(data.result[0].Name + " (idx=" + idx + ")");
      // document.getElementById("hueswitchlabel").innerHTML = data.result[0].Name + " (idx=" + idx +
")";
      // update the selector with id huestatus
      // console.log(data);
      $("#status").text("Hue Light " + idx + ", " + data.result[0].Name + ", Data: " +
data.result[0].Data);
      //document.getElementById("huestatus").innerHTML = "Hue Light " + idx + ", " + data.result[0].Name
+ ", Data: " + data.result[0].Data;
    });
  }

  function hue_switch(url, idx){
    // Get the current time
    var dateToday = new Date();
    var strTime = dateToday.format("hh:mm:ss");
    // get the togge button state
    var state = $("#hueswitch").is(":checked");
    // Define the switch command - ensure to use correct syntax for On and Off
    var switchCmd = (state == true) ? "On" : "Off";
    console.log(state);
    // Define the request url to submit to the domoticz server
    var requestUrl = url + "/json.htm?type=command&param=switchlight&idx=" + idx + "&switchcmd=" +
switchCmd;
    console.log(requestUrl);
    // Submit the request and check the json result. example: {"status" : "OK","title" : "SwitchLight"}
    // Load JSON-encoded data from the server using a GET HTTP request.
    // The response JSON keys are accessed using parameter data.status, data.status
    $.getJSON(requestUrl,{format: "json"}, function(data){
      // update the selector with id huestatus
      $("#status").text(strTime + ": Hue Light (idx=" + idx + ") switched: " + switchCmd + ", Server
response status: " + data.status + ", title: " + data.title);
      // document.getElementById("huestatus").innerHTML = "Hue Light " + IDXSWITCH + " switched: " +
switchCmd + ", Server Response title: " + data.title + ", status: " + data.status;
    });
  }

  $(document).ready(function(){
    hue_status(SH_URL_DOMOTICZ, SH_IDX_SWITCH);
    // assign the click function to the hue input checkbox with id hueswitch
    $("#hueswitch").click(function(){
      hue_switch(SH_URL_DOMOTICZ, SH_IDX_SWITCH)
    });
  });
</script>
```

Notes

Domoticz custom page: Notes.html (/home/pi/domoticz/www/templates)
This is a more complex example of a custom page.
See the function Notes for details.

Screenshot - might not be the latest – see the function Notes.

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.

Note
The file EXPLORE_SCRIPTING.md (in folder scripts) contains sample dzVents Lua, Lua, Python and Blockly scripts to explore automation events.
These scripts are not used anymore but kept as learning reference.

Event Execution Order

The order of events being executed, is:
1. file scripts - stored in the folder /home/pi/domoticz/scripts/…
2. database scripts - stored in tables [EventMaster] and [EventRules]
3. on/off action script - explore usage & storage

Important *(taken from the Domoticz Wiki)*
Scripts that written on the file system and inside Domoticz using the internal web-editor share the same namespace. If there are two scripts with the same name, only the file system script will be used – this is logged.

Event Scheduling & Trigger

The scheduling is fixed in Domoticz.

- ALL "device" scripts (scripts with "device" in their name) run everytime ANY device status changes.
- ALL "time" scripts (scripts with "time" in their name) run every MINUTE.

So far, have been using file scripts but considering moving to database solution only for device events.

For event triggerd 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.

Event Development

Whilst creating scripts using the Event Editor, two browser tabs are open:

- Domoticz Event Editor – to create/modify scripts using Ace Editor, to activate/deactivate a script,
- Domoticz Log (Domoticz GUI Setup > Log > Tab All) – to check if the script is running OK after save.

Example Basement Humidity Monitor

Purpose

To explore the various interpreter using the same event to monitor a device:
Device: Keller (idx=11)
Property: humidity
Rule:
If the Basement Humidity is >= 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_l, Lua, Device.

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_l:
[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.

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.

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 system ip with port!
domoticzserver="localhost:8080"

# Devices
KellerIdx = 11
KellerName = 'Keller'
TextInfoIdx = 52
TextInfoName = 'Info Message'

# Set the threshold to monitor the humidity (%RH)
HumidityThreshold = 70

def domoticzrequest (url):
    """Send update request to domoticz
    :url: Domoticz JSON/API url
    :result: Result is a json object
    """
    req = Request(url)
    try:
        response = urlopen(req)
    except HTTPError as e:
        # print('Server couldn\'t fulfill the request.')
        # print('Error code: ', e.code)
        return "Error fulfilling the request." + e.code
    except URLError as e:
        # print('Failed to reach a server.')
```

```
        # print('Reason: ', e.reason)
        return "Error reaching the server." + e.code
    else:
        # everything is fine
        return response.read().decode('utf-8')

if DE.changed_device_name == KellerName:
    # Test Log
    # 2018-09-02 11:32:46.085 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7, switchType:
0, s_value: 18.7;75;3, n_value: 0, n_value_string: 18.7;75;3, last_update_string: 2018-09-02 11:32:46
    DE.Log("Python: Changed: " + DE.changed_device.Describe())
    # Test s_value
    # 2018-09-02 11:32:46.085 18.7;75;3
    svalue = DE.Devices[KellerName].s_value
    DE.Log(svalue)
    # Get the humidity, which is at the 2nd entry of the svalue, i.e. 75
    # Split the svalue by ;. The result is a list with length 3, data[0]=T,data[1]=H,data[2]=Comfort
    data = svalue.split(";")
    # Log all 3 entries
    # for temp in data:
    #     DE.Log(temp)
    # Get the humidity at pos 2 which is index 1 :
    if len(data) == 3:
        Humidity = int(data[1])
        if Humidity >= HumidityThreshold:
            # Log that the humidity is above threshold. The log string uses new string formatting
            svalueinfo = 'Basement Humidity >= {}%RH ({}%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 using Lua.

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, do not use the same vars across scripts which are not local etc.

Device Properties

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

Script

This example checks if the defined devices (see idx of the devices) have changed, then dumps the properties but also shows the device state property.

```
-- Idx of the devices
local IDXMEMORYUSAGE = 1;
local IDXHDDUSAGE = 3;
local IDXTEMPERATURE = 4;
local IDXCONTROLMSG = 52;

return {
  on = {
    devices = {
      IDXMEMORYUSAGE,
      IDXHDDUSAGE,
      IDXIDXTEMPERATURE
    }
  },
  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.
This is the recommend way to use common functions.

Example

Update the virtual device named "Alert Message" (Type General, SubType Alert, Idx=55) by using a function defined in external module "msgbox.lua".

```
local message = device.name .. ' switched OFF ' -- .. msg.isnowdatetime(domoticz)
msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)
```

Module Access

The module “msgbox.lua” MUST be in folder:

```
/home/pi/domoticz/scripts/dzVents/scripts
```

To access functions from the module, use the syntax **msgbox.functionname**, when the module is defined like:

```
local msgbox = require('msgbox')
```

Example (screenshot from WinSCP)

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”.
This is the recommend way to use common functions.

Example Functions

```
return {

  -- global helper functions
  helpers = {

    -- Idx of devices used
      IDX_ALERTMSG = 55,
      IDX_CONTROLMSG = 52,

        -- Update the alert message with level and text
        alertmsg = function(domoticz, level, msg)
        domoticz.devices(domoticz.helpers.IDX_ALERTMSG).updateAlertSensor(level, msg)
        end,

        -- Update the control message with text
        controlmsg = function(domoticz, msg)
          domoticz.devices(domoticz.helpers.IDX_CONTROLMSG).updateText(msg)
        end

  }

-- end return
}
```

Usage

```
-- set the alert message
Local message = ’This is an alert message’
domoticz.helpers.alertmsg(domoticz, domoticz.ALERTLEVEL_ORANGE, message)
```

Hints

Clear Device Log

To clear a device log via dzVents Lua Script, use

```
IDXSTATUS = 106          -- text status
domoticz.openURL('http://localhost:8080/json.htm?type=command&param=clearlightlog&idx=' .. IDXSTATUS)
```

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 a browser
- Enter the 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 system IP address. For other settings, the defaults are used.

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/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/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/control?cmd=gpio,14,1
Off Action	http://espeasy-ip/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	*BH1750 = ESP8266 NodeMCU* VCC = 3v3 GND = GND SCL = D1 (GPIO5) SDA = D2 (GPIO4) ADDR = GND Default I2C address 0x23 (35)
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 then 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 List

Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
Edit 6	✔	Generic - Dummy Device	AmbientLight_Settings				Threshold: 100 State: 0 : 0.00 : 0.00

Extract from the list of devices.
The Generic - Dummy Device named AmbientLight_Settings, has task number 6 with the 3 variables with index 1 to 2.
This information is required to update a variable in the rule using the command:

```
TaskValueSet TaskNr,VariableIndex,Value
```

ESP Easy Rule

```
// Check ambient light device named ESPEasy_BH1750 value Lux (see devices properties).
// If Lux value less then threshold (i.e. 100), switch the LED ON else OFF.
// The LED is connected to pin D5 (GPIO14).
// Command syntax: gpio, gpio number, 1|0 - 1=ON;0=OFF
// A generic dummy device is used to set the threshold at boot and track the state of the LED
indicator, to avoid sending GPIO commands permanently
// [AmbientLight_Settings#Threshold], [AmbientLight_Settings#State]


// Init the variables for device AmbientLight_Settings
on System#Boot do
  // Set ambient light threshold setting
  TaskValueSet 6,1,100
  // Turn LED off
  gpio,14,0
  // Update led state setting
  TaskValueSet 6,2,0
endon

// Handleambientlight lux changes
on ESPEasy_BH1750#Lux do

  // Check if Lux below threshold and LED is OFF
  if [ESPEasy_BH1750#Lux]<[AmbientLight_Settings#Threshold] and [AmbientLight_Settings#State]=0
    // Turn LED on and update dummy variable
    gpio,14,1
    TaskValueSet 6,2,1
  endif

  // Check if Lux above/equal threshold and LED is ON
  if [ESPEasy_BH1750#Lux]>=[AmbientLight_Settings#Threshold] and [AmbientLight_Settings#State]=1
    // Turn LED off and update dummy variable
    gpio,14,0
    TaskValueSet 6,2,0
  endif

endon
```

ESP Easy Log Entries

LED switched ON triggered by Lux value 6.67 below threshold 100.

```
531124: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 6.67
531128: EVENT: ESPEasy_BH1750#Lux=6.67
531259: ACT : gpio,14,1
531263: SW : GPIO 14 Set to 1
531266: ACT : TaskValueSet 6,2,1
531273: Command: taskvalueset
531394: Domoticz: Sensortype: 1 idx: 46 values: 6.67
```

LED switched OFF triggered by Lux value 273.33 above threshold 100.

```
546131: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 273.33
546135: EVENT: ESPEasy_BH1750#Lux=273.33
546324: ACT : gpio,14,0
546328: SW : GPIO 14 Set to 0
546331: ACT : TaskValueSet 6,2,0
546338: Command: taskvalueset
546431: Domoticz: Sensortype: 1 idx: 46 values: 273.33
```

ESP controlling Hue Brightness

Purpose

To control the brightness of the Hue light “MakeLab” with idx=118 via potentiometer connected to an ESP8266 NodeMCU.

Parts

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

ESP Easy Device Configuration – Dummy Devices

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

	Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
Edit	1	✔	Analog input - internal	HUE_Dimmer			ADC (TOUT)	Analog: 40
Edit	2	✘	Switch input - Switch	Hue_Light_Switch		1 (154)	GPIO-14	Switch: 0
Edit	3	✘	Display - LCD2004				GPIO-4 GPIO-5	
Edit	4	✔	Light/Lux - BH1750	ESPEasy_BH1750		2 (46)	GPIO-4 GPIO-5	Lux: 46.67
Edit	5	✔	Generic - Dummy Device	Varstore				HUE_Dimmer: 40 HUE_Dimmer_Delta: 1 : 0.00 : 0.00

ESP Easy Rule

```
// Control Hue light via Potmeter named HUE_Dimmer with value Analog.
// A dummy device named Varstore (TaskNr 5) is used to store 2 values:
// Value 1 = HUE_Dimmer for the previous Hue light value
// Value 2 = HUE_Dimmer_Delta for the absolute delta brightness between Hue light and previous value

// Listen to Hue light analog value changes
on HUE_Dimmer#Analog do

  // Save the delta to dummy variable value 2
  TaskValueSet 5,2,[HUE_Dimmer#Analog]-[Varstore#HUE_Dimmer]
  Delay 5

  // Get the absolute delta for the new dimmer value
  // Update the dummy variable #2
  if [Varstore#HUE_Dimmer_Delta]<0

    TaskValueSet 5,2,[Varstore#HUE_Dimmer_Delta]*-1

  endif

  // Check if delta > 2, this to avoid noise
  // Submit api rest request to domoticz system to set brightness level
  if [Varstore#HUE_Dimmer_Delta]>2

    SendToHTTP domoticz-
ip,8080,/json.htm?type=command&param=switchlight&idx=118&switchcmd=Set%20Level&level=[HUE_Dimmer#Analog
]

  endif

  // Update the dummy variable #1
  TaskValueSet 5,1,[HUE_Dimmer#Analog]

endon
```

Solution 2

This solution uses an ESP Easy Analog input device, Domoticz Text device (Hardware Virtual Sensor) and Domoticz dzVents script.
It is not the preferred way to set the brightness, it is just to show converting the MQTT default command into a value.

ESP Easy

Hardware	*Potentiometer = ESP8266 NodeMCU* 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

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.

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

.

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

The Barometer forecast can be one of:
0 = No Info,1 = Sunny,2 = Paryly 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.

Kodi Music Player

Purpose

- To explore the music player functionality of the Kodi Media Player.
- To explore Kodi configuration, communicating with Kodi using JSON-RPC API requests, control & monitor the media player from various systems (Remote Browser, Domoticz and other).
- To setup a Kodi music player running on a Raspberry Pi with LibreELEC and Kodi Media Player, connected to the home stereo system and integrated in the local network.
- To use an old RPi 2 B v.1.1 (2014) as a starter - re-activated after its retirement. If the solution is running fine, the old RPi 2 will be replaced by a newer RPi 4.
- To prepare for a Domoticz Music Player Solution based on virtual sensors controller by a single dzVents Lua Automation Script (see Domoticz-Home-Automation-Workbook function Music Player)

Kodi - free and open-source media player software application developed by the XBMC Foundation.
LibreELEC - lightweight operating system designed to run Kodi.

Hardware

- RPi 2B V1.1 (2014)
- SDHC card 32GB
- USB stick with media files
- USB WiFi Adapter 802.11m

Installation

- Download image Raspberry Pi NOOBS: NOOBS_v3_3_1.zip.
- Flashed on a 32GB SDHC Card using Etcher 1.5.53 under Microsoft Windows [Version 10.0.18363.815] device.
- Inserted the SDHC card into the RPi.
- RPi connect HDMI Monitor, USB Keyboard, USB Mouse, Ethernet wired cable.
- RPi plug in power.
- RPi installation process started - select LibreELEC_RPi2 - Kodi Entertainment Center.
- After completion, reboot the RPi.
- (Brief information, for mored details lookup various websites, i.e. Raspberry Pi, LibreELEC, Kodi)

Notes

- Installed LibreELEC (official): 9.2.1 (RPi2.arm) with Kodi 18.6-Leia, Linux ARM 32-bit version 4.19.106 (267114), JSON-RPC API v8 (Krypton).
- Software versions are subject to change - checkout for newer versions.

Kodi Settings

- RPi IP-Address Static: player-ip, Port:8080
- HTTP Remote access
- Audio: output device: PI:Analoque; Channels: 2.0;
- Input: Keyboard layout: German
- Internet access: Use proxy server: disabled;
- My add-ons>All>LibeELEC Configuration>Run:
    - System>System name: KEC; Keyboard: de
    - Network>Wireless>Active
    - Connections>Selected Wifi and set statis IP address (set interface: wlan0,type: manual, address: player-ip)
    - Bluetooth disabled
    - Services: Samba and SSH enabled
- Services>Control:
    - Webserver: allow remote control via HTTP; Port:8080; Leave UN & PW empty (if not then issues with Node-RED or Domoticz HTTP RPC requests)
    - Application control: allow from this & other systems
- Addon "Script - Raspberry Pi Tools" Author:Team LibreELEC
    - Installed via GUI: Addons > Search > Script - Raspberry Pi Tools

Kodi SSH Access
Use "ssh root@player-ip" to access Kodi.
Example from a Windows 10 Command Prompt Window:

```
Microsoft Windows [Version 10.0.18363.815]
(c) 2019 Microsoft Corporation. All rights reserved.
ssh root@player-ip
root@player-ip's password:
##############################################
#                LibreELEC                   #
#            https://libreelec.tv            #
##############################################
LibreELEC (official): 9.2.1 (RPi2.arm)
```

Kodi GUI Access
Open web browser and enter URL: http://player-ip:8080

Kodi Hints

Get Kodi Version
Run command from the CLI:

```
grep Kodi .kodi/temp/kodi.log | head -3
```

Output

```
2020-05-11 08:42:00.196 T:1937043472  NOTICE: Starting Kodi (18.6 Git:newclock5_18.6-Leia). Platform:
Linux ARM 32-bit
2020-05-11 08:42:00.196 T:1937043472  NOTICE: Using Release Kodi x32 build (version for Raspberry Pi)
2020-05-11 08:42:00.196 T:1937043472  NOTICE: Kodi compiled 2020-03-05 by GCC 8.3.0 for Linux ARM 32-
bit version 4.19.106 (267114)
```

Set Raspberry Pi Volume
Set the volume to max on the RPi and control the volume remote on the connected device or remote application (like browser).

Option via Browser
In the Desktop (attach monitor, keyboard, mouse to the RPi)
Open the Kodi player in the browser and set the volume via the keyboard + key.
This enables to remote control the volume, when the Raspberry Pi is connected to an amp or a headset.

Option via Command Line
Open a terminal and use the command *amixer* with paremeter GET or SET the volume.

```
GET volume: amixer -M sget PCM
SET volume: amixer -q -M sset PCM 50%
```

Example set and get the volume to 70%
SET Volume

```
amixer -q -M sset PCM 70%
```

GET Volume

```
amixer -M sget PCM
```

Output

```
Simple mixer control 'PCM',0
  Capabilities: pvolume pvolume-joined pswitch pswitch-joined
  Playback channels: Mono
  Limits: Playback -10239 - 400
  Mono: Playback -529 [70%] [-5.29dB] [off]
```

Kodi JSON-RPC API Communication

JSON-RPC is a HTTP- and/or raw TCP socket-based interface for communicating with Kodi. The JSON RPC API methods are used to get or set data in Kodi via Domoticz or other systems, like Node-RED.
To get started, went through the JSON-RPC API documentation v8 (Krypton) at the time of writing.
Examples using the JSON RPC API are:

- **Get** the current song playing and display in a Domoticz text device - Method Player.GetItem
- **Set** the playback to play or pause via a Domoticz Switch - Method Player.PlayPause

Domoticz Automation Events

This simple music player is controlled by Domoticz devices triggering Domoticz Automation Events dzVents.

The communication between Domoticz and the player is handled via Asynchronous HTTP GET and POST requests defined in the dzVents scripts.
The dzVents command *domoticz.openURL()* with parameter is used.
The dzVents scripts are triggered by Domoticz device changes or by timer.

Read Data from Kodi

To read data from Kodi, use the **GET method**.
Example Kodi JSON-RPC API method: *Player.GetItem*
Below info has been taken from the the JSON-RPC API v8 documentation.
Have added additional clarification.

Method
Player.GetItem
Retrieves the currently played item.

Permissions
ReadData.
For _ReadData_, the HTTP GET request can be used. If permission is from type _ControlPlayback_ then a HTTP POST request is required (the post data defined in the HTTP request body).

Parameters
Player.Id playerid; List.Fields.All properties
The parameters are defined as JSON key:value pairs in the request key "params":

- Key *playerid* defined as "playerid":0
- Key *properties* defined as array with selected keys taken from "List.Fields.All" which has items defined as enumeration.
- For the example, selected "title","album","artist","duration".

Returns
Type
Object
Properties: List.Item.All item
Key
item contains the key from "List.Item.All" which has a range of properties.
Returned are the selected properties from the request:
"album":"ALBUM","artist":["ARTIST"],"duration":NNN,"id":NNN,"label":"LABEL","title":"TITLE","type":"song"
Note
The property "type" is always returned.
Note
On dzVents function openURL.

- The **postdata** MUST be encoded using the backslash character - if not done, Kodi can not parse the request.

- The **header** MUST be from content type application/json (which is set explicitly to show its important).
- The HTTP response string must be unique across all Domoticz Automation Events.

dzVents script snippet

```
local KODI_IP = 'http://player-ip:8080/jsonrpc'
local KODI_JSONRPC_GETITEM =
'{\"jsonrpc\":\"2.0\",\"method\":\"Player.GetItem\",\"params\":{\"properties\":[\"title\",\"album\",\"a
rtist\",\"duration\"],\"playerid\":0},\"id\":\"AudioGetItem\"}'
local KODI_RES = "KODISONGPLAYING"

-- HTTP GET Request to readdata
local urlRequest = string.format("%s?request=%s", KODI_IP, KODI_JSONRPC_GETITEM)
domoticz.openURL({url = urlRequest, method = 'GET', callback = KODI_RES})

-- HTTP Response
--
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
77,"id":233,"label":"LABEL","title":"TITLE","type":"song"}}}
-- Pick from the property result.item the properties artist & title
local song = item.json.result.item
-- note the artist is the first entry (1) of the artist array
domoticz.log(string.format("%s - %s", song.artist[1], song.title))
```

Trigger: Timer every minute.

Write Data to Kodi

To write data to Kodi, use the **POST method**.
Example Kodi JSON-RPC method: *Player.PlayPause*.

dzVents script snippet

```
local KODI_IP = 'http://player-ip:8080/jsonrpc'
local KODI_RES = "KODISONGPLAYBACK"
local KODI_JSONRPC_POSTDATA =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.PlayPause\",\"params\":{\"playerid\":0},\"id\":1

-- HTTP POST Request to ControlPlayback
local urlRequest = string.format("%s", KODI_IP)
domoticz.openURL({url = urlRequest, method = 'POST', headers = { ['content-type'] = 'application/json'
}, callback = KODI_RES, postData = KODI_JSONRPC_POSTDATA })

-- HTTP Response
-- {"id":1,"jsonrpc":"2.0","result":{"speed":0}}
local speed = item.json.result.speed
```

The trigger to set the player state to play or pause is via Domoticz Device change using a selector switch (virtual sensor).

JSON-RPC API HTTP Request Examples

The JSON-RPC API HTTP requests can be tested prior using in dzVents via CLI **curl** command with the JSON-RPC API command as parameter.
The next examples have been tested on the Kodi system via the CLI with url:port localhost:8080.
For more commands, see Chapter Kodi JSONRPC Commands (Selection).

GET - Player.GetItem

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.GetItem\", \"params\": { \"properties\": [\"title\"], \"playerid\": 0 }, \"id\":
\"AudioGetItem\"}" http://localhost:8080/jsonrpc
```

HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"id":606,"label":"SONGLABEL","title":"SONGTITLE"
,"type":"song"}}}
```

GET - Player.GetProperties

This method can be used to check if the player is playing or paused, i.e. speed=1 or speed=0

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.GetProperties\", \"params\" :{ \"playerid\": 0, \"properties\":[\"speed\", \"time\"]},\"id\":
\"AudioGetItem\"}"  http://player-ip:8080/jsonrpc
```

HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"speed":1,"time":{"hours":0,"milliseconds":287,"minutes"
:0,"seconds":0}}}
```

Check the total time and time played for the current song playing

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.GetProperties\", \"params\" :{ \"playerid\": 0, \"properties\":[\"speed\",
\"percentage\",\"time\", \"totaltime\"]},\"id\": \"AudioGetItem\"}"  http://player-ip:8080/jsonrpc
```

HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":25.5638370513916,"speed":1,"time":{"hours":
0,"milliseconds":268,"minutes":1,"seconds":4},"totaltime":{"hours":0,"milliseconds":402,"minutes":4,"se
conds":11}}}
```

POST - Player.PlayPause

```
curl -X POST -H "Content-Type: application/json" -d
'{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":0},"id":"AudioGetItem"}'
http://localhost:8080/jsonrpc
```

HTTP Response

```
{"id":1,"jsonrpc":"2.0","result":{"speed":0}}
```

The player set to paused. Running again sets the speed to 1 and the item is playing.

Domoticz Explore

Several devices with single Automation Scripts dzVents Lua are created to explore communicating and controlling a Kodi Music Player.
The devices created are from hardware type "Dummy" Virtual Sensors. The hardware entry is named "VirtualDevices".
For the JSONRPC commands, it is strongly recommended to escape the JSON formatted string (i.e. character " to \").

The focus of the examples is to show how to use the HTTP JSON-RPC API command requests GET to read data or POST to write data.
There are many commands available, but the shared examples are used as a base for any other solutions.

Hint
The devices used are updated rather frequent, which means the Domoticz databse growth in size. Recommend to clear the device logs regulary eith manual or Domoticz settings or use JSON/API request.

Get Song Playing Timer

Display the current song playing; Lists the history of songs played including error messages.

Device(s)	117,VirtualDevices,Kodi Song,General,Text,DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playing_timer.dzvents
Trigger	Timer every minute
Async HTTP request method	GET
JSON-RPC API Command	local JSONRPC_GETITEM = "{\"jsonrpc\":\"2.0\", \"method\":\"Player.GetItem\", \"params\":{\"properties\":[\"title\", \"album\", \"artist\", \"duration\"], \"playerid\":0}, \"id\":\"AudioGetItem\"}"

Automation Event

```
--[[
-- kodi_explore_playing_timer.dzvents
-- Get the current song playing from the Kodi Media Center and display in a text device
-- Uses a timer and updates every minute - highest update interval possible via timer
-- url get (=readdata) request from the kodi server, to obtain artist & title playing
-- http example:
-- http://player-
ip:8080/jsonrpc?request={"jsonrpc":"2.0","method":"Player.GetItem","params":{"properties":["title","alb
um","artist","duration"],"playerid":0},"id":1}
-- http response JSON string:
--
{"id":1,"jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":274,"id":227,"
label":"LABEL","title":"TITLE","type":"song"}}}
-- Notes:
-- * Method Player.GetItem still responses a title if the song played is paused.
-- * Introspect: detailed information about an API method - example Player.GetItem
-- http://player-ip:8080/jsonrpc?request={ "jsonrpc": "2.0", "method": "JSONRPC.Introspect", "params":
{ "filter": { "id": "Player.GetItem", "type": "method" } }, "id": 1 }
--
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
```

```
-- Event unique callback
local REQUEST_CALLBACK = "KODIPLAYINGTIMER"
-- Request for method Player.GetItem for artist:title
local JSONRPC_GETITEM =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GetItem\",\"params\":{\"properties\":[\"title\",\"album\",\"a
rtist\",\"duration\"],\"playerid\":0},\"id\":\"AudioGetItem\"}"
-- Domoticz device - text song playing artist:title
local IDX_PLAYING = 117

return {
  on = { timer = { 'every minute' }, httpResponses = { REQUEST_CALLBACK } },
     data = { songprevious  = { initial = "" } },
  execute = function(domoticz, item)
      -- timer every minute
      if (item.isTimer) then
            local urlRequest = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM)
          domoticz.log(urlRequest)
     domoticz.openURL({url = urlRequest, method = 'GET', callback = REQUEST_CALLBACK})
    end

    if (item.isHTTPResponse) then
             domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
        domoticz.log(string.format('Item data: %s', item.data))
     if (item.statusCode == 200 and item.callback == REQUEST_CALLBACK) then
       if (item.isJSON) then
           -- Pick the song artist & title
           local song = item.json.result.item
                     -- Check if there is a title
                     local songcurrent = "Nothing playing"
                     if (song.title ~= "") then
                         -- note the artist is the first entry (1) of the artist array
               songcurrent = string.format("%s:%s", song.artist[1], song.title)
                     end
                     domoticz.log(string.format("Song playing: %s", songcurrent))
                     -- check if the song has changed
                     if (songcurrent ~= domoticz.data.songprevious) then
                         domoticz.log(string.format("Song playing changed: %s to %s",
domoticz.data.songprevious, songcurrent))
                 -- Update the text device
               domoticz.devices(IDX_PLAYING).updateText(songcurrent)
               -- Update the data
               domoticz.data.songprevious = songcurrent
                     end
            end
            return
     else
                 domoticz.devices(IDX_PLAYING).updateText(item.statusText)
         domoticz.log('[ERROR] Handling the request:' .. item.statusText, domoticz.LOG_ERROR)
            return
     end
    end

     end
}
```

Domoticz Log

```
2020-05-23 11:22:00.642 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_timer: HTTPResponse: "KODIEXPLOREPLAYINGTIMER"
2020-05-23 11:22:00.643 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGTIMER, Status Code = 200
2020-05-23 11:22:00.643 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
54,"id":1531,"label":"LABEL","title":"TITLE","type":"song"}}}
2020-05-23 11:22:00.643 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 11:22:00.643 Status: dzVents: Info: Song playing changed: Nothing playing to ARTIST:TITLE
2020-05-23 11:22:00.662 Status: dzVents: Info: ------ Finished kodi_explore_playing_timer
```

Get Song Playing & Progress Timer

Display the current song playing with progress Percentage, Time, Total Time.
There are two JSON-RPC API commands required: GetItem for Artist:Title and GetProperties for Percentage, Time, Total Time.
The commands are initiated every minute by a timer starting with GetItem. If GetItem completes, GetProperties is initiated.
Note
The log of the devices (i.e. progress) can contain many entries if the player runs for a while. Clear the devices from time to time.

Challenge
The devices are updated ONLY every minute, because this is the highest frequency possible by the Domoticz event system.
It would be good to have a higher device update frequency, lets say every 10 seconds or even ad hoc.
See next example exploring solutions.

Device(s)	117,VirtualDevices,Kodi Song,General,Text,DATA 123,VirtualDevices,Player Progress,General,Text,DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playing_progress_timer.dzvents
Trigger	Timer every minute
Async HTTP request method	GET
JSON-RPC API Command	local JSONRPC_GETITEM = "{\"jsonrpc\":\"2.0\", \"method\":\"Player.GetItem\", \"params\":{\"properties\":[\"title\", \"album\", \"artist\", \"duration\"], \"playerid\":0}, \"id\":\"AudioGetItem\"}"
JSON-RPC API Command	local JSONRPC_GETPROPERTIES = "{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\" :{ \"playerid\": 0, \"properties\":[\"speed\", \"percentage\",\"time\", \"totaltime\"]},\"id\": \"AudioGetItem\"}"

Automation Event

```
--[[
-- kodi_explore_playing_progress_timer.dzvents
-- Get the current song playing and the progress percentage, time, totaltime from the Kodi Media Center
and display in two text devices.
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
-- Event unique callbacks
local REQUEST_GETITEM_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETITEM"
local REQUEST_GETPROPERTIES_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
-- Request for method Player.GetItem for artist:title
local JSONRPC_GETITEM =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GetItem\",\"params\":{\"properties\":[\"title\",\"album\",\"a
rtist\",\"duration\"],\"playerid\":0},\"id\":\"AudioGetItem\"}"
-- Request for method Player.GetProperties for the total time and time played for the current song
playing
local JSONRPC_GETPROPERTIES = '{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\"
:{ \"playerid\": 0, \"properties\":[\"speed\", \"percentage\",\"time\", \"totaltime\"]},\"id\":
\"AudioGetItem\"}'
```

```
-- Domoticz devices
local IDX_PLAYING = 117     -- text song playing artist:title
local IDX_PROGRESS = 123    -- percentage, time, totaltime

return {
  on = {
      timer = { 'every minute' },
      httpResponses = { REQUEST_GETITEM_CALLBACK, REQUEST_GETPROPERTIES_CALLBACK }
      },
    data = {
        songprevious  = { initial = "" }
        },
  execute = function(domoticz, item)
      -- timer every minute
      if (item.isTimer) then
     domoticz.openURL({
         url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM),
         method = 'GET',
         callback = REQUEST_GETITEM_CALLBACK})
   end

   if (item.isHTTPResponse) then
            domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
        domoticz.log(string.format('Item data: %s', item.data))
     if (item.statusCode == 200) then

         if (item.callback == REQUEST_GETITEM_CALLBACK and item.isJSON) then
                    --
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
54,"id":1531,"label":"LABEL","title":"TITLE","type":"song"}}}
           -- Pick the song artist & title
           local song = item.json.result.item
                    -- Check if there is a title
                    local songcurrent = "Nothing playing"
                    if (song.title ~= "") then
                        -- note the artist is the first entry (1) of the artist array
             songcurrent = string.format("%s:%s", song.artist[1], song.title)
                    end
                    domoticz.log(string.format("Song playing: %s", songcurrent))
                    -- check if the song has changed
                    if (songcurrent ~= domoticz.data.songprevious) then
                        domoticz.log(string.format("Song playing changed: %s to %s",
domoticz.data.songprevious, songcurrent))
                -- Update the text device
               domoticz.devices(IDX_PLAYING).updateText(songcurrent)
               -- Update the data
               domoticz.data.songprevious = songcurrent
                    end
            domoticz.openURL({
                url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETPROPERTIES),
                method = 'GET',
                 callback = REQUEST_GETPROPERTIES_CALLBACK})
            end

         if (item.callback == REQUEST_GETPROPERTIES_CALLBACK and item.isJSON) then
                    --
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":25.5638370513916,"speed":1,"time":{"hours":
0,"milliseconds":268,"minutes":1,"seconds":4},"totaltime":{"hours":0,"milliseconds":402,"minutes":4,"se
conds":11}}}
           -- init percentage with 0%
                    local progress = string.format("%s", "0%")
                    -- check seconds played
                    if (item.json.result.time.seconds > 0) then
                        -- set progress with percentage, time played and totaltime
               progress = string.format('%.1f%% / %02d:%02d:%02d / %02d:%02d:%02d',
                     item.json.result.percentage,
                     item.json.result.time.hours, item.json.result.time.minutes,
item.json.result.time.seconds,
                     item.json.result.totaltime.hours, item.json.result.totaltime.minutes,item.
json.result.totaltime.seconds
                     )
                     end
                     domoticz.devices(IDX_PROGRESS).updateText(progress)
                end
```

```
     else
                 domoticz.devices(IDX_PLAYING).updateText(item.statusText)
         domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
            return
     end
   end

    end
}
```

Domoticz Log

```
2020-05-23 12:27:00.228 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 12:27:00.363 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 12:27:00.363 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_progress: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 12:27:00.364 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETITEM, Status Code
= 200
2020-05-23 12:27:00.364 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ALTIST"],"duration":2
44,"id":1542,"label":"LABEL","title":"TITLE","type":"song"}}}
2020-05-23 12:27:00.364 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 12:27:00.364 Status: dzVents: Info: Song playing changed: ARTIST:TITLE to ARTIST:TITLE
2020-05-23 12:27:00.383 Status: dzVents: Info: ------ Finished kodi_explore_playing_progress
2020-05-23 12:27:00.383 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 12:27:00.510 Status: dzVents: Info: Handling events for: "Player Playing", value:
"ARTIST:TITLE"
2020-05-23 12:27:00.510 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_progress: Device: "Player Playing (VirtualDevices)", Index: 117
2020-05-23 12:27:00.511 Status: dzVents: Info: ------ Finished kodi_explore_playing_progress
2020-05-23 12:27:00.584 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 12:27:00.585 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_progress: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 12:27:00.586 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES,
Status Code = 200
2020-05-23 12:27:00.586 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":12.370973587036133,"speed":1,"time":{"hours
":0,"milliseconds":238,"minutes":0,"seconds":30},"totaltime":{"hours":0,"milliseconds":427,"minutes":4,
"seconds":4}}}
2020-05-23 12:27:00.604 Status: dzVents: Info: ------ Finished kodi_explore_playing_progress
2020-05-23 12:27:00.604 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

Get Song Playing & Progress afterSec

This example is based on the previous example updating the devices via timer every minute.
Challenge
The devices are updated ONLY every minute, because this is the highest frequency possible by the Domoticz event system.
It would be good to have a higher device update frequency, lets say every 10 seconds or even ad hoc.
How to do so - explore solutions

- Method afterSec to control a delay for the method openURL(). This solution is tested are running fine (used by the function Music_Player)
- Timer with a for loop (for var=start_value,end_value,step_value do GETITEM followed by GETPROPERTIES end) within the trigger. This solution is not worked out further <TODO>
- Listen to MQTT messages from other applications, i.e. Node-RED

Solution dzVents Method afterSec

Make use of the dzVents device method afterSec(NN) to control a delay for the method openURL().
Pseudocode

```
DELAYSECONDS = 15
Trigger openURL(GETITEM, CALLBACK_GETITEM)
Handle CALLBACK_GETITEM
  Run openURL(GETPROPERTIES, CALLBACK_GETPROPERTIES)
Handle CALLBACK_GETPROPERTIES
  Run openURL(GETITEM, CALLBACK_GETITEM).afterSec(DELAYSECONDS)
```

Notes

- The trigger could be a device change, i.e. setting a switch to ON (see below) or a selector to play.
- The change of the switch device triggers the first openURL for the JSON-RPC API GETITEM.
- The GETITEM callback runs openURL for the JSON-RPC API GETPROPERTIES.
- The GETPROPERTIES callback runs GETITEM again, but after a delay, set by the constant DELAYSECONDS to 15 seconds.
- After the delay, the loop starts again from the GETITEM callback.

This solution is used for the function Music Player.

Automation Event

```
--[[
-- kodi_explore_playing_progress_aftersec.dzvents
-- Get the current song playing and the progress percentage, time, totaltime from the Kodi Media Center
and display in two text devices.
-- The data is refreshed every 10 seconds handled by afterSec method initiated by a switch device.
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
-- Event unique callbacks
local REQUEST_GETITEM_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETITEM"
local REQUEST_GETPROPERTIES_CALLBACK = "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
-- Request for method Player.GetItem for artist:title
```

```
local JSONRPC_GETITEM =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.GetItem\",\"params\":{\"properties\":[\"title\",\"album\",\"a
rtist\",\"duration\"],\"playerid\":0},\"id\":\"AudioGetItem\"}"
-- Request for method Player.GetProperties for the total time and time played for the current song
playing
local JSONRPC_GETPROPERTIES = '{\"jsonrpc\": \"2.0\", \"method\": \"Player.GetProperties\", \"params\"
:{ \"playerid\": 0, \"properties\":[\"speed\", \"percentage\",\"time\", \"totaltime\"]},\"id\":
\"AudioGetItem\"}'

-- Refresh Artist:Title & Progress Rate every NN seconds - afterSec(NN)
local REFRESHRATE = 10

-- Domoticz devices
local IDX_PLAYING = 117         -- Text song playing artist:title
local IDX_PROGRESS = 123        -- Text percentage, time, totaltime
local IDX_REMOTECONTROL = 126   -- Switch to enable or disable the player

local MSG_PLAYERTURNEDOFF = "Player turned Off"
local MSG_PLAYERTURNEDON = "Player turned On"
local MSG_INITPROGRESS = "0%"

-- clear the various device logs to limit database size with not needed entries
local function clearDeviceLogs(domoticz)
    local ip = "http://domoticz-ip:8080/json.htm"
  local urlBase = string.format('%s?type=command&param=clearlightlog&idx=', ip)
    devices = {IDX_PLAYING, IDX_PROGRESS}
    for key,idx in ipairs(devices) do
        domoticz.openURL({url = string.format('%s%d', urlBase, idx), method = 'GET'})
    end
    domoticz.log("Device logs cleared.")
end


return {
  on = {
      devices = {
          IDX_REMOTECONTROL                    -- turn the player on or off
      },
      httpResponses = {
          REQUEST_GETITEM_CALLBACK,            -- get artist:title
          REQUEST_GETPROPERTIES_CALLBACK }     -- get percetage,time,totaltime
      },
    data = {
        songprevious  = { initial = "" }
        },
  execute = function(domoticz, item)

      -- Get the player remote control state
        playerRemoteControl = domoticz.devices(IDX_REMOTECONTROL).nValue

      -- Check device
      -- If ON run GETITEM request first time
      if (item.isDevice and item.idx == IDX_REMOTECONTROL) then
          domoticz.log(string.format("%s=%s|%d|%s",item.name,item.state,item.nValue,item.sValue))
            -- reset songprevious
            domoticz.data.songprevious = ""
            -- Player remote control turned OFF
            if (playerRemoteControl == 0) then
                domoticz.devices(IDX_PLAYING).updateText(MSG_PLAYERTURNEDOFF)
                domoticz.devices(IDX_PROGRESS).updateText(MSG_INITPROGRESS)
                -- OPTION to clear the log of the various devices
                clearDeviceLogs(domoticz)
            else
                -- Player remote control turned ON
                domoticz.devices(IDX_PLAYING).updateText(MSG_PLAYERTURNEDON)
                domoticz.devices(IDX_PROGRESS).updateText(MSG_INITPROGRESS)
        domoticz.openURL({
            url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM),
            method = 'GET',
             callback = REQUEST_GETITEM_CALLBACK})
            end
   end

   if (item.isHTTPResponse) then
            -- stop if playerRemoteControl is OFF
```

```
            if (playerRemoteControl == 0) then
                return
            end
            --
            domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
        domoticz.log(string.format('Item data: %s', item.data))
     if (item.statusCode == 200) then

         if (item.callback == REQUEST_GETITEM_CALLBACK and item.isJSON) then
                    --
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
54,"id":1531,"label":"LABEL","title":"TITLE","type":"song"}}}
           -- Pick the song artist & title
           local song = item.json.result.item
                    -- Check if there is a title
                    local songcurrent = "Nothing playing"
                    if (song.title ~= "") then
                        -- note the artist is the first entry (1) of the artist array
             songcurrent = string.format("%s:%s", song.artist[1], song.title)
                    end
                    domoticz.log(string.format("Song playing: %s", songcurrent))
                    -- check if the song has changed
                    if (songcurrent ~= domoticz.data.songprevious) then
                        domoticz.log(string.format("Song playing changed: %s to %s",
domoticz.data.songprevious, songcurrent))
                -- Update the text device
              domoticz.devices(IDX_PLAYING).updateText(songcurrent)
              -- Update the data
              domoticz.data.songprevious = songcurrent
                    end
            domoticz.openURL({
                url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETPROPERTIES),
                method = 'GET',
                  callback = REQUEST_GETPROPERTIES_CALLBACK})
            end

         if (item.callback == REQUEST_GETPROPERTIES_CALLBACK and item.isJSON) then
                    --
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":25.5638370513916,"speed":1,"time":{"hours":
0,"milliseconds":268,"minutes":1,"seconds":4},"totaltime":{"hours":0,"milliseconds":402,"minutes":4,"se
conds":11}}}
           -- init percentage with 0%
                    local progress = string.format("%s", "0%")
                    -- check seconds played
                    if (item.json.result.time.seconds > 0) then
                        -- set progress with percentage, time played and totaltime
              progress = string.format('%.1f%% / %02d:%02d:%02d / %02d:%02d:%02d',
                   item.json.result.percentage,
                   item.json.result.time.hours, item.json.result.time.minutes,
item.json.result.time.seconds,
                   item.json.result.totaltime.hours, item.json.result.totaltime.minutes,item.
json.result.totaltime.seconds
                   )
                    end
                    domoticz.devices(IDX_PROGRESS).updateText(progress)
                    -- Start next GETITEM request after NN seconds
            domoticz.openURL({
                url = string.format("%s?request=%s", REQUEST_URL, JSONRPC_GETITEM),
                method = 'GET',
                  callback = REQUEST_GETITEM_CALLBACK}).afterSec(REFRESHRATE)
                end
     else
                domoticz.devices(IDX_PLAYING).updateText(item.statusText)
         domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
            return
     end
    end

    end
}
```

Domoticz Log

```
2020-05-23 19:02:58.611 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 19:03:07.942 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 19:03:07.942 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_progress_aftersec: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETITEM"
2020-05-23 19:03:07.972 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETITEM, Status Code
= 200
2020-05-23 19:03:07.972 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"item":{"album":"ALBUM","artist":["ARTIST"],"duration":2
90,"id":2322,"label":"LABEL","title":"TITLE","type":"song"}}}
2020-05-23 19:03:07.972 Status: dzVents: Info: Song playing: ARTIST:TITLE
2020-05-23 19:03:07.973 Status: dzVents: Info: ------ Finished kodi_explore_playing_progress_aftersec
2020-05-23 19:03:07.973 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 19:03:08.093 Status: dzVents: Info: Handling httpResponse-events for:
"KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 19:03:08.093 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playing_progress_aftersec: HTTPResponse: "KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES"
2020-05-23 19:03:08.112 Status: dzVents: Info: Callback: KODIEXPLOREPLAYINGPROGRESSGETPROPERTIES,
Status Code = 200
2020-05-23 19:03:08.112 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"percentage":94.3416748046875,"speed":1,"time":{"hours":
0,"milliseconds":872,"minutes":4,"seconds":33},"totaltime":{"hours":0,"milliseconds":298,"minutes":4,"s
econds":50}}}
2020-05-23 19:03:08.114 Status: dzVents: Info: ------ Finished kodi_explore_playing_progress_aftersec
```

Solution dzVents for Loop

Timer with a for loop within the time trigger.
Pseudocode

```
for seconds_value = start_value, end_value, step_value do
  GETITEM.afterSec(seconds_value) followed by GETPROPERTIES
end
```

If the time triggeris set by every minute and the data to be updated every 10 seconds, then the for loop variables could be:
start_value=10, end_value=50, step_value=10. The seconds_value gets the 5 values 10,20,30,40,50.
This solution has not been worked out further <TODO>.

Solution MQTT with Node-RED

This solution uses Node-RED as the _man-in-the-middle_, i.e. between the player and Domoticz.

Domoticz to listen to MQTT messages published by Node-RED with topic "domoticz/in" (for incoming messages) and let Domoticz update the device data.
This requires the Domoticz hardware, like "MQTT Client with LAN Interface".

The MQTT messages published by Node-RED could either via PULL or PUSH:
PULL data from the player in regular intervals using the "inject" node with a trigger repeating every NN seconds submitting a HTTP request.
The HTTP request contains the JSON-RPC API command "Player.GetItem" as used previous (with some extra properties just to check out):

```
{"jsonrpc":"2.0","method":"Player.GetItem","params":{"properties":["title","album","artist","duration",
"thumbnail","file","fanart","streamdetails"],"playerid":0},"id":"AudioGetItem"}
```

OR

PUSH data from the player by creating a websocket using the "websocket in" node (port:9090 as configured in the player) and use JSON-RPC API callback method.
A configuration node "websocket-listener" acting as a WebSocket Server endpoint using the path: ws://player-ip:9090/jsonrpc and send/receice: payload.

Publishing of the message via a "mqtt out" node to the Domoticz topic "domoticz/in".

Payload example or the Domoticz device with idx 117 ("Player Playing") which receives as msg.payload the string "ARTIST:TITLE".
This msg.payload is extracted from the result of the JSON-RPC API Player.GetItem.

```
const IDX = 117;
const ARTISTTITLE = msg.payload; // ARTIST:TITLE
msg.payload = {"command": "udevice", "idx" : IDX, "nvalue" : 0, "svalue" : ARTISTTITLE}
```

Domoticz Log

```
2020-05-24 09:57:44.860 MQTT: Topic: domoticz/in, Message:
{"command":"udevice","idx":117,"nvalue":0,"svalue":"ARTIST:TITLE"}
```

Node-RED Flow Source: kodi_explore_node-red-domoticz-mqtt.flow

Set Player to Play or Pause

This is a POST request, to change the player play mode to PLAY or PAUSE.

Device(s)	119, VirtualDevices, Kodi PlayPause, Light/Switch, Switch, Selector, DATA (Idx,Hardware,Name,Type,SubType,Data)
Automation Event	kodi_explore_playback_device.dzvents
Trigger	Device change state with idx=119
Async HTTP request method	POST
JSON-RPC API Command	See automation event source local constants, like JSONRPC_PLAY, JSONRPC_PAUSE

Automation Event

```
--[[
-- kodi_explore_playpause_device.dzvents
-- Change the player play mode to PLAY or PAUSE using a selector switch
-- 20200523 rwbl
]]--

-- Define the url for the kodi jsonrpc get request (the parameter '?request=' is added)
local REQUEST_URL = "http://player-ip:8080/jsonrpc"
-- Event unique callback (=eventname in uppercase without underscores
local REQUEST_CALLBACK = "KODIEXPLOREPLAYPAUSE"
-- Command for method to set the player to play, pause
local JSONRPC_PLAY =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.PlayPause\",\"params\":{\"playerid\":0,\"play\":true},\"id\":
\"AudioGetItem\"}"
local JSONRPC_PAUSE =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.PlayPause\",\"params\":{\"playerid\":0,\"play\":false},\"id\"
:\"AudioGetItem\"}"
-- Domoticz device
local IDX_PLAYPAUSE = 119
local LEVEL_PLAY = 10
local LEVEL_PAUSE = 20
-- HINT: instead constants for the jsonrpc command and the device levels, a lua table could be used
local modePlayPause = {
    [10] =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.PlayPause\",\"params\":{\"playerid\":0,\"play\":true},\"id\":
\"AudioGetItem\"}",
    [20] =
"{\"jsonrpc\":\"2.0\",\"method\":\"Player.PlayPause\",\"params\":{\"playerid\":0,\"play\":false},\"id\"
:\"AudioGetItem\"}",
    }

return {
  on = {
      devices = { IDX_PLAYPAUSE },
      httpResponses = { REQUEST_CALLBACK }
      },
  execute = function(domoticz, item)
      -- check device
      if (item.isDevice and item.idx == IDX_PLAYPAUSE) then
            local jsonrpcCommand = ""
            -- set jsonrpc command using constants
            if (item.level == LEVEL_PLAY) then jsonrpcCommand = JSONRPC_PLAY end
            if (item.level == LEVEL_PAUSE) then jsonrpcCommand = JSONRPC_PAUSE end
            domoticz.log(string.format("CONST=Level selected: %d, %s", item.level, jsonrpcCommand))
            -- set jsonrpc command using lua table
            jsonrpcCommand = modePlayPause[item.level]
            domoticz.log(string.format("TABLE=Level selected: %d, %s", item.level, jsonrpcCommand))
          domoticz.openURL({
                url = REQUEST_URL,
                method = 'POST',
                headers = { ['content-type'] = 'application/json' },
                callback = REQUEST_CALLBACK,
```

```
                postData = jsonrpcCommand})
            domoticz.log(string.format("POST Request: %s, %s", REQUEST_URL, jsonrpcCommand))
   end

   if (item.isHTTPResponse) then
            domoticz.log(string.format('Callback: %s, Status Code = %s', item.callback,
item.statusCode))
        domoticz.log(string.format('Item data: %s', item.data))
     if (item.statusCode == 200 and item.callback == REQUEST_CALLBACK) then
         domoticz.log(string.format("[INFO] Handling request: %s", item.statusText))
     else
         domoticz.log(string.format("[ERROR] Handling request: %s", item.statusText),
domoticz.LOG_ERROR)
     end
   end
    end
}
```

Domoticz Log

The log shows the playpause mode set to pause = method Player.PlayPause with parameter play:false.

```
2020-05-23 11:52:45.448 Status: User: Admin initiated a switch command (119/Player PlayPause/Set Level)
2020-05-23 11:52:45.594 Status: dzVents: Info: Handling events for: "Player PlayPause", value: "Pause"
2020-05-23 11:52:45.594 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playpause_device: Device: "Player PlayPause (VirtualDevices)", Index: 119
2020-05-23 11:52:45.594 Status: dzVents: Info: CONST=Level selected: 20,
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":0,"play":false},"id":"AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: TABLE=Level selected: 20,
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":0,"play":false},"id":"AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: POST Request: http://player-ip:8080/jsonrpc,
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":0,"play":false},"id":"AudioGetItem"}
2020-05-23 11:52:45.595 Status: dzVents: Info: ------ Finished kodi_explore_playpause_device
2020-05-23 11:52:45.595 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-23 11:52:45.793 Status: dzVents: Info: Handling httpResponse-events for: "KODIEXPLOREPLAYPAUSE"
2020-05-23 11:52:45.793 Status: dzVents: Info: ------ Start internal script:
kodi_explore_playpause_device: HTTPResponse: "KODIEXPLOREPLAYPAUSE"
2020-05-23 11:52:45.794 Status: dzVents: Info: Callback: KODIEXPLOREPLAYPAUSE, Status Code = 200
2020-05-23 11:52:45.794 Status: dzVents: Info: Item data:
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"speed":0}}
2020-05-23 11:52:45.794 Status: dzVents: Info: [INFO] Handling request: OK
2020-05-23 11:52:45.794 Status: dzVents: Info: ------ Finished kodi_explore_playpause_device
```

Appendix: Kodi JSONRPC Command (Selection)

Note
Ensure to escape special characters with a backslash (\).
Test with curl prior using in scripts.

Example
This method can be used to check if the player is playing or paused, i.e. speed=1 or speed=0
Curl Command:

```
curl -H "Content-Type: application/json" --data-binary "{\"jsonrpc\": \"2.0\", \"method\":
\"Player.GetProperties\", \"params\" :{ \"playerid\": 0, \"properties\":[\"speed\", \"time\"]},\"id\":
\"AudioGetItem\"}"  http://player-ip:8080/jsonrpc
```

HTTP Response

```
{"id":"AudioGetItem","jsonrpc":"2.0","result":{"speed":1,"time":{"hours":0,"milliseconds":287,"minutes"
:0,"seconds":0}}}
```

Transport

Play pause:
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{ "playerid": 1},"id":1}
Play:
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":1,"play":true},"id":1}
Pause:
{"jsonrpc":"2.0","method":"Player.PlayPause","params":{"playerid":1,"play":false},"id":1}
Stop:
{"jsonrpc":"2.0", "method":"Player.Stop","params":{ "playerid":1},"id":1}
Next:
{"jsonrpc":"2.0","method":"Player.GoTo","params":{ "playerid":1,"to":"next"},"id":1}
Previous:
{"jsonrpc":"2.0","method":"Player.GoTo","params":{ "playerid":1,"to":"previous"},"id":1}
Ffwd with speed 2:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":2},"id":1}
Ffwd with speed 4:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":4},"id":1}
Ffwd with speed 8:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":8},"id":1}
Ffwd with speed 16:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":16},"id":1}
Ffwd with speed 32:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":32},"id":1}
Rwd with speed 2:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":-2},"id":1}
Rwd with speed 4:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":-4},"id":1}
Rwd with speed 8:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":-8},"id":1}
Rwd with speed 16:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":-16},"id":1}

Rwd with speed 32:
{"jsonrpc":"2.0","method":"Player.SetSpeed","params":{"playerid":1,"speed":-32},"id":1}
Jump forward:
{"jsonrpc":"2.0","method":"Player.Seek","params":{
"playerid":1,"value":"smallforward"},"id":1}
Jump backward:
{"jsonrpc":"2.0","method":"Player.Seek","params":{ "playerid":1,
"value":"smallbackward"},"id":1}

Repeat and Shuffle

Repeat one:
{"jsonrpc": "2.0","method":"Player.SetRepeat","params":{ "playerid":1,"repeat":"one"
},"id":1}
Repeat all:
{"jsonrpc":"2.0","method":"Player.SetRepeat","params":{ "playerid":1,"repeat":"all"
},"id":1}
Shuffle on:
{"jsonrpc":"2.0","method":"Player.SetShuffle","params":{"playerid":1,"shuffle":true},"id":1}
Shuffle false:
{"jsonrpc":"2.0","method":"Player.SetShuffle","params":{"playerid":1,"shuffle":false},"id":1
}
Repeat off:
{"jsonrpc":"2.0","method":"Player.SetRepeat","params":{ "playerid":1,"repeat":"off"
},"id":1}

Navigation

Back:
{"jsonrpc": "2.0", "method": "Input.Back", "id": 1}
Select:
{"jsonrpc": "2.0", "method": "Input.Select", "id": 1}
Left:
{"jsonrpc": "2.0", "method": "Input.Left", "id": 1}
Right:
{"jsonrpc": "2.0", "method": "Input.Right", "id": 1}
Up:
{"jsonrpc": "2.0", "method": "Input.Up", "id": 1}
Down:
{"jsonrpc": "2.0", "method": "Input.Down", "id": 1}
Home:
{"jsonrpc": "2.0", "method": "Input.Home", "id": 1}
Info:
{"jsonrpc": "2.0", "method": "Input.info", "id": 1}
Show OSD:
{"jsonrpc": "2.0", "method": "Input.showOSD", "id": 1}
Context Menu: {"jsonrpc": "2.0", "method": "Input.ContextMenu", "id": 1}
Show codec information of playing item:
{"jsonrpc": "2.0", "method": "Input.ShowCodec", "id": 1}

Page Down:
{"jsonrpc":"2.0","method":"input.executeaction","params":{"action":"pagedown"},"id":1}
Page Up:
{"jsonrpc":"2.0","method":"input.executeaction","params":{"action":"pageup"},"id":1}

Additional

Volume up:
{"jsonrpc":"2.0","method":"Application.SetVolume","params":{"volume":"increment"},"id":1 }
Volume down:
{"jsonrpc":"2.0","method":"Application.SetVolume","params":{"volume":"decrement"},"id":1 }
Get Playlists:
{"jsonrpc":"2.0","method":"Playlist.GetItems","params":{"properties":["title","album","artist","duration"],"playlistid":1},"id":1}
Get Albums:
{"jsonrpc":"2.0","method":"AudioLibrary.GetAlbums","params":{"properties":["playcount","artist","genre"],"limits":{"end":10,"start":0},"sort":{"order":"ascending","method":"album","ignorearticle":true}},"id":"libAlbums"}
Activate Genres:
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://1/"] },"id":1}
Activate artists:
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://2/"]},"id":1}
Activate albums:
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://3/"]},"id":1}
Activate songs:
{"jsonrpc":"2.0","method":"GUI.ActivateWindow","params":{"window":"music","parameters":["musicdb://4/"]},"id":1}

System

Reboot:
{"jsonrpc":"2.0","method":"System.Reboot","id":1}
Shutdown:
{"jsonrpc":"2.0","method":"System.Shutdown","id":1}

Appendix: Various Hints

Config.txt Modifications

LibreELEC is read-only by default (Info)

Modify config.txt via SSH:

```
ssh root@player-ip
```

Set write-permissions

```
mount -o remount,rw /flash
```

Edit config.txt

```
nano /flash/config.txt
```

Set readonly-permissions:

```
mount -o remount,ro /flash
```

Reboot for the changes to take effect.

Python

Kodi uses python 2.7
Test from CLI

```
python
Python 2.7.16 (default, Mar  5 2020, 07:14:57)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

RPi Button Shutdown & Restart

Add a push-button to enable shutdown and restart.
The push-button is connected to GPIO03 (=SCL pin #05). The dtoverlay gpio-shutdown is used.
Pressing the push-button once: shortens the GPIO3 (pin 5) to ground/0V, which is the same as the shutdown command. Both shutdown the OS and go into a low-power halt state.
Pressing the push-button again: the Raspberry restarts from the halt state.

NOTE: On the old RPi2 did not found way to use the push-button conncted to a pin other than GPIO3.
The GPIO3 pin is also the SCL pin, means not possible to connect a I2C device, i.e. LCD display.

Reference
Name: gpio-shutdown
Info: Initiates a shutdown when GPIO pin changes. The given GPIO pin is configured as an input key that generates KEY_POWER events. This event is handled by systemd-logind by initiating a shutdown. This overlay only handles shutdown. After shutdown, the system can be powered up again by driving GPIO3 low.

Steps
Wiring Push-Button
The push-button is connected to GPIO3 (=SCL, physical pin #05) and GND (=GND, physical pin #06).

Add dtoverlay to config.txt

```
ssh root@player-ip
# Set readwrite
mount -o remount,rw /flash
# Edit config.txt
nano /flash/config.txt
#Add line:
dtoverlay=gpio-shutdown,active_low=1,gpio_pull=up
Set readonly:
#mount -o remount,ro /flash
```

Set key map to avoid Kodi Shutdown Menu

```
nano /storage/.kodi/userdata/keymaps/keyboard.xml:
# Add
<keymap>
    <global>
        <keyboard>
            <power>shutdown</power>
        </keyboard>
    </global>
</keymap>
```

Reboot

```
shutdown -r 0
```

Check bootlog
This is to ensure the dtoverlay is defined properly

```
vcdbg log msg
```

RPi Addition Status LED

In addition to the push-button, a RED LED indicates the status of the Raspberry Pi running Kodi. LED on = Kodi is running.
The LED is turned on during boot and turned off during shutdown.

Steps
Wiring LED
The LED RED is connected to GPIO13 (=physical pin #33) and GND (=physical pin #34).

Status LED Python Script
Requires the Kodi Addon "Script - Raspberry Pi Tools"
(installed via GUI Addons > Search > Raspberry > select ""Script - Raspberry Pi Tools" > install).
The addon is installed in folder:

```
/storage/.kodi/addons/virtual.rpi-tools/
```

Python Script Location:

```
/storage/scripts/status_led.py
```

Python Script:

```
import sys
sys.path.append('/storage/.kodi/addons/virtual.rpi-tools/lib')
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
led_gpio_number = 13

# Use BCM pin numbering (i.e. the GPIO number, not pin number).
GPIO.setmode(GPIO.BCM)

# Switch LED on.
# LED off at shutdown by using GPIO state.
GPIO.setup(led_gpio_number, GPIO.OUT)

state = True
# state = False
GPIO.output(led_gpio_number, state)
```

Autostart Bash Script

Bash Script Location:

```
/storage/.config/autostart.sh
```

Bash Script:

```
#!/bin/sh
## Status led RED connected to gpio13
python /storage/scripts/status_led.py
```

Make Scripts Executable

```
chmod +x /storage/scripts/status_led.py
chmod +x /storage/.config/autostart.sh
```

MQTT

Purpose

To enable subscribing and publishing MQTT messages (see Wiki).
Ensure also to lookup the reference Domoticz API/JSON Url's.

Install

Raspberry Pi

Install mosquitto on the Raspberry Pi.

Open Terminal and run

```
sudo apt-get install mosquitto mosquitto-clients
```

Check if mosquitto is running:

```
mosquitto -h
```

Output

```
mosquitto version 1.4.10 (build date Fri, 22 Dec 2017 08:19:25 +0000)
mosquitto is an MQTT v3.1 broker.
```

Hint
The MQTT parameter to add the host to the mosquitto command is

```
-h localhost
```

Domoticz

Domoticz Add Hardware: Domoticz GUI > Setup > Hardware and add the “MQTT Client Gateway with LAN Interface”.

Property	Value
Type	MQTT Client Gateway with LAN Interface
Name	MQTT
Remote Address	domoticz-ip
Port	1883
Publish Topic	Out
Username	Empty
Password	Empty

Check the Domoticz Log: Domoticz GUI > Setup > Log

```
2018-08-27 12:21:12.126 Status: MQTT: Connecting to domoticz-ip:1883
2018-08-27 12:21:12.228 Status: MQTT: connected to: domoticz-ip:1883
2018-08-27 12:21:12.328 Status: MQTT: Subscribed
```

Domoticz Topics

Subscribe to Topics from Domoticz	domoticz/out
Publish a Topic to Domoticz	domoticz/in

Mosquitto

Test MQTT messaging using mosquitto commands from a terminal.

Subscribe

Command

Subscribe to all MQTT messages published by Domoticz (topic: domoticz/out):

```
mosquitto_sub -t 'domoticz/out'
```

Sample Messages

Some messages published by Domoticz in JSON format.
The source is from the devices provided by the Hardware Device Motherboard (idx=1 and 2):

☐	2	Motherboard	0001	1	Internal Temperature	Temp	LaCrosse TX3	42.4 C
☐	1	Motherboard	0000044D	1	CPU_Usage	General	Percentage	0.19%

For each of the messages the topic is *domoticz/out* and the payload is a JSON string.
To get a value from a key, the JSON string needs to be parsed.

CPU Usage

```
{
   "Battery" : 255,"RSSI" : 12,"description" : "","dtype" : "General","id" : "0000044D",
   "idx" : 1, "name" : "CPU_Usage", "nvalue" : 0, "stype" : "Percentage", "svalue1" : "0.19",
   "unit" : 1
}
```

CPU Temperature

```
{
   "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "Temp", "id" : "1",
   "idx" : 2, "name" : "Internal Temperature", "nvalue" : 0, "stype" : "LaCrosse TX3",
   "svalue1" : "42.4", "unit" : 1
}
```

Publish

Let'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

Command & Output

```
mkdir /home/pi/python
git clone https://github.com/eclipse/paho.mqtt.python.git
cd paho.mqtt.python
sudo python setup.py install
**Output***
mkdir python
git clone https://github.com/eclipse/paho.mqtt.python.git
Cloning into 'paho.mqtt.python'...
remote: Counting objects: 2932, done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 2932 (delta 27), reused 85 (delta 24), pack-reused 2834
Receiving objects: 100% (2932/2932), 835.76 KiB | 447.00 KiB/s, done.
Resolving deltas: 100% (1487/1487), done.
cd paho.mqtt.python
sudo python setup.py install
running install
…
Installed /usr/local/lib/python2.7/dist-packages/paho_mqtt-1.3.1-py2.7.egg
Processing dependencies for paho-mqtt==1.3.1
Finished processing dependencies for paho-mqtt==1.3.1
```

Pip (Python2) & Pip3 (Python3)

Pip Command & Output

```
pip install paho-mqtt
**Output***
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading
https://files.pythonhosted.org/packages/59/11/1dd5c70f0f27a88a3a05772cd95f6087ac479fac66d9c7752ee5e16dd
bbc/paho-mqtt-1.5.0.tar.gz (99kB)
Building wheels for collected packages: paho-mqtt
  Running setup.py bdist_wheel for paho-mqtt ... done
  Stored in directory:
/home/pi/.cache/pip/wheels/02/94/6c/8474137cb7a5a3e001d70a22c8ff919caee69435376bccce79
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.0
```

Pip3 Command & Output

```
pip3 install paho-mqtt
**Output***
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://www.piwheels.org/simple/paho-mqtt/paho_mqtt-1.5.0-py3-none-any.whl (61kB)
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.0
```

Python Subscribe Script

This Python sample script connects to the MQTT broker running on the Raspberry Pi Domoticz System, subscribes to the Domoticz Out topic ("domoticz/out") and filters the value for the device with Idx 39 and display the properties name & svalue … plus additional debug information.

Domoticz Device

```
Idx=39, Hardware=Raspberry Pi, Type=General, SubType=Percentage
```

Python3 Script

Source code: mqtt_sub_idx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
File:       mqtt_sub_idx.py
Project:    Domoticz-Home-Automation-Workbook
Purpose:    Subscribe to MQTT messages and filter device with Idx=39
Run:        python3 /home/pi/domoticz/scripts/python/mqtt_sub_idx.py
Version:    20200227 by rwbl
"""
import time
import paho.mqtt.client as mqtt
import json
BROKER = "localhost"
TOPIC = "domoticz/out"
IDX = 39

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(TOPIC)

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
    json_string = str(message.payload.decode("utf-8"))
    getidxvalue(json_string, IDX)

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

# JSON Parsing
def getidxvalue(json_string, idx):
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        parsed_name = str(parsed_json['name'])
        parsed_value = str(parsed_json['svalue1'])
        print(parsed_idx, parsed_name, parsed_value)

# Main
if __name__ == "__main__":
    print("Creating new instance")
    client = mqtt.Client("P1")
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_log = on_log
    print("Connecting to the broker", BROKER)
    client.connect(BROKER, 1883, 60)
    time.sleep(4)
    client.loop_forever()
```

Output Sample with lots of prints to show the steps & actions

```
python3 mqtt_sub_ix.py
creating new instance
connecting to broker
log:  Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
log:  Received CONNACK (0, 0)
Connected with result code 0
log:  Sending SUBSCRIBE (d0) [(b'domoticz/out', 0)]
log:  Received SUBACK
log:  Received PUBLISH (d0, q0, r0, m0), 'domoticz/out', ...  (229 bytes)
message received  {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "General",
   "id" : "0000044D",
   "idx" : 39,
   "name" : "CPU_Usage",
   "nvalue" : 0,
   "stype" : "Percentage",
   "svalue1" : "0.77",
   "unit" : 1
}

message topic= domoticz/out
message qos= 0
message retain flag= 0
39 CPU_Usage 0.77

message topic= domoticz/out
message qos= 0
message retain flag= 0
1 CPU_Usage 0.33
^CTraceback (most recent call last):
  File "mqtttest.py", line 54, in <module>
    client.loop_forever()
  File "/home/pi/.local/lib/python3.5/site-packages/paho/mqtt/client.py", line 1481, in loop_forever
    rc = self.loop(timeout, max_packets)
  File "/home/pi/.local/lib/python3.5/site-packages/paho/mqtt/client.py", line 988, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt
```

Python Publish Script

This sample script connects to the MQTT broker running on the Raspberry Pi Domoticz System and publishes text to a Virtual Device.
The Python script sets the text of the device to i.e. “Hello World at 2020-02-27 19:35:39”.

Domoticz Device

```
Idx=108, Hardware=VirtualDevices, Type General, SubType Text
```

Python3 Script

Source code: mqtt_pub_idx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
File:       mqtt_sub_idx.py
Project:    Domoticz-Home-Automation-Workbook
Purpose:    Publish a value for a Domoticz text device with idx=108
Run:        python3 /home/pi/domoticz/scripts/python/mqtt_pub_idx.py
Result:     Updated Domoticz text device with text i.e. Hello World at 2020-02-27 19:35:39
Version:    20200227 by rwbl
"""

import time
from datetime import datetime
import paho.mqtt.client as mqtt

BROKER = "localhost"
TOPIC = "domoticz/in"
IDX = 108
payload = '{"command": "udevice", "idx": %s, "svalue": "Hello World at %s"}' %
(IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))

# MQTT Callback Functions
def on_connect(client, userdata, flags, rc):
    print("on_connect: Client={},userdata={},flags={},rc={}".format(client, userdata, flags, rc))

def on_message(client, userdata, message):
    print("on_message: topic={}".format(message.topic))
    print("on_message: payload={}".format(message.payload.decode("utf-8")))
    print("on_message: qos={},retail=".format(message.qos,message.retain))

def on_log(client, userdata, level, buf):
    print("on_log:client={},userdata={},level=={},buf={}".format(client, userdata, level, buf))

# Main
if __name__ == "__main__":
    print("Creating new instance")
    client = mqtt.Client("P1")
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_log = on_log
    print("Connecting to the broker={}".format(BROKER))
    client.connect(BROKER, 1883, 60)
    time.sleep(4)
    print("Publishing message to topic={},payload=".format(TOPIC,payload))
    client.publish(TOPIC,payload)
    time.sleep(4)
    client.loop_stop()
    print("Disconnected from the broker={}".format(BROKER))
```

Output

```
python3 mqtt_pub_idx.py

Creating new instance
Connecting to the broker=localhost
```

```
on_log:client=<paho.mqtt.client.Client object at 0x766d4b70>,userdata=None,level==16,buf=Sending
CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
Publishing message to topic=domoticz/in,payload=
on_log:client=<paho.mqtt.client.Client object at 0x766d4b70>,userdata=None,level==16,buf=Sending
PUBLISH (d0, q0, r0, m1), 'b'domoticz/in'', ... (82 bytes)
Disconnected from the broker=localhost
```

Widget with updated text

Node-RED

Purpose

To explore Node-RED functionality for the Domoticz Home Automation system:

- Node-RED as alternative script engine, i.e. perform tasks, i.e. MQTT subscribe or publish
- Node-RED UI Dashboard Add-On as a simple monitor using Gauges & Charts or to control using buttons / slider / input-fields
- Trigger notifications or as watchdog for Domoticz processes

Node-RED = Low-code programming for event-driven applications.

If not familiar with the Node-RED concept, strongly recommend reading the Node-RED get started documentation.

Several Functions make use of Node-RED, like Volumio or Email Control and many other.

Node-RED runs on the same Raspberry Pi as the Domoticz Production & Development systems running Raspbian Buster.
In addition tests with Node-RED under Ubuntu running on a notebook has been carried out.

Install & Update

Installation & Update Command Linux

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

Important
Prior updating Node-RED, backup the flows file, located in

```
/home/pi/.node-red/
```

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

```
flows_DoPro.json
flows_DoDev.json
```

After updating, reboot the Raspberry Pi.

Read the Update Reference.

Update Log Example (Raspberry Pi)

```
This script will remove versions of Node.js prior to version 7.x, and Node-RED and if necessary replace
them with Node.js 10.x LTS (dubnium) and the latest Node-RED from Npm.

It also moves any Node-RED nodes that are globally installed into your user ~/.node-red/node_modules
directory, and adds them to your package.json, so that you can manage them with the palette manager.
```

```
It also tries to run 'npm rebuild' to refresh any extra nodes you have installed that may have a native
binary component. While this normally works ok, you need to check that it succeeds for your combination
of installed nodes.

To do all this it runs commands as root - please satisfy yourself that this will not damage your Pi, or
otherwise compromise your configuration. If in doubt please backup your SD card first.
Are you really sure you want to do this ? [y/N] ? y
Would you like to install the Pi-specific nodes ? [y/N] ? y
Running Node-RED install for user pi at /home/pi on raspbian
This can take 20-30 minutes on the slower Pi versions - please wait.

  Stop Node-RED                       ✔
  Remove old version of Node-RED      ✔
  Using N to manage Node.js           +
  Update Node.js LTS                  ✔   Node v12.14.0   Npm 6.13.4
  Clean npm cache                     ✔
  Install Node-RED core               ✔   1.0.3
  Move global nodes to local          -
  Install extra Pi nodes              ✔
  Npm rebuild existing nodes          -
  Add shortcut commands               ✔
  Update systemd script               ✔

Any errors will be logged to   /var/log/nodered-install.log
All done.
You can now start Node-RED with the command  node-red-start or using the icon under Menu / Programming
/ Node-RED
Then point your browser to localhost:1880 or http://{your_pi_ip}:1880

Started  Sat 21 Dec 10:10:17 CET 2019  -  Finished  Sat 21 Dec 10:12:28 CET 2019
```

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

To enable autostart of Node-RED at 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://domoticz-ip:1880).

Use disable, to disable starting Node-RED at autostart:

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

Install Add-Ons

It is strongly recommended to install Node-RED add-ons, via the Node-RED menu “Manage Palette”.

Access Flows & Dashboard UI

To access Node-RED, point a browser at URL:

```
http://localhost:1880 (when using a Raspberry Pi with a connected monitor)
```

or

```
http://domoticz-ip:1880 (remote)
```

This enables to manage Node-RED, define & run flows etc.

If the Dashboard add-on is installed, access the UI with its tabs vua URL:

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

Start, Stop, Log

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

Manage Node Packages

npm

To manage node packages in Node-RED, the npm package is required.
Check if installed:

```
npm -v
```

If package not found, then install via Desktop > Preferences > Add / Remove Software and restart Node-RED (terminal commands node-red-stop, node-red-start)

Check npm version:

```
npm -v
Output
6.4.1
```

Install/Updates

Important
To install or update any node package, Node-RED recommends via
Node-RED Menu Settings > Manage Palette > Install.

Note
If direct installed via npm, Node-RED might be missing dependencies.

After install/update, restart Node-RED (terminal commands node-red-stop, node-red-start).
It is not recommended to install using terminal commands.

Folder Locations

Data	Folder
Modules	/home/pi/.node-red
Flows	/home/pi/.node-red/ flows_dodev.json
Additional Node Packages	/home/pi/.node-red/node_modules/<package name>
Additional Node Packages: Sample Dashboard Nodes	/home/pi/.node-red/node_modules/node-red-dashboard

Example Flows

Log Domoticz MQTT Messages

A simple flow, to test Node-RED receiving data from Domoticz and get a key:value.

Node-RED Flow

Node mqtt in

Connects to the MQTT broker and subscribes to the MQTT topic “domoticz/out” to listen to all incoming Domoticz MQTT messages.
The MQTT Broker is localhost with default port 1883 because Node-RED and Domoticz are running on the same hardware (Raspberry Pi with Raspian Buster).
The output of the node is a parsed JSON object.

Node-RED Flow Source

Source: mqtt_listener_example.flow

```
[
    {
        "id": "fe01be83.5c17d","type": "mqtt in","z": "41e8885e.6f1a68",
        "name": "",
        "topic": "domoticz/out",
        "qos": "2", "datatype": "json",
        "broker": "de20bce4.03135", "x": 110, "y": 60,"wires": [["739fc80e.4b63f8"]]
    },
    {
        "id": "739fc80e.4b63f8", "type": "debug","z": "41e8885e.6f1a68",
        "name": "DEBUG MQTTDOPRO",
        "active": true,"tosidebar": true,"console": false,"tostatus": false, "complete": "payload",
        "targetType": "msg", "x": 350,"y": 60,"wires": []
    },
    {
        "id": "de20bce4.03135","type": "mqtt-broker","z": "",
        "name": "MQTT Broker DoPro",
        "broker": "localhost", "port": "1883", "clientid": "",
        "usetls": false, "compatmode": false, "keepalive": "60", "cleansession": true,
        "birthTopic": "","birthQos": "0","birthPayload": "",
        "closeTopic": "","closeQos": "0","closePayload": "",
        "willTopic": "","willQos": "0","willPayload": ""
    }
]
```

Output debug

Payload example (Hue Light with idx=118)
{ "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}
Access a key (case sensitive), convert the Domoticz message to a JSON object:
msg.payload.idx returns 118 msg.payload.Level returns 90

Log Raspberry Pi CPU Usage

Subscribe to MQTT topic “Domoticz/in” to listen to all Domoticz MQTT messages.
From the message payload (after conversion to JSON), select the property idx with value 1.
Then select the property svalue1 to get the CPU Usage.

Domoticz Topic “Domoticz/in” Message Payload for idx = 1


```
{"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"0000044D","idx":1,"name":"CPU_Usage",
"nvalue":0,"stype":"Percentage","svalue1":"0.28","unit":1}
```


Node-RED Flow

Node-RED Flow Source


```
[{"id":"f67403bf.ebdc88","type":"mqtt
in","z":"76cb1798.6a823","name":"","topic":"domoticz/out","qos":"2","broker":"d58d713d.0e0eb8","x":110,
"y":100,wires":[["db5fff92.295628"]]},{"id":"35214a2f.f89656","type":"debug","z":"76cb1798.6a823","nam
e":"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":"payloa
d","action":"","pretty":false,"x":270,"y":100,"wires":[["5f7a4a09.abb57c","13e0d06f.2b15d"]]},{"id":"5f
7a4a09.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","t
ype":"function","z":"76cb1798.6a823","name":"Get CPU Usage (from idx:5)","func":"// Get the property
idx from the payload\nidx = msg.payload.idx;\ncpu_usage=-1;\n// Check for idx 1\nif (idx == 5) {\n
// Get the cpu usage from property svalue1\n    cpu_usage = msg.payload.svalue1;\n    //
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":"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":"c030
082a.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.

Domoticz as Node-RED Data Logger

Log data from Node-RED and update Domoticz devices to access the charts for the devices.

This example is using the data provided by WASC, a tailor made Home Automation & Information System with a WebUI for use with smartphones.
WASC – Web-App Site Control created by the author - is an experimental solution based on Node-RED.

Concept
(Provider) The Node-RED flow

- builds a JSON data object from several WASC functions, i.e. weather, energy
- updates the JSON data object by subscribing to MQTT messages from the functions
- listens to HTTP GET requests and returns the JSON data object

HTTP GET Request to Node-RED WASC

```
http://node-red-ip:1880/wasc
```

HTTP Response


```
{
  "weather":
    {
      "temperature":12.3, "humidity":79, "windspeed":5, "windgust":12, "winddirection":"WSW",
      "windbearing":247, "airpressure":999,"timestamp":"09:32"
    },
  "energy":
    {
      "house":140,"makelab":0,"timestamp":"09:33"
    }
}
```

(Requestor) The HTTP response JSON data object is parsed to get the key values.

In Domoticz, an automation event using a dzVents Lua script places a GET request (function domoticz.openURL) in regular intervals and parses from the HTTP response, the key values to update the Domoticz devices.

The HTTP response item data is converted to a Lua table (example weather data wind):

```
-- HTTP reponse main keys
local KEY_WEATHER = 'weather';
-- Convert selected key into Lua table
local weatherData = item.json[KEY_WEATHER];
```

The variable “weatherData” contains a Lua table:

```
{
  ["timestamp"]="16:24", ["temperature"]=15.7, ["windgust"]=6, ["humidity"]=70, ["winddirection"]="N",
["windspeed"]=7, ["airpressure"]=1000, ["windbearing"]=0
}
```

To access keys and update the Domoticz device, use

```
Variable.key, i.e. weatherData.windbearing
```

```
-- 117,VirtualSensors,Windstärke,Wind,WTGR800,203;SSW;10;21.0;14.0;14.0
local IDX_WIND = 117;

domoticz.devices(IDX_WIND).updateWind(weatherData.windbearing, weatherData.winddirection,
weatherData.windspeed / 3.6, weatherData.windgust / 3.6, weatherData.temperature,
weatherData.temperature);
```

Node-Red Flow

The flow is an extract of the WASC-Interface flow.

Node http in

This is a HTTP end-point for the web service listening to a GET request
(method: GET, URL: /wasc).
The GET request uses the IP address of the system running Node-RED (which is the same as the Domoticz Production system in the authors setup on the Raspberry Pi running Raspian Buster).

```
http://node-red-ip:1880/wasc
```

Node change

The msg.payload is set with the JSON object from the flow context “wascdata”.
The JSON object has nested key:value pairs based on the function. In next example, the functions “weather” and “energy” are used:

Example of the flow context “wascdata” JSON object:


```
{
  "weather":
    {"temperature":-1,"humidity":-1,"windgust":-1,"winddirection":"","airpressure":-1,"timestamp":""},
  "energy":
    {"house":210,"makelab":0,"timestamp":"09:28"}
}
```


Node http response
Sends the response back to request received from the HTTP Input node.
The msg.payload is the wascdata as JSON formatted string – as shared previous.

Domoticz
See the concept earlier.

The Domoticz devices are based on the hardware VirtualSensors (Dummy (Does nothing, use for virtual switches only).
For this solution, several virtual sensors are created – to mention:

```
(Idx,Hardware,Name,Type,SubType,Data)
117,VirtualSensors,Windstärke,Wind,WTGR800,203;SSW;10;21.0;14.0;14.0
116,VirtualSensors,Luftdruck,General,Barometer,999 hPa
300,VirtualSensors,Außentemperatur,Temp + Humidity, THGN122/123/132, THGR122/228/238/268, 0.0 C, 50 %
171,VirtualSensors,Energie Haus,General,kWh,2250.074 kWh
```

Domoticz Dashboard
Roomplan: WASC_Data_Logger with the devices Idx 117,116,30,171.
Settings UI: Language: English, Theme: simple-gray, Dashboard: Mobile, Mobile: Mobile

Example Chart Device Aussen

The devices are updated via the automation event dzVents Lua script:
Event name: wasc_data_logger

```
local URL_DOMOTICZ = 'http://domoticz-ip:8080'
local URL_WASC_DATA = 'http://node-red-p:1880/wasc';
local RES_WASC_DATA = 'RES_WASC_DATA';
local KEY_WEATHER = 'weather';
local KEY_ENERGY = 'energy';
local IDX_WIND = 117;
local IDX_AIRPRESSURE = 116;
local IDX_OUTDOORTEMPHUM = 300;
local IDX_ELECTRICUSAGEHOUSE = 171  -- Type=General, SubType=kWh
return {
  on = {timer = { 'every minute' },httpResponses = { RES_WASC_DATA, }},
  execute = function(domoticz, item)
    if (item.isTimer) then
      domoticz.openURL({url = URL_WASC_DATA, method = 'GET', callback = RES_WASC_DATA,})
    end
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        local weatherData = item.json[KEY_WEATHER];
        domoticz.devices(IDX_WIND).updateWind(weatherData.windbearing, weatherData.winddirection,
weatherData.windspeed / 3.6, weatherData.windgust / 3.6, weatherData.temperature,
weatherData.temperature);
        local status = -1; -- not used
        domoticz.devices(IDX_OUTDOORTEMPHUM).updateTempHum(weatherData.temperature,
weatherData.humidity, status);
        local forcast = -1; -- not used
       domoticz.devices(IDX_AIRPRESSURE).updateBarometer(weatherData.airpressure, forecast)
        local energyData = item.json[KEY_ENERGY];
        local power = math.floor(energyData.house);
        domoticz.openURL({url = string.format(
'%s/json.htm?type=command&param=udevice&idx=%d&nvalue=0&svalue=%d;0', URL_DOMOTICZ,
IDX_ELECTRICUSAGEHOUSE, power), method = 'GET'})
        return
      else
        domoticz.log('[ERROR]:' .. item.statusText, domoticz.LOG_ERROR)
       return
      end
    end
 end
}
```

Domoticz Log Entries

```
2020-05-01 10:00:03.726 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-01 10:01:00.529 Status: dzVents: Info: ------ Start internal script: wasc_data_logger:,
trigger: "every minute"
2020-05-01 10:01:00.529 Status: dzVents: Info: ------ Finished wasc_data_logger
2020-05-01 10:01:00.530 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
2020-05-01 10:01:00.639 Status: dzVents: Info: Handling httpResponse-events for: "RES_WASC_DATA"
2020-05-01 10:01:00.640 Status: dzVents: Info: ------ Start internal script: wasc_data_logger:
HTTPResponse: "RES_WASC_DATA"
2020-05-01 10:01:00.640 Status: dzVents: Info: Status Code = 200, Name=RES_WASC_DATA,
Data={"weather":{"temperature":12.2,"humidity":77,"windspeed":3,"windgust":5,"winddirection":"WSW","win
dbearing":247,"airpressure":999,"timestamp":"09:57"},"energy":{"house":242,"makelab":0,"timestamp":"09:
57"}}
2020-05-01 10:01:00.655 Status: dzVents: Info: Windstärke: 0.8, 12.2
2020-05-01 10:01:00.656 Status: dzVents: Info: Aussen: 12.2, 77.0
2020-05-01 10:01:00.657 Status: dzVents: Info: Luftdruck: 999
2020-05-01 10:01:00.658 Status: dzVents: Info: Energie Haus: Wh = Actual: 242, Today: 2188, Total:
2255916
2020-05-01 10:01:00.658 Status: dzVents: Info: ------ Finished wasc_data_logger
```

Example Dashboard UI Flows

The Node-RED Dashboard module offers UI nodes to build a Dashboard.

Monitor Raspberry Pi CPU Usage

Build a simple example of a dashboard using the Raspberry Pi CPU Usage information, which is displayed in a Gauge and Line Chart.

Dashboard

Node-RED Flow

The Function Node selects idx 5 and builds message payload holding the CPU Usage in %.

```
// Get the property idx from the payload
idx = msg.payload.idx;
cpu_usage=-1;
// Check for idx 1
if (idx == 5) {
    // Get the cpu usage from property svalue1
    cpu_usage = msg.payload.svalue1;
    // node.warn("Idx=1, CPU Usage=" + cpu_usage);
    // Assign the cpu usage to the message payload
    msg.payload = cpu_usage;
    // Return the message
    return msg;
}
```

Node-RED Flow Source


```
[{"id":"f67403bf.ebdc88","type":"mqtt
in","z":"76cb1798.6a823","name":"","topic":"domoticz/out","qos":"2","broker":"d58d713d.0e0eb8","x":110,
"y":240,"wires":[["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;\ncpu_usage=-1;\n//
Check for idx 1\nif (idx == 5) {\n    // Get the cpu usage from property svalue1\n    cpu_usage =
msg.payload.svalue1;\n    // 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_Usa
ge","group":"c3b6865e.2b93e","order":0,"width":0,"height":0,"gtype":"gage","title":"","label":"%","form
at":"{{value}}","min":0,"max":"100","colors":["#00b500","#e6e600","#ca3838"],"seg1":"","seg2":"","x":82
0,"y":300,"wires":[]},{"id":"d5bc4ce8.ae1f28","type":"ui_chart","z":"76cb1798.6a823","name":"ui_chart_C
PU_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","r
emoveOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"colors":[
"#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useOldStyle
":false,"x":820,"y":340,"wires":[[],[]]},{"id":"d58d713d.0e0eb8","type":"mqtt-
broker","z":"","name":"Domoticz MQTT
Broker","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"
60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","closeQos":"0"
,"closePayload":"","willTopic":"","willQos":"0","willPayload":""},{"id":"c3b6865e.2b93e","type":"ui_gro
up","z":"","name":"CPU","tab":"8a1743ab.9d931","disp":true,"width":"6","collapse":false},{"id":"8a1743a
b.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","nam
e":"DEBUG switch
move","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","x":800,"y":
140,"wires":[]},{"id":"db5fff92.295628","type":"json","z":"76cb1798.6a823","name":"","property":"payloa
d","action":"","pretty":false,"x":270,"y":100,"wires":[["5f7a4a09.abb57c"]]},{"id":"5f7a4a09.abb57c","t
ype":"switch","z":"76cb1798.6a823","name":"Switch Payload on Property
idx","property":"payload.idx","propertyType":"msg","rules":[{"t":"eq","v":"5","vt":"num"}],"checkall":"
true","repair":false,"outputs":1,"x":370,"y":140,"wires":[["912289d2.e8c1"]]},{"id":"912289d2.e8c1","ty
pe":"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":400,"y":180,"wires":[["35214a2f.f89656","61fef7cc.4fc0
2","d5bc4ce8.ae1f28"]]},{"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":800,"y":180,"wires":[]},{"id":"d5bc4ce8.ae1f28","type":"ui_chart","z":"76cb1798.6a823","name":"ui_
chart_CPU_Usage","group":"c3b6865e.2b93e","order":0,"width":0,"height":0,"label":"","chartType":"line",
"legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":true,"ymin":"0","ymax":"
100","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"co
lors":["#1f77b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useO
ldStyle":false,"x":800,"y":220,"wires":[[],[]]},{"id":"d58d713d.0e0eb8","type":"mqtt-
broker","z":"","name":"Domoticz MQTT
Broker","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"
60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","closeQos":"0"
,"closePayload":"","willTopic":"","willQos":"0","willPayload":""},{"id":"c3b6865e.2b93e","type":"ui_gro
up","z":"","name":"CPU","tab":"8a1743ab.9d931","disp":true,"width":"6","collapse":false},{"id":"8a1743a
b.9d931","type":"ui_tab","z":"","name":"Raspberry Pi","icon":"dashboard"}]
```

Virtual Sensor Text Update

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

Widget showing the updated text

Node-RED Dashboard UI with ui_text_input Node

Node-RED Flow with Nodes ui_text_input > function > mqtt out

The text input node has a delay of 0, means the content is sent after pressing enter.
The function node created the new message which is published via MQTT using the localhost broker.

```
// Build a new message
var msginfo = {};
msginfo.topic = "domoticz/in";

// Get the text from the incoming payload from the text input node.
text = msg.payload;

// Build the new payload
msginfo.payload = { "idx" : 52, "nvalue": 0, "svalue": text};
node.warn(msginfo.payload);

// Return the message
return msginfo;
```

The information for building the MQTT payload is taken from the Domoticz API/JSON URL’s documentation (here).

Node-RED Flow Source


```
[{"id":"38d6f42f.539a94","type":"ui_text_input","z":"76cb1798.6a823","name":"New
Message","label":"Enter New
Message","group":"27ca36a9.4c3cba","order":0,"width":0,"height":0,"passthru":false,"mode":"text","delay
":"0","topic":"","x":120,"y":820,"wires":[["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\ntext = 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;\n","outputs":1,"noerr":0,"x":290,"y":820,"wires":[["d4be5444.33e23","f167bf33.2066e8"]]},{"id"
:"f167bf33.2066e8","type":"debug","z":"76cb1798.6a823","name":"DEBUG new
message","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"true","x":500,"y":
880,"wires":[]},{"id":"27ca36a9.4c3cba","type":"ui_group","z":"","name":"Hue","tab":"8a1743ab.9d931","d
isp":true,"width":"6","collapse":false},{"id":"d58d713d.0e0eb8","type":"mqtt-
broker","z":"","name":"Domoticz MQTT
Broker","broker":"localhost","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"
60","cleansession":true,"birthTopic":"","birthQos":"0","birthPayload":"","closeTopic":"","closeQos":"0"
,"closePayload":"","willTopic":"","willQos":"0","willPayload":""},{"id":"8a1743ab.9d931","type":"ui_tab
","z":"","name":"Raspberry Pi","icon":"dashboard"}]
```

Plugin Development

Purpose

The Python Plugin Framework allows to create an interface between Hardware (or Virtual Hardware) and Domoticz.

Developed various Python Plugins mainly interacting with TinkerForge building blocks:

- Soil Moisture Monitor **(**Info)
- Traffic Light **(**Info)

Must state, it makes real fun to see how new hardware devices with according devices are created and used.

Mandatory to consult this manual as reference.

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

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 tín 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 Setpup > 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'.

OR
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) Device creation failed, Domoticz settings prevent
accepting new devices.
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) 'onStart' failed 'KeyError'.
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) ----> Line 307 in '/home/pi/domoticz/plugins/hmip-
etrv/plugin.py', function onStart
2019-12-14 14:21:53.070 Error: (MakeLab Thermostat) ----> Line 126 in '/home/pi/domoticz/plugins/hmip-
etrv/plugin.py', function onStart
```

TabError

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

```
2019-05-24 11:07:17.254 Status: (Air Quality Livingroom) Started.
2019-05-24 11:07:17.424 Error: (AirQualityMonitor) failed to load 'plugin.py', Python Path used was ...
2019-05-24 11:07:17.424 Error: (Air Quality Livingroom) Module Import failed, exception: 'TabError'
2019-05-24 11:07:17.425 Error: (Air Quality Livingroom) Import detail: File:
/home/pi/domoticz/plugins/airqualitymonitor/plugin.py, Line: 165, offset: 92
2019-05-24 11:07:17.425 Error: (Air Quality Livingroom) Error Line ' iaq_index, iaq_index_accuracy,
temperature, humidity, air_pressure = ab.get_all_values()
```

SyntaxError

```
2019-05-26 11:24:47.046 Error: (TrafficLight) failed to load 'plugin.py', Python Path used was
'/home/pi/domoticz/plugins/TrafficLight/:/usr/lib/python35.zip:/usr/lib/python3.5:/usr/lib/python3.5/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

```
2019-06-04 14:58:29.215 Error: (CDevice_update) AQM - IAQ Index: Failed to parse parameters: 'nValue',
'sValue', 'Image', 'SignalLevel', 'BatteryLevel', 'Options', 'TimedOut', 'Name', 'TypeName', 'Type',
'Subtype', 'Switchtype', 'Used', 'Description', 'Color' or 'SuppressTriggers' expected.
2019-06-04 14:58:29.215 Error: (AQM) 'CDevice_update' failed 'TypeError':'Required argument 'sValue'
(pos 2) not found'.
```

Cause
Caused by missing sValue parameter for Air Quality update (Devices[1].Update).

```
Devices[1].Update(nValue=iaq_index)
```

Fix
Add empty sValue parameter.
Update the value - only nValue is used, but mandatory to add an sValue

```
Devices[1].Update(nValue=iaq_index, sValue="")
```

Initial Value New Device

To ensure a new device as an initial value, add update after device create.
Example code snippet from homematicIP HmIP-eTRV plugin.

```
class BasePlugin:

    def __init__(self):
…
        self.SetPoint = 0       # setpoint in C
        self.Temperature = 0    # actual temperature
        self.LowBat = "true"    # low battery "true" or "false"
        self.Level = 0          # valve position 0 - 100%
…

    def onStart(self):
        # if there no devices, create these and set initial value
        if (len(Devices) == 0):
            Domoticz.Debug("Creating new devices ...")
            ## SET_POINT_TEMPERATURE - TypeName: Thermostat
            Domoticz.Device(Name="Setpoint", Unit=UNITSETPOINTTEMPERATURE, Type=242, Subtype=1,
Used=1).Create()
            Devices[UNITSETPOINTTEMPERATURE].Update( nValue=1, sValue= str(self.SetPoint) )
            Domoticz.Debug("Device created: "+Devices[UNITSETPOINTTEMPERATURE].Name)

            ## ACTUAL_TEMPERATURE - TypeName: Temperature
            Domoticz.Device(Name="Temperature", Unit=UNITACTUALTEMPERATURE, TypeName="Temperature",
Used=1).Create()
            Devices[UNITACTUALTEMPERATURE].Update( nValue=0, sValue=str(self.Temperature) )
            Domoticz.Debug("Device created: "+Devices[UNITACTUALTEMPERATURE].Name)

            ## LOW_BAT - TypeName: Alert
            Domoticz.Device(Name="Battery", Unit=UNITLOWBAT, TypeName="Alert", Used=1).Create()
            Devices[UNITLOWBAT].Update( nValue=1, sValue=LOWBATMSGOK )
            Domoticz.Debug("Device created: "+Devices[UNITLOWBAT].Name)

            ## LEVEL - TypeName: Percentage
            Domoticz.Device(Name="Valve", Unit=UNITLEVEL, TypeName="Percentage", Used=1).Create()
            Devices[UNITLEVEL].Update( nValue=0, sValue=str(self.Level) )
            Domoticz.Debug("Device created: "+Devices[UNITLEVEL].Name)

…
```

Extract Value XPath

Get a value from an XML tree.
Requires the Python library lxml XML toolkit.
The Python library lxml XML toolkit is used to parse the XML-API response.

Install

```
sudo apt-get install python3-lxml
```

Note
If the plugin is added to the Domoticz Hardware without lxml installed, then delete the hardware prior installing lxml. After installing lxml, restart Domoticz: sudo service domoticz.sh restart

Python Script Import

```
from lxml import etree
```

Example

Get the value from XML tree entry - taken from function Radiator Thermostat HmIP-eTRV (Plugin).

```
<datapoint type="ACTUAL_TEMPERATURE" ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
timestamp="1577027367" valueunit="" valuetype="4" value="19.600000"/>
```

XPath Statement

```
actualtemperaturevalue = tree.xpath('//datapoint[@ise_id=1566]/@value')
```

used below with variable.

Returned is an array with a single entry as the datapoint ise_id is unique, i.e. to get the value use actualtemperaturevalue[0] after ensuring the array length is 1.

```
def onMessage(self, Connection, Data):
        Domoticz.Debug("onMessage called")

        # If not conected, then leave
        if self.httpConnected == 0:
            return

        # Parse the JSON Data Object with keys Status (Number) and Data (ByteArray)
        ## 200 is OK
        responseStatus = int(Data["Status"])
        ## Domoticz.Debug("STATUS="+ str(responseStatus))

        ## decode the data using the encoding as given in the xml response string
        responseData = Data["Data"].decode('ISO-8859-1')
        ## Domoticz.Debug("DATA=" + responseData)

        # Parse the xml string
        # Get the xml tree - requires several conversions
        tree = etree.fromstring(bytes(responseData, encoding='utf-8'))
        …
        # Get the value for datapoint actual_temperature & update the device and log
        ## note that a list is returned from tree.xpath, but holds only 1 value
        actualtemperaturevalue = tree.xpath('//datapoint[@ise_id=' +
                                self.DatapointsList[DATAPOINTINDEXACTUALTEMPERATURE] + ']/@value')
```

```
            if len(actualtemperaturevalue) == 1:
                Domoticz.Debug("T Act=" + actualtemperaturevalue[0])
                ## convert the raspberrymatic value to float
                atv = float(actualtemperaturevalue[0])
```

Plugin Traffic Light (Tinkerforge Building Blocks)

Sample building a Domoticz Plugin using Tinkerforge Building Blocks.

Objectives

- To set the color of a Tinkerforge RGB Bricklet to red, yellow or green using a Domoticz Selector Switch Device.
- To learn how to write a Python Plugin for the Domoticz system.
- To learn how to interact with Tinkerforge building blocks.
- To use this sample plugin as a template for other Domoticz plugins interacting with Tinkerforge Building Blocks.

Solution

A Domoticz Python plugin "Traffic Light" with a Tinkerforge RGB LED Bricklet.
The Tinkerforge RGB LED Bricklet is connected to a Tinkerforge Master Brick with WiFi extension.

Hardware

- Raspberry Pi 3B+ (Info)
- Tinkerforge Master Brick 1.1 (Info)
- Tinkerforge WIFI Master Extention 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/ttyACMo, Fiirmware: Master (2.4.10)
9. Flash
10. Master Brick reboots > Blue LED turns on and the Brick Viewer shows tabs Brick and Bricklets

Traffic Light Prototype

Build the prototype by connecting the Tinkerforge building blocks (see hardware).
Connect the Master Brick to a device running the Brick Deamon and Viewer.

Summary steps to setup the Tinkerforge building blocks using the Tinkerforge Brick Viewer.

- Update the devices firmware
- Set the WiFi master extension fixed IP address in client mode
- Obtain the UID's of the Tinkerforge bricklets as required by the Python plugin

After setting up the Tinkerforge building blocks, reset the master brick and check if the master brick can be reached via WLAN:

```
ping tf-wifi-ext-ip
```

Domoticz Plugin Folder & File

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

Folder:

```
mkdir /home/pi/domoticz/plugins/trafficlight
```

File:
As a starter, take the template from here.
Save as **plugin.py** in the folder /home/pi/domoticz/plugins/trafficlight

Tinkerforge API Bindings Path

In the Python Plugin code **plugin.py** amend the import path to enable using the Tinkerforge API Bindings:

```
from os import path
import sys
sys.path
sys.path.append('/usr/local/lib/python3.5/dist-packages')
```

Note
Use **pip3** to install the bindings in a common dist-packages folder, which is on the Raspberry Pi Domoticz system, folder:

```
/usr/lib/python3/dist-packages
```

Running pip3:

```
sudo pip3 install tinkerforge
```

Log Output Sample

```
Collecting tinkerforge
Installing collected packages: tinkerforge
Successfully installed tinkerforge-2.1.22
```

The Tinkerforge Python API bindings are installed in folder:

```
/usr/local/lib/python3.5/dist-packages
```

Check the content results in two folders, from which the folder tinkerforge is required

```
ls /usr/local/lib/python3.5/dist-packages
tinkerforge tinkerforge-2.1.22.dist-info
```

Important

Depending setup of the Python distributed packages, following steps are required.
Check the folder /usr/local/lib/python3.5/dist-packages if there are more distributed packages.
If thats the case, then leave the next steps out.

Copy the content of the folder /usr/local/lib/python3.5/dist-packages/tinkerforge to folder /usr/lib/python3/dist-packages/tinkerforge,
because all the Python distributed packages are in folder /usr/lib/python3/dist-packages/:

```
sudo cp -r /usr/local/lib/python3.5/dist-packages/tinkerforge /usr/lib/python3/dist-packages
```

Check the content of the dist-packages folder tinkerforge:

```
ls /usr/lib/python3/dist-packages/tinkerforge
brick_dc.py bricklet_led_strip.py brick_hat.py bricklet_led_strip_v2.py ...
```

Remove the dist-packages folders:

```
sudo rm -r /usr/local/lib/python3.5/dist-packages/tinkerforge
sudo rm -r /usr/local/lib/python3.5/dist-packages/tinkerforge-2.1.22.dist-info
ls /usr/local/lib/python3.5/dist-packages
```

Development Setup

Development PC:

- A shared drive Z: is defined pointing to /home/pi/domoticz
- Domoticz GUI > Setup > Log
- Domoticz GUI > Setup > Hardware
- Domoticz GUI > Setup > Devices
- WinSCP session connected to the Domoticz system
- Putty session connected to the Domoticz system

The browser tabs are required to add the new hardware with its device and monitor if the plugin code is running without errors.

Development Iteration

The development process step:

1. Editor to develop z:\plugins\trafficlight\plugin.py – *Note* z: is shared Samba drive.
2. Make changes and save plugin.py
3. Restart Domoticz from Terminal: sudo service domoticz.sh restart
4. Wait a moment and refresh the browser tab Domoticz GUI > Setup > Log
5. Check the log and fix as required

Important
In the Domoticz GUI > Setup > Settings, enable accepting new hardware.
This is required to add the new hardware with its device and monitor if the plugin code is running without errors.

Domoticz GUI's

In a browser, opened three tabs Domoticz GUI

1. Setup > Hardware
2. Setup > Log
3. Setup > Devices

This is required to add the new hardware with its device and monitor if the plugin code is running without errors.

Create the plugin

The plugin has a mandatory filename **plugin.py** located in the created plugin folder /home/pi/domoticz/plugins/trafficlight.
For Python development used Idle, running on a Windows 10 device.

See more, in the well documented Python Plugin source code **plugin.py**.

Domoticz Devices are created and set as used by the plugin.

```
Domoticz.Device(Name="State Selector", Unit=1, TypeName="Selector Switch", Options=Options,
Used=1).Create()
Domoticz.Device(Name="Alert Indicator", Unit=2, TypeName="Alert", Used=1).Create()
Domoticz.Device(Name="Status", Unit=3, TypeName="Text", Used=1).Create()
```

The devices are manually added to the Domoticz Dashboard.

Handling the state change of the Traffic Light Selector Switch is done by the **onCommand** function.
This function updates the state of the Domoticz Devices: State Selector (idx=13), Alert Indicator (idx=14), Status (idx=15).

Plugin Pseudo Code

FIRST TIME: Function **onStart** to create the Domoticz Devices

NEXT TIME(S): Function **onCommand** to handle state changes

1. Domoticz make IP connection to the Tinkerforge Master Brick
   If error, update Domoticz Devuce “Status” and return
2. Get the Level of the Domoticz Device "State Selector" (Selector Switch)
   If error, update Domoticz Devuce “Status” and return
3. Update the Tinkerforge Bricklet RGB LED with the new Color depending Level
4. Update the Domoticz Devices "Alert Indicator" and "Status"
5. Domoticz to disconnect from the Tinkerforge Master Brick

Note
Function **onHeartbeat** not used as not polling for the state of a Tinkerforge Bricklet and update Domoticz Device(s) accordingly.

Restart Domoticz

Restart Domoticz to find the plugin:

```
sudo systemctl restart domoticz.service
```

Note
When making changes to the Python plugin code, ensure to restart Domoticz and refresh any of the Domoticz GUI 's.

Domoticz Add Hardware Traffic Light

Prior adding set in the Domoticz Settings the option to allow new hardware.
If this option is not enabled, no new soilmoisture device is created.
Check in the Domoticz log as error message Python script at the line where the new device is used i.e. Domoticz.Debug("Device created: "+Devices[1].Name).

Add Hardware - Check the Domoticz Log
After adding, ensure to check the Domoticz Log (Domoticz GUI, select tab Setup > Log)
Example:

```
2019-05-27 09:43:31.992 Status: (Traffic Light) Started.
2019-05-27 09:43:32.550 (Traffic Light) Debug logging mask set to: PYTHON PLUGIN QUEUE IMAGE DEVICE
CONNECTION MESSAGE ALL
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzHash':'fccd39bb'
2019-05-27 09:43:32.550 (Traffic Light) 'HardwareID':'7'
2019-05-27 09:43:32.550 (Traffic Light) 'Database':'/home/pi/domoticz/domoticz.db'
2019-05-27 09:43:32.550 (Traffic Light) 'Version':'1.0.0'
2019-05-27 09:43:32.550 (Traffic Light) 'HomeFolder':'/home/pi/domoticz/plugins/TrafficLight/'
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzVersion':'4.10826'
2019-05-27 09:43:32.550 (Traffic Light) 'Name':'Traffic Light'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode6':'Debug'
2019-05-27 09:43:32.550 (Traffic Light) 'DomoticzBuildTime':'2019-05-24 10:04:40'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode4':'100'
2019-05-27 09:43:32.550 (Traffic Light) 'StartupFolder':'/home/pi/domoticz/'
2019-05-27 09:43:32.550 (Traffic Light) 'Port':'4223'
2019-05-27 09:43:32.550 (Traffic Light) 'Key':'TrafficLight'
2019-05-27 09:43:32.550 (Traffic Light) 'Language':'en'
2019-05-27 09:43:32.550 (Traffic Light) 'Address':'domoticz-ip'
2019-05-27 09:43:32.550 (Traffic Light) 'Author':'rwbl'
2019-05-27 09:43:32.550 (Traffic Light) 'UserDataFolder':'/home/pi/domoticz/'
2019-05-27 09:43:32.550 (Traffic Light) 'Mode1':'zMF'
2019-05-27 09:43:32.550 (Traffic Light) Device count: 0
2019-05-27 09:43:32.550 (Traffic Light) Creating new Devices
```

```
2019-05-27 09:43:32.551 (Traffic Light) Creating device 'State Selector'.
2019-05-27 09:43:32.552 (Traffic Light) Device created: Traffic Light - State Selector
2019-05-27 09:43:32.552 (Traffic Light) Creating device 'Alert Indicator'.
2019-05-27 09:43:32.553 (Traffic Light) Device created: Traffic Light - Alert Indicator
2019-05-27 09:43:32.553 (Traffic Light) Creating device 'Status'.
2019-05-27 09:43:32.554 (Traffic Light) Device created: Traffic Light - Change Info
2019-05-27 09:43:32.554 (Traffic Light) Pushing 'PollIntervalDirective' on to queue
2019-05-27 09:43:32.554 (Traffic Light) Processing 'PollIntervalDirective' message
2019-05-27 09:43:32.554 (Traffic Light) Heartbeat interval set to: 60.
2019-05-27 09:43:32.547 Status: (Traffic Light) Entering work loop.
2019-05-27 09:43:32.548 Status: (Traffic Light) Initialized version 1.0.0, author 'rwbl'
```

Domoticz Log 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" : "",   "dtype" : "Light/Switch",   "id" :
"00050001",
"idx" : 13,   "name" : "Traffic Light - State Selector",
"nvalue" : 2,   "stype" : "Selector Switch",   "svalue1" : "30",   "switchType" : "Selector",  "unit" :
1
}
```

RaspberryMatic

Purpose

- To explore RaspberryMatic and how to control HomeMatic compatible devices via Domoticz.
- To setup a RaspberryMatic operating system running a HomeMatic Central-Control-Unit (CCU) interfacing with Domoticz.
- To create several functions using homematicIP devices, like Pluggable Switch and Meters (HmIP-PSM), Radiator Thermostats (HmIP-eTRV-B & HmIP-eTRV-2), Window/Door Contacts (HmIP-SWDM & HmIP-SWDO=.
  Note: only homematicIP devices are used.

Notes
As taken from here: The RaspberryMatic project is a collaborate effort to provide a lightweight, Linux/buildroot-based Homematic compatible operating system for embedded single board computers (SBC) like the RaspberryPi or Tinkerboard.

Solution

The solution runs the RaspberryMatic CCU on a Raspberry Pi 3B+ with an RPI-RF-MOD GPIO Radio Module HAT (~40EUR).

Installation

RaspberryMatic

The Raspberry Pi 3B+ has been setup according these guidelines.

RaspberryMatic version 3.47.22.20191130 is used (check out for newer versions).

Several Addons installed:

Addon XML-API

Information

The XML-API CCU Addon is used to communicate between the CCU and Domoticz.

Installed via the RaspberryMatic WebUI > Home page > Settings > Control panel > Additional Software.
After installation, no further settings required.
The option *set*, lists the available scripts with their parameter.

Understanding the XML-API concept is essential to communicate between Domoticz and the CCU v.v. = keep the reference at hand.

XML-API Scripts

There is a variety of scripts available (see the XML-API documentation).
These are used by the automation scripts with relevant parameters.

Examples are

- Lists all devices with channels and current values: statelist.cgi
  This is a key script to get id's required by automation script to get or set values.
  Example:
  http://ccu-ip/addons/xmlapi/statelist.cgi
- List channels and values for single or multiple devices (1234,5678): state.cgi
  Example:
  http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541.
- Changes one or more channel states: statechange.cgi
  Example:
  http:// ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=20

Communication Flow

Domoticz triggers an HTTP XML-API request (with parameters depending script) to the CCU and parses the HTTP response in XML format using XPath.

Note

In the next examples, the wording “push” and “pull” are used:

- CCU to push data to Domoticz (i.e. device data)
- Domoticz to pull data from the CCU (i.e. device value read)
- Domoticz to push data to the CCU (i.e. device value change)

Example List Devices

To be able to communicate between Domoticz and the CCU, the device id's and datapoint id's are used These can be obtained via the statelist.cgi script.

HTTP XML-API Request

```
http:// ccu-ip/addons/xmlapi/statelist.cgi
```

HTTP Response (extract)

The response is an XML formatted list with all devices, their channels and datapoints with values.
The device name is also listed in the HomeMatic WebUI > Settings > Devices > Click on a device entry shows the general devices settings which enables to change the device name.

Each device has a unique serial number.
The default device name is "device type serial number", i.e. "HmIP-eTRV-2 000A18A9A64DAC".
For this device, changed the name to "Radiator Thermostat MakeLab".

```
<?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/addons/xmlapi/state.cgi?device_id=1541.
2. The CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents Lua script handles the callback, parses XML string using XPath to get the values of the setpoint & temperature by reading their datapoints.
4. dzVents Lua script updates the Domoticz text device with idx=175.

HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.
The response lists the datapoints to pick, i.e. use in the automation script (dzVents Lua).

HTTP URL

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

HTTP Response (extract)

The datapoints required are the SET_POINT_TEMPERATURE (ise_id=1584) and the ACTUAL_TEMPERATURE (ise_id=1567).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <device config_pending="false" unreach="false" ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
      ...
      <datapoint ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE" timestamp="1575647467"
valueunit="" valuetype="4" value="2.800000" type="OPERATING_VOLTAGE"/>
      <datapoint ise_id="1554" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS"
timestamp="1575647467" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
      ...
    </channel>
    <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
      ...
      <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1575647467"
valueunit="" valuetype="4" value="19.300000" type="ACTUAL_TEMPERATURE"/>
      <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_MODE" timestamp="1575647467"
valueunit="" valuetype="16" value="1" type="SET_POINT_MODE"/>
      <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
timestamp="1575647467" valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
...
    </channel>
...
  </device>
</state>
```

Automation Script

Event name: thermostat_makelab_info

```
-- Domoticz device text (named MakeLab Thermostat Info)
local IDX_MAKELAB_THERMOSTAT_INFO = 50;
local ID_DEVICE = 1541;
-- url of the raspmatic webserver to obtain device information
local URL_RASPMATIC = 'http:// ccu-ip/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
-- callback of the url request - must be unique across all automation events
local RES_RASPMATIC = 'RES_makelab_thermostat_info';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
  on = {
      timer = {
          -- 'every minute'   -- for tests
          'every 5 minutes'
        },
   httpResponses = {
     RES_RASPMATIC -- must match with the callback passed to the openURL command
   }
  },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({url = URL_RASPMATIC, method = 'GET', callback = RES_RASPMATIC})
    end

   -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        -- multiple datapoints - domoticz.log(item.data);
        -- parse the response using XPath
        -- select the attribute value of the datapoint element (this is XPath syntax)
        -- SETPOINT °C
        local setpointvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
        -- TEMPERATURE °C
        local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')

        -- log
        local msg = 'Setpoint: ' .. tostring(round(setpointvalue,DECIMALS)) .. ', Temperature: ' ..
tostring(round(temperaturevalue,DECIMALS))
        domoticz.log(msg)

        -- update the domoticz device
        domoticz.devices(IDX_MAKELAB_THERMOSTAT_INFO).updateText(msg);

      else
        domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

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/addons/xmlapi/state.cgi?datapoint_id=1567.
2. The CCU sends a HTTP response, in XML format, to Domoticz system.
3. dzVents Lua script handles the callback, parses XML string using XPath to get the values of the temperature datapoint.
4. dzVents Lua script updates the Domoticz temp device with idx=65.

Domoticz Device
Created as Virtual Sensor Type Temperature, listed as device type Temp, SubType LaCrosse TX3.

HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.
The response lists the datapoint to pick, i.e. use in the automation script (dzVents Lua).

HTTP URL

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=1567
```

HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="22.400000" ise_id="1567"/>
</state>
```

Automation Script

Event name: thermostat_makelab_temp

```
local IDX_MAKELAB_THERMOSTAT_TEMP = 65;
local ID_DATAPOINT = 1567;
-- url of the raspmatic webserver to obtain device information
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=' .. ID_DATAPOINT;
-- callback of the url request - must be unique across all automation events
local RES_RASPMATIC = 'RES_makelab_thermostat_temp';
-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
  local power = 10^decimals
  return math.floor(number * power) / power
end

return {
  on = {
      timer = {
          -- 'every minute'   -- for tests
          'every 5 minutes'
        },
     httpResponses = {
       RES_RASPMATIC -- must match with the callback passed to the openURL command
    }
  },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({url = URL_RASPMATIC, method = 'GET', callback = RES_RASPMATIC})
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then

      if (item.statusCode == 200) then
        -- <datapoint value="22.400000" ise_id="1567"/>
       local temperaturevalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1567"]/@value')
       -- log
       local msg = 'Temperature: ' .. tostring(round(temperaturevalue,DECIMALS))
       domoticz.log(msg)
        -- update the domoticz device
        domoticz.devices(IDX_MAKELAB_THERMOSTAT_TEMP).updateTemperature(round(temperaturevalue,1));
      else
        domoticz.log('[ERROR] Handling request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end

  end
}
```

Note

By opening multiple URL's with unique callback string, it is possible to obtain multiple datapoints. This could be used to check the battery status of several homematicIP devices.

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 URL:

```
http://ccu-ip/addons/xmlapi/state.cgi
```

```
<datapoint ise_id="2543" name="HmIP-RF.0000DA498D5859:0.OPERATING_VOLTAGE" operations="5"
timestamp="1575914670" valueunit="" valuetype="4" value="1.500000" type="OPERATING_VOLTAGE"/>
```

HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.
The response lists the datapoint information value and id, to be used in the automation script .

HTTP URL

```
http://ccu-ip/addons/xmlapi/state.cgi?datapoint_id=2543
```

HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  <datapoint value="1.400000" ise_id="2543"/>
</state>
```

Automation Script

Event name: postbox_voltage.lua

```
--[[
  postbox_voltage.lua
  Check the voltage of the homematicIP device SWDO built into the postbox.
  Log if voltage is below theshold.
]]--
local URL_RASPMATIC = 'http:// ccu-ip/addons/xmlapi/state.cgi?datapoint_id=';
JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()  -- For Linux
local DATAPOINT2543 = JSON:decode('{"name":"Postbox
Voltage","id":2543,"xpath":"//datapoint[@ise_id=2543]/@value","response":"RESPOSTBOXVOLTAGE"}');
-- 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
  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: "RESPOSTBOXVOLTAGE"
2019-12-12 09:46:00.362 Status: dzVents: Info: ------ Start internal script: postbox_voltage:
HTTPResponse: "RESPOSTBOXVOLTAGE"
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.

HTTP XML-API Request Test

Test in a browser the HTTP URL and the HTTP response.
The HTTP URL used is the same as the device level action defined.
This example request changes the setpoint to 18°C for datapoint=1584.
The response confirms the changed id and the new value.

HTTP URL

```
http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=18
```

HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<result>
  <changed id="1584" new_value="18"/>
</result>
```

Solution Automation Script

Instead device level actions, an option is to use an Automation Script triggered by device changes (the level name is used as the new setpoint), which opens the URL (HTTP XML-API request) with a HTTP callback.
The callback handles the response.
The Automation Script could also handle multiple devices.

Automation Script

```
-- makelab_thermostat_setpoint
-- set the setpoint of the radiator thermostat channel HmIP-eTRV-2 000A18A9A64DAC:1
-- set a new setpoint via url example:
-- http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=1584&new_value=17
-- response:
-- <result><changed id="1584" new_value="17.0"/></result>

-- Domoticz device idx
local IDX_MAKELAB_THERMOSTAT_SETPOINT = 49;

-- raspmatic datapoint id taken from statelist
-- <datapoint type="SET_POINT_TEMPERATURE" ise_id="1584" name="HmIP-
RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7" timestamp="1575709506" valueunit="°C"
valuetype="4" value="20.000000"/>
local ID_DATAPOINT_SETPOINT = 1584;

-- url of the raspmatic webserver to set the new setpoint
local URL_RASPMATIC = 'http:// ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' ..  ID_DATAPOINT_SETPOINT
.. '&new_value='

-- callback of the url request - must be unique across all dzevents - use prefix res + script name
local RES_RASPMATIC = 'res_makelab_thermostat_setpoint';

-- helper to round a number to n decimals
local DECIMALS = 1;

function round(number, decimals)
    local power = 10^decimals
    return math.floor(number * power) / power
end

return {
  on = {
    devices = {
      IDX_MAKELAB_THERMOSTAT_SETPOINT
    },
```

```
    httpResponses = {
      RES_RASPMATIC
    }
  },
  execute = function(domoticz, item)
    -- check if item is device to trigger setpoint change using the levelname (0,17,18,...)
    if (item.isDevice) then
      domoticz.log('Device ' .. domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).name .. ' was changed
to ' .. tostring(domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName), domoticz.LOG_INFO)
      -- get the new setpoint from the levelname
      -- 0%=0(OFF) ; 10%-40%=18-21 ; min=18, max=21
      local level = domoticz.devices(IDX_MAKELAB_THERMOSTAT_SETPOINT).levelName;
      local newsetpoint = tonumber(level);
      -- set the new setpoint
      domoticz.log('New setpoint:' .. URL_RASPMATIC .. tostring(newsetpoint), domoticz.LOG_INFO);
      domoticz.openURL({url = URL_RASPMATIC ..tostring(newsetpoint), method = 'POST', callback =
RES_RASPMATIC});
    end

        -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
          -- the full http xml response: domoticz.log(item.data);
          -- parse the response using XPath
          -- select the attribute value of the changed element (this is XPath syntax)
          -- <changed id="1584" new_value="18"/>
        domoticz.log('CCU Response new value: ' ..
domoticz_applyXPath(item.data,'//changed[@id="1584"]/@new_value'))
        -- The response could also be logged to an alert or control message device to log changes
      else
        domoticz.log('[ERROR] handling HTTP request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

Domoticz Log

The log shows the HTTP request and the HTTP response.

```
2019-12-07 11:48:54.970 (VirtualDevices) Light/Switch (MakeLab Thermostat Setpoint)
2019-12-07 11:48:54.963 Status: User: Admin initiated a switch command (49/MakeLab Thermostat
Setpoint/Set Level)

2019-12-07 11:48:55.103 Status: dzVents: Info: Handling events for: "MakeLab Thermostat Setpoint",
value: "19"
2019-12-07 11:48:55.103 Status: dzVents: Info: ------ Start internal script:
makelab_thermostat_setpoint: Device: "MakeLab Thermostat Setpoint (VirtualDevices)", Index: 49
2019-12-07 11:48:55.103 Status: dzVents: Info: Device MakeLab Thermostat Setpoint was changed to 19
2019-12-07 11:48:55.103 Status: dzVents: Info: New setpoint:http://ccu-
ip/addons/xmlapi/statechange.cgi?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/addons/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 system is used.

1. The RaspMatic script "HmIP-eTRV-2-MakeLab" immediately runs if the following rule applies:
   "Channel status: HmIP-eTRV-2 000A18A9A64DAC:1 when Actual temperature within value range / with value from 1.00 and less than 0.00 trigger when changed"
2. A script reads the actual state of the datapoint "HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE", builds & submits the URL of the Domoticz API request to update the value of the Domoticz device "MakeLab Thermostat Temperature" (idx=65). The HTTP response is logged.

Note
The solution is rather device bound, means for every device 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: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 *within value range / with value* from 1.00 and less than 0.00 trigger when changed	Script: ... immediately run

RaspberryMatic Script: thermostat_makelab_temp.script

```
! Function: Thermostat MakeLab Notify Actual Temperature change
! 20191208 by rwbl
string function = "Thermostat MakeLab Temperature";
string actdate = system.Date("%d.%m"); ! sDate = "09.08";
! string actDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string acttime = system.Date("%H:%M"); ! sTime = "07:32";
! string actime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";

! A domoticz virtual temperature device is used
string idx = 65; ! Type: Temp; SubType: LaCrosse TX3
! Define the parameter nvalue and svalue - see domoticz api documentation
string nvalue = 0;
! The object string is taken from the raspmatic XML-API script statelist.cgi
string svalue = dom.GetObject("HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE").State();
! for tests: string svalue = 20;

! Build the domoticz http api request url to set the actual temperature on the domoticz device
! Example: http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=20.89
string amp = "&";
string urldom = "'http://domoticz-ip:8080/json.htm"; ! Production=60,Development=179
string urlrequest =
urldom#"?type=command"#amp#"param=udevice"#amp#"idx="#idx#amp#"nvalue="#nvalue#amp#"svalue="#svalue#"'"
;
! For Tests.
! Output'http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=65&nvalue=0&svalue=21.700000'
WriteLine(urlrequest);

! OPTION: Run the command without a return result
! res = dom.GetObject("CUxD.CUX2801001:1.CMD_EXEC").State("wget -q -O - "#sUrl);
! OPTION: Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#urlrequest);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: The script running on the CCU waits until the completion - ensure not to execute commands which
take long time.
string res = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("VT="#res.VarType()#"/"#res); ! VT=4, {"status" : "OK","title" : "Update Device"}

! handle result
var domlog = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (res.Find("OK") > 0) {
  domlog.Variable(function#"-Last update OK:"#actdate#" "#acttime )
}
else {
  domlog.Variable(function#"-Last update ERROR:"#actdate#" "#acttime)
};
! WriteLine("VT="#domlog.VarType());  ! VT=9
WriteLine(domlog.Variable());       ! shows the last update string
```

RasperryMatic 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 Geräte 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.

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

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

Resulting in:


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
 - <device sticky_unreach="false" unreach="false" ise_id="1596" name="HM-RC-19 CUX2801001">
    - <channel ise_id="1597" name="HM-RC-19 CUX2801001:0" operate="" visible="" index="0">
        <datapoint ise_id="1598" name="CUxD.CUX2801001:0.LOWBAT" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="LOWBAT"/>
        <datapoint ise_id="1606" name="CUxD.CUX2801001:0.UNREACH" operations="5" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="UNREACH"/>
        <datapoint ise_id="1602" name="CUxD.CUX2801001:0.STICKY_UNREACH" operations="7" timestamp="1562428172" valueunit="" valuetype="2" value="false" type="STICKY_UNREACH"/>
      </channel>
    - <channel ise_id="1610" name="HM-RC-19 CUX2801001:1" operate="true" visible="true" index="1">
        <datapoint ise_id="1628" name="CUxD.CUX2801001:1.PRESS_SHORT" operations="6" timestamp="1562482685" valueunit="" valuetype="2" value="false" type="PRESS_SHORT"/>
        <datapoint ise_id="1627" name="CUxD.CUX2801001:1.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
        <datapoint ise_id="1612" name="CUxD.CUX2801001:1.CMD_KILL" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_KILL"/>
        <datapoint ise_id="1611" name="CUxD.CUX2801001:1.CMD_EXEC" operations="2" timestamp="0" valueunit="" valuetype="20" value="" type="CMD_EXEC"/>
        <datapoint ise_id="1634" name="CUxD.CUX2801001:1.WORKING" operations="5" timestamp="1562428036" valueunit="" valuetype="2" value="false" type="WORKING"/>
      </channel>
    - <channel ise_id="1636" name="HM-RC-19 CUX2801001:2" operate="true" visible="true" index="2">
        <datapoint ise_id="1654" name="CUxD.CUX2801001:2.PRESS_SHORT" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_SHORT"/>
        <datapoint ise_id="1653" name="CUxD.CUX2801001:2.PRESS_LONG" operations="6" timestamp="0" valueunit="" valuetype="2" value="" type="PRESS_LONG"/>
```

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.

RaspMatic Script
Submit Domoticz HTTP API Request via system command wget, to update the Domoticz Text Device.

```
wget -q -O - URL
```

The URL is the HTTP API Request as defined per Domoticz HTTP API/JSON documentation.

```
http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=iDX&nvalue=0&svalue=MESSAGE
```

```
string sDate = system.Date("%d.%m.%Y"); ! sDate = "09.08.2019";
string sTime = system.Date("%H:%M:%S"); ! sTime = "07:32:00";
string nIdx = 36;
string sMsg = sDate # " " # sTime # " - Threshold reached! Take action...";
string cAmp = "&";
string sDomUrl = "'http://domoticz-ip:8080/json.htm";
string sUrl =
sDomUrl#"?type=command"#cAmp#"param=udevice"#cAmp#"idx="#nIdx#cAmp#"nvalue=0"#cAmp#"svalue="#sMsg#"'";
! Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);
! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);
! Start the command, wait till completed and get the result JSON string, i.e.
! {"status" : "OK","title" : "Update Device"}
! NOTE: script running on CCU waits until completion – use commands which do not take long time.
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_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("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

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 woud be, to develop for each of the homematicIP devices a Domoticz Python Plugin.

After various iterations, developed plugins for the homematicIP devices Pluggable Switch and Meter (HMIP-PSM) and Radiator Thermostat (HMIP-eTRV-2).

These can be found in my GitHub repositories starting with domoticz-plugin-hmip-<short description>, i.e. domoticz-plugin-hmip-psm, domoticz-plugin-hmip-etrv-2.

More under consideration …

CCU Device(s) Information

All Devices List

HTTP request to obtain the list of all devices.

```
http://ccu-ip/addons/xmlapi/devicelist.cgi
```

Example response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<deviceList>
  <device ready_config="true" device_type="HmIP-eTRV-2" interface="HmIP-RF" ise_id="1541"
address="000A18A9A64DAC" name="HmIP-eTRV-2 000A18A9A64DAC">
    <channel ready_config="true" ise_id="1542" address="000A18A9A64DAC:0" name="HmIP-eTRV-2
000A18A9A64DAC:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
group_partner="" index="0" parent_device="1541" direction="UNKNOWN" type="30"/>
…
  </device>

  <device ready_config="true" device_type="HMIP-PSM" interface="HmIP-RF" ise_id="1418"
address="0001D3C99C6AB3" name="HMIP-PSM 0001D3C99C6AB3">
    <channel ready_config="true" ise_id="1419" address="0001D3C99C6AB3:0" name="HMIP-PSM
0001D3C99C6AB3:0" operate="true" visible="true" transmission_mode="AES" aes_available="false"
...
```

Notes

The example response extract lists two devices

- HmiP-eTRV-2 (device id=1541)
- HMIP-PSM (device id=1418)

with their device id and serial number (address).

All Devices State Information

To get detailed information about the state of all the devices, submit

```
http://ccu-ip/addons/xmlapi/statelist.cgi
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device ise_id="1011" name="HM-RCV-50 BidCoS-RF">
<device ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false" unreach="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0" operate="true" visible="true" index="0">
    <datapoint ise_id="1543" name="HmIP-RF.000A18A9A64DAC:0.CONFIG_PENDING" type="CONFIG_PENDING"
operations="5" timestamp="1561796713" valueunit="" valuetype="2" value="false"/>
    …
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1" operate="true" visible="true" index="1">
    <datapoint ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE" type="ACTIVE_PROFILE"
operations="7" timestamp="1561796714" valueunit="" valuetype="16" value="1"/>
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
type="ACTUAL_TEMPERATURE" operations="5" timestamp="1561796714" valueunit="" valuetype="4"
value="22.900000"/>
```

Notes

The response lists for each device the channels and the datapoints with unique ids.

Single Device Information

Submit HTTP request to obtain detailed information for a single device, i.e. the thermostat.

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
<device ise_id="1541" name="HmIP-eTRV-2 000A18A9A64DAC" config_pending="false" unreach="false">
  <channel ise_id="1542" name="HmIP-eTRV-2 000A18A9A64DAC:0">
    ...
    <datapoint ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" type="LOW_BAT"
timestamp="1561799020" valueunit="" valuetype="2" value="false"/>
    ...
  </channel>
  <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
    ...
    <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE"
type="ACTUAL_TEMPERATURE" timestamp="1561799020" valueunit="" valuetype="4" value="22.900000"/>
    ...
    <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE"
type="SET_POINT_TEMPERATURE" timestamp="1561799020" valueunit="°C" valuetype="4" value="4.500000"/>
    ...
  </channel>
  ...
</device>
</state>
```

Notes
The response lists all the device datapoints, which can be used to obtain information or change a value via HTTP URL XML-API request.
These requests are used by the Domoticz dzVents Lua scripts.
To test, submit the HTTP URL and check the response on a browser, but also in the HomeMatic Web-UI.

CCU Devices homematicIP

To get started with homematicIP (controlled by RaspberryMatic via Domoticz), these three devices are used – for details see the respective functions.

List

Device	Function	Notes
Pluggable Switch and Meter (PSM) Product ref.: 140666A0 Device ID: 1418	• 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	• Room MakeLab • Control room temperature • Measure room temperature	
Window / Door Contact – optical (SWDO) Product ref.: 140733A0 Device ID: 2530	• Postbox Watcher • Notify if the hatch of the postbox has been opened	

Communication Devices

CCU Duty Cycle Monitor

Purpose

To monitor the RaspberryMatic System Variable “Duty Cycle”.

Solutions

Two solutions worked out:

1. (Pull) Domoticz dzVents Lua script requests (pull, HTTP XML-API) in regular intervals the value of the Duty Cycle system variable and updates the Domoticz percentage device. The device log is updated every 5 min.
2. (Push) Node-RED Dashboard which in addition publishes (pushes) via MQTT the Duty Cycle value to a Domoticz device.
   The dashboard has a gauge and a chart with 1-day-view. Data is updated every 2 min (value can be changed. Be careful updating too frequent as this will increase the DutyCycle value which could interrupt other devices).

Bevor explaining the solutions, which uses HTTP XML-API requests, info on the requests.

HTTP XML-API Requests

Required is the system variable Duty Cycle. This is obtained from the list of system variables. The ise_id is 1390.

GET System Variables

```
http://ccu-ip/addons/xmlapi/sysvarlist.cgi
```

HTTP Response

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<systemVariables>
  <systemVariable value_name_1="${sysVarAlarmZone1Triggered}"
value_name_0="${sysVarAlarmZone1NotTriggered}" timestamp="0" visible="true" logged="false" subtype="6"
type="2" unit="" max="" min="" ise_id="1235" value_list="" value="" variable="4"
name="${sysVarAlarmZone1}"/>
  <systemVariable value_name_1="${sysVarPresencePresent}" value_name_0="${sysVarPresenceNotPresent}"
timestamp="1587921018" visible="true" logged="false" subtype="2" type="2" unit="" max="" min=""
ise_id="950" value_list="" value="true" variable="1" name="${sysVarPresence}"/>
  <systemVariable value_name_1="" value_name_0="" timestamp="1587970740" visible="true" logged="false"
subtype="0" type="4" unit="%" max="100" min="-1" ise_id="1390" value_list="" value="2.000000"
variable="2.000000" name="DutyCycle"/>
  <systemVariable value_name_1="${sysVarAlarmZone1Triggered}"
value_name_0="${sysVarAlarmZone1NotTriggered}" timestamp="0" visible="true" logged="false" subtype="6"
type="2" unit="" max="" min="" ise_id="1416" value_list="" value="" variable="4" name="WatchDog-
Alarm"/>
</systemVariables>
```

GET Single System Variable
Example DutyCycle with ise_id 1390 (see previous HTTP XML-API Response).

HTTP Request

```
http://ccu-ip/addons/xmlapi/sysvar.cgi?ise_id=1390
```

HTTP Reponse

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <systemVariables>
  <systemVariable value_name_1="" value_name_0="" timestamp="1587970920" subtype="0" type="4" unit="%"
max="100" min="-1" ise_id="1390" value_text="" value_list="" value="3.000000" variable="3.000000"
name="DutyCycle"/>
</systemVariables>
```

Domoticz Configuration

The RaspberryMatic system variable “Duty Cycle” is configured as a Percentage device in Domoticz.

1. Create a virtual sensor for hardware Virtual Devices:
   Name: RaspMatic Duty Cycle
   Sensor Type: Percentage
2. After adding, the Domoticz devices list has a new entry:
   Idx: 116, Hardware: VirtualDevices, Name: RaspMatic Duty Cycle, Type: General, SubType: Percentage, Data: 0%.
3. The device s shown in the GUI > Utility > Widget RaspMatic Duty Cycle

Solution Domoticz

A Domoticz dzVents automation scripts requests, via HTTP XML-API, in regular intervals (i.e. 3 min), the status of the RaspberryMatic system variable “DutyCycle”.
The HTTP XML response is parsed to get the key value and update the Domoticz Device “RaspberryMatic Duty Cycle” (idx: 116).

Enhancements (not implemented)

- Set an Alert message in case the DutyCycle value exceeds a threshold (could be set as Domoticz User Variable)

Device Widget and Day Log (extract)

Automation Event dzVents Lua

Source: raspmatic_dutycycle_monitor.dzvents

```
-- url http xml-api request system variable
local URL_RASPMATIC = 'http://ccu-ip//addons/xmlapi/sysvar.cgi?ise_id=';
-- define the unique (across all dzVents) callback
local RES_RASPMATIC = "RES_RASPMATIC_DUTYCYCLE_MONITOR";
-- set the datapoint ise_id of the "DutyCycle" with ise_id 1390
local DATAPOINT_ISE_ID = 1390;
-- domoticz dutycycle device
local IDX_RASPMATIC_DUTYCYCLE = 116;
-- set the alert message if dutycycle value is above threshold
local TH_DUTYCYCLE = 4;

return {
  on = {
    timer = { 'every 3 minutes' },
    httpResponses = { RES_RASPMATIC, }
  },
  execute = function(domoticz, item)
    -- check if the item is a device, then request information
    if (item.isTimer) then
      domoticz.openURL({url = URL_RASPMATIC .. DATAPOINT_ISE_ID, 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
        -- domoticz.log(item.data);
        -- Select the callback
        -- in this case there is only one (but for any next developments here might be more)
        if (item.callback == RES_RASPMATIC) then
          -- get the key value from the http response
          -- <systemVariable value_name_1="" value_name_0="" timestamp="1587970920" subtype="0" type="4"
unit="%" max="100" min="-1" ise_id="1390" value_text="" value_list="" value="3.000000"
variable="3.000000" name="DutyCycle"/>
          local value =
tonumber(domoticz_applyXPath(item.data,'//systemVariable[@ise_id="'..DATAPOINT_ISE_ID..'"]/@value'))
          domoticz.devices(IDX_RASPMATIC_DUTYCYCLE).updatePercentage(value);
          domoticz.log(string.format("Duty Cycle: %.0f", value));
          if value > TH_DUTYCYCLE then
            local message= string.format("DutyCycle above threshold:%.0f", value);
            domoticz.log(message)
          end
        end
      else
        domoticz.log('[ERROR] Request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

Solution Node-RED

The picture below shows the Node-RED flow, a simple dashboard with chart & gauge in a 1-day-view.

Source: raspmatic_dutycycle_dashboard.flow

Flow

- The flow sends in regular intervals (2 min) a HTTP XML-API request to RaspberryMatic to get the datapoint for the system variable Duty Cycle.
- The HTTP XML-API requests uses the ise_id of the Duty Cycle datapoint: http://ccu-ip/addons/xmlapi/sysvar.cgi?ise_id=1390.
- The ise_id is obtained by requesting the list of system variables: http://ccu-ip/addons/xmlapi/sysvarlist.cgi .
- The HTTP XML-API response is converted to a JSON object from which a msg is created used to update the ui_gauge, ui_chart and Domoticz device (via MQTT to the Domoticz broker). The chart holds a 1-day-view.

The core of the flow are the functions

- Set Data – create the message for the ui_gauge & ui_chart
- Set MQTT Domoticz – create the message for the Domoticz broker

Function Set Data

```
/*
Get in regular intervals the Raspberry matic System Variable Duty Cycle.
{"systemVariables":
  {"systemVariable":[
  {"$":{"name":"DutyCycle","variable":"2.000000","value":"2.000000","value_list":"","value_text":"","is
e_id":"1390","min":"-
1","max":"100","unit":"%","type":"4","subtype":"0","timestamp":"1587971040","value_name_0":"","value_na
me_1":""}}
    ]
  }
}
The msg.payload is an object i.e.
{"ise_id":"1390","name":"DutyCycle","value":"1.000000","min":0,"max":2,"seg1":2,"seg2":5,"timestamp":"1
0:06"}
The gauge scaling is set dynamic depending dutycycle value.
*/
var maxValue = 5;

// Helpers
var getTimeStamp = function(timestamp,usesecs){
    dateObj = new Date(timestamp * 1000);
    hours = dateObj.getHours();
    minutes = dateObj.getMinutes();
    seconds = dateObj.getSeconds();
    var result = hours.toString().padStart(2, '0') + ':' + minutes.toString().padStart(2, '0');
    result = (usesecs) ? result + ':' + seconds.toString().padStart(2, '0') : result;
    return result
```

```
}

// Get the systemvariable from systemvariabless JSON array. There is (should) a single entry
var systemVariable = msg.payload.systemVariables.systemVariable[0].$;

// Get selected key:value pairs
// Create JSON object to return as payload and used by gauge, charts etc.
// Example: {"ise_id":"1390","name":"DutyCycle","value":"1.000000","min":0,"max":2,"seg1":2,"seg2":5}
data = {}
data.ise_id = systemVariable.ise_id;
data.name = systemVariable.name;
data.value = Math.round(systemVariable.value,0);             // should be low range 1-2%
maxValue = (data.value * 2 > maxValue) ? data.value * 2 : maxValue;
data.min = 0;
data.max = maxValue;
data.ymax = data.max;
data.seg1 = 2;
data.seg2 = 5;
data.timestamp = getTimeStamp(systemVariable.timestamp, false);

msg = {payload:data, ui_control:{min:data.min, max:data.max, seg1:data.seg1, seg2:data.seg2,
ymax:data.max}, topic:data.name};

node.status({fill:"green",shape:"dot",text:msg.payload.value + ", " + msg.payload.timestamp});
return msg;
```

Function Set MQTT Domoticz

```
/*
Create the domoticz mqtt message to publish.
Note the key svalue must be a string with converted from float ith base 10!
Domoticz log entry receiving mqtt message:
2020-04-28 09:29:48.380 MQTT: Topic: domoticz/in, Message:
{"command":"udevice","idx":116,"nvalue":0,"svalue":"2"}
*/
var value = msg.payload.value;
var msgDomoticz = {};
msgDomoticz.topic = "domoticz/in";
msgDomoticz.payload = {
    "command":"udevice",
    "idx":116,
    "nvalue":0,
    "svalue": value.toString(10)
}
node.status({fill:"green",shape:"dot",text:"Duty Cycle: " + value});
return msgDomoticz;
```

Experiments

Next are just a few experiments briefly described, to test connecting and using devices.

HMIP-PSM

To switch the power on/off for all devices and measure the Power (W) and Energy Consumption (Wh) of the room MakeLab.

A homematicIP Pluggable Switch and Meter (HmIP-PSM, Device ID: 1418) connected to a RaspberryMatic operating system running a HomeMatic Central-Control-Unit (CCU).

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

RaspberryMatic

RaspberryMatic Device Configuration	**Domoticz Device Configuration (Name/Type/SubType)**
	Device Powerswitch MakeLab Light/switch Switch **Device** Electric Usage MakeLab General kWh

Domoticz

Automation Scripts

Event name: powerswitch_makelab

```
local IDX_POWERSWITCH_MAKELAB = 177;
local ID_DATAPOINT_STATE = 1451
local URL_CCU = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' .. ID_DATAPOINT_STATE ..
'&new_value='
local RES_CCU = 'powerswitchmakelab';
return {
  on = {
    devices = {IDX_POWERSWITCH_MAKELAB},
    httpResponses = {RES_CCU}
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
      local state = 'true';
      if domoticz.devices(IDX_POWERSWITCH_MAKELAB).state == 'Off' then
        state = 'false';
      end
      domoticz.openURL({ url = URL_CCU .. state, method = 'GET',   callback = RES_CCU });
    end;
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        domoticz.log('[INFO] Switch : ' .. item.statusText)
      else
        domoticz.log('[ERROR] 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/addons/xmlapi/state.cgi?device_id=' .. ID_DEVICE;
local RES_CCU = 'electricusagemakelab';
local DECIMALS = 2;

return {
  on = {
    timer = {'every minute'},
    httpResponses = {RES_CCU}
  },
  execute = function(domoticz, item)
    if (item.isTimer) then
      domoticz.openURL({ url = URL_CCU, method = 'GET', callback = RES_CCU })
    end
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
        local powervalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1471"]/@value')   -- W
        local energyvalue = domoticz_applyXPath(item.data,'//datapoint[@ise_id="1467"]/@value')  -- Wh
        domoticz.devices(IDX_ELECTRICUSAGEMAKELAB).updateElectricity(
            domoticz.helpers.roundnumber(tonumber(powervalue),2),     -- W
            domoticz.helpers.roundnumber(tonumber(energyvalue),2) )   -- Wh

      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
  end
}
```

HMIP-eTRV-2

To measure and control the MakeLab room temperature.

A homematicIP Radiator Thermostat (HmIP-eTRV-2, Device ID: 1541) connected to a RaspberryMatic operating system running a HomeMatic Central-Control-Unit (CCU).
The thermostat setpoint is set by a Domoticz Selector Switch.

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

RaspberryMatic

Device Configuration

Domoticz

Domoticz Device Configuration

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
176	VirtualSensors	00014100	1	Thermostat MakeLab Setpoint	Light/Switch	Switch	Off
175	VirtualSensors	00082175	1	Thermostat MakeLab Info	General	Text	Setpoint: 4.5, Temperature: 23.2

Change Thermostat Setpoint

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

Domoticz Automation Script

Event name: thermostat_makelab_setpoint

```
local IDX_THERMOSTAT_MAKELAB_SETPOINT = 176;
local ID_DATAPOINT_SETPOINT = 1584;
local URL_RASPMATIC = 'http://ccu-ip/addons/xmlapi/statechange.cgi?ise_id=' ..  ID_DATAPOINT_SETPOINT
.. '&new_value='
local RES_RASPMATIC = 'thermostatmakelabsetpoint';
local DECIMALS = 2;

return {
  on = {
    devices = {IDX_THERMOSTAT_MAKELAB_SETPOINT},
    httpResponses = {RES_RASPMATIC}
  },
  execute = function(domoticz, item)
    if (item.isDevice) then
        -- get the new SETPOINT 0%=0(OFF); 10%-40%=18-21; min=18, max=21
        local level = domoticz.devices(IDX_THERMOSTAT_MAKELAB_SETPOINT).level;
        local newsetpoint = 0;
        if level > 0 then
            newsetpoint = 17 + (level * 0.1);
        end
        -- set the new setpoint
        domoticz.log('New setpoint:' .. URL_RASPMATIC .. tostring(newsetpoint), domoticz.LOG_INFO);
       domoticz.openURL({url = URL_RASPMATIC ..tostring(newsetpoint), method = 'POST', callback =
RES_RASPMATIC});
    end

    -- check if the item is a httpresponse from the openurl callback
    if (item.isHTTPResponse) then
      if (item.statusCode == 200) then
          -- multile datapoints
          domoticz.log(item.data);
          -- parse the response using XPath
          -- select the attribute value of the datapoint element (this is XPath syntax)
          -- domoticz.log(domoticz_applyXPath(item.data,'//datapoint[@ise_id="1584"]/@value')
      else
        domoticz.log('[ERROR] Problem handling the request:' .. item.statusText, domoticz.LOG_ERROR)
      end
    end
 end
}
```

Node-RED Dasboard Thermostat

State upfront: Domoticz is not involved in this experiment. There is direct communication between RaspberryMatic and Node-RED.

To create a Node-RED dashboard showing thermostat datapoints

- ACTUAL_TEMPERATURE – measured at the thermostat (°C)
- SET_POINT_TEMPERATURE – at the thermostat (°C)
- LEVEL - the alve position (%)

The Node-RED dashboard add-on, enables to create live dashboards (see Node-RED).

This experiment is developed on the Domoticz development system running Node-RED.

As already mentioned: a Domoticz system is not required.
Node-RED communicates direct with the RaspberryMatic system to obtain
the value of the datapoints by triggering HTTP XML-API requests (using the state.cgi script).

As a first test, several datapoints of the thermostat MakeLab are selected.

Node-RED

The picture below shows the Node-RED flow, a simple dashboard with chart & gauge and the debug log.

Note
In a browser, enter URL http://domoticz-ip:1880/ui to view the Node-RED dashboard tab Homematic > Group Thermostat MakeLab.

Flow

The **Inject node** triggers every 30s, the action for the **HTTP request node** to submit the URL

```
http://ccu-ip/addons/xmlapi/state.cgi?device_id=1541
```

to get the state information for the device with datapoint id 1541.

The HTTP request returns XML structured string with all device datapoints.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<state>
  - <device config_pending="false" unreach="false" ise_id="1541" name="Thermostat MakeLab">
    - <channel ise_id="1542" name="Thermostat MakeLab:0">
        <datapoint ise_id="1543" name="HmIP-RF.000A18A9A64DAC:0.CONFIG_PENDING" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
        <datapoint ise_id="1547" name="HmIP-RF.000A18A9A64DAC:0.DUTY_CYCLE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
        <datapoint ise_id="1549" name="HmIP-RF.000A18A9A64DAC:0.LOW_BAT" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="LOW_BAT"/>
        <datapoint ise_id="1553" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE" timestamp="1576843927" valueunit="" valuetype="4" value="2.800000" type="OPERATING_VOLTAGE"/>
        <datapoint ise_id="1554" name="HmIP-RF.000A18A9A64DAC:0.OPERATING_VOLTAGE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
        <datapoint ise_id="1555" name="HmIP-RF.000A18A9A64DAC:0.RSSI_DEVICE" timestamp="1576843927" valueunit="" valuetype="8" value="189" type="RSSI_DEVICE"/>
        <datapoint ise_id="1556" name="HmIP-RF.000A18A9A64DAC:0.RSSI_PEER" timestamp="1576778404" valueunit="" valuetype="8" value="178" type="RSSI_PEER"/>
        <datapoint ise_id="1557" name="HmIP-RF.000A18A9A64DAC:0.UNREACH" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="UNREACH"/>
        <datapoint ise_id="1561" name="HmIP-RF.000A18A9A64DAC:0.UPDATE_PENDING" timestamp="1575413140" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
      </channel>
    - <channel ise_id="1565" name="HmIP-eTRV-2 000A18A9A64DAC:1">
        <datapoint ise_id="1566" name="HmIP-RF.000A18A9A64DAC:1.ACTIVE_PROFILE" timestamp="1576843927" valueunit="" valuetype="16" value="1" type="ACTIVE_PROFILE"/>
        <datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927" valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
        <datapoint ise_id="1568" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="ACTUAL_TEMPERATURE_STATUS"/>
        <datapoint ise_id="1569" name="HmIP-RF.000A18A9A64DAC:1.BOOST_MODE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="BOOST_MODE"/>
        <datapoint ise_id="1570" name="HmIP-RF.000A18A9A64DAC:1.BOOST_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="BOOST_TIME"/>
        <datapoint ise_id="1571" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_DIFFERENTIAL_TEMPERATURE" timestamp="0" valueunit="" valuetype="4" value="" type="CONTROL_DIFFERENTIAL_TEMPERATURE"/>
        <datapoint ise_id="1572" name="HmIP-RF.000A18A9A64DAC:1.CONTROL_MODE" timestamp="0" valueunit="" valuetype="16" value="" type="CONTROL_MODE"/>
        <datapoint ise_id="1573" name="HmIP-RF.000A18A9A64DAC:1.DURATION_UNIT" timestamp="0" valueunit="" valuetype="16" value="" type="DURATION_UNIT"/>
        <datapoint ise_id="1574" name="HmIP-RF.000A18A9A64DAC:1.DURATION_VALUE" timestamp="0" valueunit="" valuetype="16" value="" type="DURATION_VALUE"/>
        <datapoint ise_id="1575" name="HmIP-RF.000A18A9A64DAC:1.FROST_PROTECTION" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="FROST_PROTECTION"/>
        <datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" timestamp="1576843927" valueunit="" valuetype="4" value="0.270000" type="LEVEL"/>
        <datapoint ise_id="1577" name="HmIP-RF.000A18A9A64DAC:1.LEVEL_STATUS" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="LEVEL_STATUS"/>
        <datapoint ise_id="1578" name="HmIP-RF.000A18A9A64DAC:1.PARTY_MODE" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="PARTY_MODE"/>
        <datapoint ise_id="1579" name="HmIP-RF.000A18A9A64DAC:1.PARTY_SET_POINT_TEMPERATURE" timestamp="0" valueunit="" valuetype="4" value="0.000000" type="PARTY_SET_POINT_TEMPERATURE"/>
        <datapoint ise_id="1580" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_END" timestamp="0" valueunit="" valuetype="20" value="" type="PARTY_TIME_END"/>
        <datapoint ise_id="1581" name="HmIP-RF.000A18A9A64DAC:1.PARTY_TIME_START" timestamp="0" valueunit="" valuetype="20" value="" type="PARTY_TIME_START"/>
        <datapoint ise_id="1582" name="HmIP-RF.000A18A9A64DAC:1.QUICK_VETO_TIME" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="QUICK_VETO_TIME"/>
        <datapoint ise_id="1583" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_MODE" timestamp="1576843927" valueunit="" valuetype="16" value="1" type="SET_POINT_MODE"/>
        <datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" timestamp="1576843927" valueunit="°C" valuetype="4" value="20.000000" type="SET_POINT_TEMPERATURE"/>
        <datapoint ise_id="1585" name="HmIP-RF.000A18A9A64DAC:1.SWITCH_POINT_OCCURED" timestamp="1576843927" valueunit="" valuetype="2" value="false" type="SWITCH_POINT_OCCURED"/>
        <datapoint ise_id="1586" name="HmIP-RF.000A18A9A64DAC:1.VALVE_ADAPTION" timestamp="0" valueunit="" valuetype="2" value="false" type="VALVE_ADAPTION"/>
        <datapoint ise_id="1587" name="HmIP-RF.000A18A9A64DAC:1.VALVE_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="4" type="VALVE_STATE"/>
        <datapoint ise_id="1588" name="HmIP-RF.000A18A9A64DAC:1.WINDOW_STATE" timestamp="1576843927" valueunit="" valuetype="16" value="0" type="WINDOW_STATE"/>
      </channel>
      <channel ise_id="1589" name="HmIP-eTRV-2 000A18A9A64DAC:2"/>
      <channel ise_id="1590" name="HmIP-eTRV-2 000A18A9A64DAC:3"/>
      <channel ise_id="1591" name="HmIP-eTRV-2 000A18A9A64DAC:4"/>
      <channel ise_id="1592" name="HmIP-eTRV-2 000A18A9A64DAC:5"/>
      <channel ise_id="1593" name="HmIP-eTRV-2 000A18A9A64DAC:6"/>
      <channel ise_id="1594" name="HmIP-eTRV-2 000A18A9A64DAC:7"/>
    </device>
</state>
```


Note
All datapoints with the example datapoint ACTUAL_TEMPERATURE highlighted as used in the example next

The **XML node** converts the XML structure to a Javascript object which is used by the various **Change nodes** to get the value of a specific datapoint by using JSONata expressions. JSONata makes it simple to parsing this kind of data.

Example JSONata expressions

Get the ACTUAL_TEMPERATURE value for datapoint 1567.
The payload returned is a string.

```
payload.**[ise_id="1567"].value
```

Notes
The multi-level wildcard ** traverses all descendants. In this case, select all “ise_id equals 1567” values, regardless of how deeply nested the information is in the XML structure.

The XML structure snippet parsed:

```
…
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" timestamp="1576843927"
valueunit="" valuetype="4" value="20.900000" type="ACTUAL_TEMPERATURE"/>
…
```

JSONata enables also to change the value, like for example the LEVEL value (=the valve position) from 0-1 to 0-100 (displayed with unit %).
The payload returned is a number.

```
$number(payload.**[ise_id=\"1576\"].value) * 100
```

The value is used by Node-Red Dashboard nodes like the Gauge (ui_gauge) or Chart (ui_chart).

The next flow has the two thermostats MakeLab and LivingRoom based on previous flow explained.

RaspMatic XML-API Datapoints
Inject HTTP Request Statelist DataPoints
Send HTTP Request
Convert XML to JavaScript Object
DEBUG JSONNATA
ML-D LEVEL (1576)
ML-D ACTUAL_TEMPERATURE (1567)
ML-D SET_POINT_TEMPERATURE (1584)
WZ-D LEVEL (3578)
ML-D ACTUAL_TEMPERATURE (3569)
ML-D SET_POINT_TEMPERATURE (3586)
ML-G Valve Opening
ML-G Temperature
ML-G Setpoint
WZ-G Valve Opening
WZ-G Temperature
WZ-G Setpoint
ML-C Valve opening
ML-C Temperature
ML-C Setpoint
WZ-C Valve opening
WZ-C Temperature
WZ-C Setpoint

Thermostat MakeLab
Setpoint
20
Temperature
19.8
Valve Position
47
Thermostat WZ
Setpoint
21
Temperature
20.2
Valve Position
100
07:00
15:00

RaspberryMatic triggers Node-RED flow (Push)

State upfront: Domoticz is not involved in this experiment, i.e. direct communication between RaspberryMatic and Node-RED.

To handle the state change of a window and door contact with magnet device (HmIP-SWDM).
Every state change is handled by a RaspberryMatic script which triggers a Node-RED HTTP GET request (RaspberryMatic pushes the state information the Node-RED flow).
The HTTP GET request parameter are used by Node-RED to take action accordingly.

Raspberry Matic

The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE
If the SWDM device state changes, an HTTP request is sent (pushed) to Node-RED.
Example:

```
http://domoticz-ip:1880/3597?dp=3622&state=1
```

The parameters are the device id SWDM, datapoint id STATE and current state (0=closed, 1=open).
The request is handled by Node-RED.

RaspberryMatic Script
The script is triggered if the state of the device changes either open or close.

Script Source Code

```
string sFunction = "Eingang Monitor";
string sDate = system.Date("%d.%m"); ! sDate = "09.08";
string sTime = system.Date("%H:%M"); ! sTime = "07:32";
! The object string is taken from the raspmatic XML-API script statelist.cgi
string sState = dom.GetObject("HmIP-RF.001558A99D5A78:1.STATE").State();
! Node-RED Test
string cQMark = "?";
string cAmp = "&";
! XMLAPI datapoints Device & STATE
string sIDDevice = "3597";
string sIDState = "3622";
! Datapoint STATE property
string sUrlNR = "'http://domoticz-ip:1880/"; ! Development
string sUrl = sUrlNR#sIDDevice#cQMark#"dp="#sIDState#cAmp#"state="#sState#"'";
WriteLine(sUrl);
```

```
! Run the command with a return result
! Define the command to execute
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State ("wget -q -O - "#sUrl);

! Set the return flag to 1 to be able to read the json result
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State (1);

! Start the command
! NOTE: script running on CCU waits until completion - ensure not to execute commands which take long
string sRes = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! handle result
var dl = dom.GetObject ("DomoticzLog");
! Update the var with result text
if (sRes.Find("OK") > 0) {
  dl.Variable(sFunction#"-Last update OK:"#sDate#" "#sTime )
}
else {
  dl.Variable(sFunction#"Last update ERROR:"#sDate#" "#sTime)
};
! WriteLine("VT="#dl.VarType());  ! VT=9
! WriteLine(dl.Variable());       ! shows the last update string
```

Node-RED

Flow

Flow Description

Node	Comments
http in	Method: GET URL: /3597 Output to nodes: http response, switch, various debug *Notes* The output of the http in node is a JSON string containing the parameter. Example: {"dp":"3622","state":"1"} from the URL http://domoticz-ip:1880/3597?dp=3622&state=1 with device id 3597 and datapoint 3622.
http response	No changes made, kept the defaults
switch	Property: msg.req.query.state == a/z: 0 == a/z: 1

	Output to nodes: switch, debug There are two outputs covering 0 and 1 Notes: The property "state" from the URL query is used with value 0 | 1 (req.query.state). The property "dp" has value "3622 (req.query.dp).
change	There are two change nodes handling 0 and 1. For 0, the value is to the string Closed. For 1, the value is to the string Open.
debug	Outputs the value of various nodes, i.e. http in, change

Node-RED Debug log from testing the RaspberryMatic script
The script sends HTTP request: http://domoticz-ip:1880/3597?dp=3622&state=1

Node-RED triggers HTTP GET Request (Pull)

State upfront: Domoticz is not involved in this experiment, i.e. direct communication between RaspberryMatic and Node-RED.

To request the state change of a window and door contact with magnet device (HmIP-SWDM).
The SWDM datapoints id used are: 3597 = Device ID, 3622 = STATE

Node-RED

Node-RED to pull from RaspberryMatic the state of the SWDM device.
This is an example how to trigger, by Node-RED, the value of a datapoint via the XML-API.

Flow

Flow Description

Node	Comments
Inject	Start the request. An interval can be set to regular request device information. For this example does not make sense as the state can be changed within an interval.
http request	Send the HTTP request using the defined URL. Example: http://ccu-ip/config/xmlapi/state.cgi?device_id=3597
Json	Convert the HTTP XML response to a JavaScript object.
change	Get the value from the XML tree using JSONata & xpath:
	`$join(["STATE:", msg.payload.**[ise_id="3622"].value])`
Debug	Output the value of the state, i.e. 0 |1

Node-RED triggers HTTP GET Request All Datapoints

This concept is based on the tool RaspberryMatic Statelist and is enhanced further to:

- get, in regular intervals, using HTTP XML-API request (i.e. http://ccu-ip/config/xmlapi/statelist.cgi),
- a device datapoint value and
- update the Domoticz device via an HTTP API request (i.e. http://domoticz-ip:8080/json.htm?type=command&param=udevice&idx=113&nvalue=0&svalue=24.1).

Instead of setting a single device, there is also an example how to set multiple devices from multiple datapoints.

Example Single Device

Get the RaspberryMatic datapoint 1567 (Thermostat MakeLab, Type ACTUAL_TEMPERATURE, Value °C) and assign to the Domoticz device with idx=113.

	idx=113 Hardware=VirtualDevices, Name=MakeLab Temperature Type=Temp SubType=LaCrosse TX3

Node-RED Flow
Source: raspmatic_single_device.flow

Function Node Get Devices and Datapoints
The purpose of this function is to

- return a json array with devices used for the ui_dropdown
- set flow context "datapoints" for the ui_template to display the datapoints for the selected device id

Notes

- Example device entry: deviceobj["Briefkasten Status"]=2530;

- Example datapoint entry: device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-RF.0000DA498D5859:0.CONFIG_PENDING"}
- the second array below is not used as replaced by the flow context "datapoints" - but is kept in case any enhancements planned

```
statelist = msg.payload.stateList;
devices = statelist.device
arrdevices = [];
arrdatapoints = [];
devices.forEach(function(device) {
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename]=deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            channeldatapoints.forEach(function(datapoint){
                obj = datapoint.$;
                if (obj !== undefined) {
                  dpobj = {
                    "device":deviceise_id,
                    "datapoint": {
                      "id":obj.ise_id,"type":obj.type,"value":obj.value,"name":obj.name
                    }
                  };
                arrdatapoints.push(dpobj);
                }
            });
        }
    });
});
flow.set("datapoints",arrdatapoints);

msgdevices = {};
msgdevices.options = arrdevices;

// not used
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices,msgdatapoints];
```

Function Node Get Datapoint Value

Get the value from the flow context “datapoints” using the datapoint id and returns the value property.

```
// get the datapoint id, i.e. 1657 for the selected channel
var datapointid = msg.payload;
// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");
datapoints.forEach(function(datapoint){
    if (datapoint.datapoint.id == datapointid) {
        msg.payload = datapoint.datapoint.value;
    }
});
return msg;
```

Function Node Set HTTP API request

Set the msg.url to request Domoticz to update the device with idx=113.
The flow context “domip” holds the Domoticz system IP address.

```
const temp = msg.payload;
const idx = flow.get("device").idx;

// Define the url to set the temperature of the Domoticz device
// /json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP

msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx=" + idx +
"&nvalue=0&svalue=" + temp;
node.warn(msg.url);

return msg;
```

Example Multiple Devices

To get & set the value for multiple Domoticz devices a JSON array is defined in the Node-RED flow change node “Set Devices”. This array is used by the function node “Get Devices and Set Value”.

For this example, get three RaspMatic datapoints and set the value of three Domoticz devices.

Extract from the RaspMatic full statelist obtained using HTTP XML-API request:

```
<datapoint ise_id="1567" name="HmIP-RF.000A18A9A64DAC:1.ACTUAL_TEMPERATURE" operations="5"
timestamp="1583752872" valueunit="" valuetype="4" value="22.800000" type="ACTUAL_TEMPERATURE"/>

<datapoint ise_id="1576" name="HmIP-RF.000A18A9A64DAC:1.LEVEL" operations="7" timestamp="1583752872"
valueunit="" valuetype="4" value="0.310000" type="LEVEL"/>

<datapoint ise_id="1584" name="HmIP-RF.000A18A9A64DAC:1.SET_POINT_TEMPERATURE" operations="7"
timestamp="1583752872" valueunit="Â°C" valuetype="4" value="21.000000" type="SET_POINT_TEMPERATURE"/>
```

The Node-RED change node to configure the datapoints in JSON format (array with several key:value pairs) and the three Domoticz devices (from the GUI > Setup > Devices list).

Edit change node > **JSON editor**

Edit JSON | Visual editor

```
[
    {
        "datapoint": 1567,
        "type": "ACTUAL_TEMPERATURE",
        "idx": 113
    },
    {
        "datapoint": 1576,
        "type": "LEVEL",
        "idx": 114
    },
    {
        "datapoint": 1584,
        "type": "SET_POINT_TEMPERATURE",
        "idx": 115
    }
]
```

Idx	Hardware	ID	Unit	Name	Type	SubType	
115	VirtualDevices	140C3	1	MakeLab Thermostat Setpoint	Temp	LaCrosse TX3	6.0 C
114	VirtualDevices	00082114	1	MakeLab Thermostat Valve	General	Percentage	0%
113	VirtualDevices	140C1	1	MakeLab Thermostat Temperature	Temp	LaCrosse TX3	21.8 C

RaspMatic Datapoint = Domoticz Idx - see left.

Node-RED Flow

Source: raspmatic_multiple_devices.flow

Function Node Get Devices and Datapoints

Same as for the single device.

Function Node Get Devices and Set Value

```
/*
The purpose of this function is to set the value for multiple domoticz devices.
This function does not return a value as msg are send during device json array loop.
*/
const DEBUG = true;
// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");
// devices: [{"datapoint": 1567,"type": "ACTUAL_TEMPERATURE","idx": 113}, ...]
devices = flow.get("devices");
//
devices.forEach(function(device){
    datapoints.forEach(function(datapoint){
        if (device.datapoint == datapoint.datapoint.id) {
            idx = device.idx;
            value = datapoint.datapoint.value;
            if (idx == "114") value = value * 100;
            msg.url = "http://" + flow.get("domip") + ":8080/json.htm?type=command&param=udevice&idx="
+ idx + "&nvalue=0&svalue=" + value;
            node.send(msg);
            if (DEBUG) node.warn("URL: " + msg.url);
        }
    });
});
```

SQL

Purpose

To execute CRUD operations on the Domoticz SQLite Database domoticz.db.
CRUD = Create (Create, Insert), Read (Select), Update and Delete.
The operations are performed via SQL statements:

- A statement is any text that the database engine recognizes as a valid command.
- A query is a statement that returns a recordset or nothing if empty.

Thoughts on using SQL

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

Approach

- **Explore Database Structure & SQL Statements**
  To analyse the database, external tools like the SQLiteSpy are useful helpers, but also using SQLite3 from the terminal command-line (Command Line Interface CLI).
- **For Tests**
  The Domoticz database is exported from the Domoticz production system to the Development Device:
  GUI > Setup > Settings > Backup/Restore > Backup Database.
  The CRUD operations are therefor executed in offline mode.
- **For Production**
  Ensure Domoticz is stopped first, then perform CRUD operations on the Domoticz database and start Domoticz after completing.
  This is only required if changes (Update or Delete) to the database are made. For Read (Select), no need to stop Domoticz.

SQLite Shell

The sqlite3 package (include SQLite libraries) enables to perform SQL operations from the command-line shell - SQLite shell or SQLite CLI (Command-Line-Interface) Shell.

Install SQLite3 Package

Login as user “pi” and install the “sqlite3” package.

```
sudo apt install sqlite3
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  sqlite3-doc
The following NEW packages will be installed:
  sqlite3
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 709 kB of archives.
After this operation, 1,991 kB of additional disk space will be used.
Get:1 http://mirror.de.leaseweb.net/raspbian/raspbian stretch/main armhf sqlite3 armhf 3.16.2-5+deb9u1
[709 kB]
Fetched 709 kB in 2s (325 kB/s)
Selecting previously unselected package sqlite3.
(Reading database ... 96101 files and directories currently installed.)
Preparing to unpack .../sqlite3_3.16.2-5+deb9u1_armhf.deb ...
Unpacking sqlite3 (3.16.2-5+deb9u1) ...
Setting up sqlite3 (3.16.2-5+deb9u1) ...
Processing triggers for man-db (2.7.6.1-2) ...
```

Prepare SQL Statements

For the examples, the Domoticz production database “domoticz.db”, located on the Raspberry Pi in folder “/home/pi/domoticz”, is used.

Login as user “pi”, change directory to “domoticz”, start sqlite3 and connect to the database domoticz.db.

```
Login username "pi"
$cd domoticz
$sqlite3
SQLite version 3.16.2 2017-01-06 16:32:41
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open domoticz.db
```

Notes
Usage of the semicolon “;” in SQL statements.

- On dot “.” commands, to control the SQLite shell, do NOT use the semicolon to close the command off. This gives an error or non expected results.
  Example: “.open domoticz.db;” creates a new database with filename “domoticz.db;”.
- Use the semicolon on SQL statements, i.e. SELECT, UPDATE etc.

SQL Statement Examples

Some examples SQL statement. See also in several function, like function Notes section Database Records.

Get All Tables

```
sqlite> .tables

BackupLog             Meter                 SceneTimers
Cameras               Meter_Calendar        Scenes
CamerasActiveDevices  MobileDevices         SetpointTimers
CustomImages          MultiMeter            SharedDevices
DeviceStatus          MultiMeter_Calendar   Temperature
DeviceToPlansMap      MySensors             Temperature_Calendar
EnoceanSensors        MySensorsChilds       TimerPlans
EventMaster           MySensorsVars         Timers
EventRules            Notifications         ToonDevices
Fan                   Percentage            UV
Fan_Calendar          Percentage_Calendar   UV_Calendar
FibaroLink            Plans                 UserSessions
Floorplans            Preferences           UserVariables
GooglePubSubLink      PushLink              Users
Hardware              Rain                  WOLNodes
HttpLink              Rain_Calendar         Wind
LightSubDevices       SceneDevices          Wind_Calendar
LightingLog           SceneLog              ZWaveNodes
```

Note
The various tables holding measurement data are

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

Get Structure DeviceStatus

There are several ways to get the structure of a table.

Command: .schema

```
sqlite> .schema devicestatus

CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT NULL, [DeviceID]
VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(100) DEFAULT Unknown, [Used] INTEGER
DEFAULT 0, [Type] INTEGER NOT NULL, [SubType] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0,
[Favorite] INTEGER DEFAULT 0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0,
[nValue] INTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFAULT
(datetime('now','localtime')),[Order] INTEGER BIGINT(10) default 0, [AddjValue] FLOAT DEFAULT 0,
[AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0, [AddjMulti2] FLOAT DEFAULT 1, [StrParam1]
VARCHAR(200) DEFAULT '', [StrParam2] VARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0,
[Protected] INTEGER DEFAULT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '',
[Options] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL);
CREATE INDEX ds_hduts_idx    on DeviceStatus(HardwareID, DeviceID, Unit, Type, SubType);
CREATE TRIGGER devicestatusupdate AFTER INSERT ON DeviceStatus
BEGIN
  UPDATE DeviceStatus SET [Order] = (SELECT MAX([Order]) FROM DeviceStatus)+1 WHERE DeviceStatus.ID =
NEW.ID;
END;
CREATE INDEX ds_hduts_idx    on DeviceStatus(HardwareID, DeviceID, Unit, Type, SubType);
CREATE TRIGGER devicestatusupdate AFTER INSERT ON DeviceStatus
BEGIN
  UPDATE DeviceStatus SET [Order] = (SELECT MAX([Order]) FROM DeviceStatus)+1 WHERE DeviceStatus.ID =
NEW.ID;
END;
```

Command: Pragma

```
sqlite> .header on
sqlite> .mode column
sqlite> pragma table_info('devicestatus');

cid         name        type        notnull     dflt_value  pk
----------  ----------  ----------  ----------  ----------  ----------
0           ID          INTEGER     0                       1
1           HardwareID  INTEGER     1                       0
2           DeviceID    VARCHAR(25  1                       0
3           Unit        INTEGER     0           0           0
4           Name        VARCHAR(10  0           Unknown     0
5           Used        INTEGER     0           0           0
6           Type        INTEGER     1                       0
7           SubType     INTEGER     1                       0
8           SwitchType  INTEGER     0           0           0
9           Favorite    INTEGER     0           0           0
10          SignalLeve  INTEGER     0           0           0
11          BatteryLev  INTEGER     0           0           0
12          nValue      INTEGER     0           0           0
13          sValue      VARCHAR(20  0           null        0
14          LastUpdate  DATETIME    0           datetime('  0
15          Order       INTEGER BI  0           0           0
16          AddjValue   FLOAT       0           0           0
17          AddjMulti   FLOAT       0           1           0
18          AddjValue2  FLOAT       0           0           0
19          AddjMulti2  FLOAT       0           1           0
20          StrParam1   VARCHAR(20  0           ''          0
21          StrParam2   VARCHAR(20  0           ''          0
22          LastLevel   INTEGER     0           0           0
23          Protected   INTEGER     0           0           0
24          CustomImag  INTEGER     0           0           0
25          Descriptio  VARCHAR(20  0           ''          0
26          Options     TEXT        0           null        0
27          Color       TEXT        0           NULL        0
```

Command: Select Query

```
sqlite> SELECT sql FROM sqlite_master WHERE name = "DeviceStatus";

sql
--------------------------------------------------------------------------------
CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT
NULL, [DeviceID] VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(
100) DEFAULT Unknown, [Used] INTEGER DEFAULT 0, [Type] INTEGER NOT NULL, [SubTyp
e] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0, [Favorite] INTEGER DEFAULT
0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0, [nValue] I
NTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFA
ULT (datetime('now','localtime')),[Order] INTEGER BIGINT(10) default 0, [AddjVal
ue] FLOAT DEFAULT 0, [AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0,
[AddjMulti2] FLOAT DEFAULT 1, [StrParam1] VARCHAR(200) DEFAULT '', [StrParam2] V
ARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0, [Protected] INTEGER DEFAU
LT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '', [O
ptions] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL)
```

Other Examples

```
sqlite> .schema Temperature

CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill]
FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT
0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));
CREATE INDEX t_id_idx        on Temperature(DeviceRowID);
CREATE INDEX t_id_date_idx   on Temperature(DeviceRowID, Date);
```

```
sqlite> .schema LightingLog

CREATE TABLE [LightingLog] ([DeviceRowID] BIGINT(10) NOT NULL, [nValue] INTEGER DEFAULT 0, [sValue]
VARCHAR(200), [Date] DATETIME DEFAULT (datetime('now','localtime')), [User] VARCHAR(100) DEFAULT (''));
```

```
CREATE INDEX ll_id_idx       on LightingLog(DeviceRowID);
CREATE INDEX ll_id_date_idx  on LightingLog(DeviceRowID, Date);
```

Select DeviceStatus ID,Name

```
sqlite> select ID, Name from DeviceStatus;

1|RPi RAM Usage
2|RPi HDD /boot
… and many more
12|Blind Living Room
207|Hzg Bad Sollwert
208|Hzg Bad Temperatur
209|Hzg Bad Batterie
…
```

Select DeviceStatus ID,Name,nValue,sValue,LastUpdate

For the device with id=201. The ID is the Idx of the device.

```
sqlite> sqlite> select ID,Name,nValue,sValue,LastUpdate from DeviceStatus where ID=201;

201|Hzg WZ Sollwert|1|21.0|2019-12-17 06:15:02
```

Select LightingLog Device idx=161

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

161|0|Security Main Door changed to Alarm + Tamper|2019-05-21 09:01:07|
161|0|Security Main Door changed to Normal|2019-05-21 09:05:00|
161|0|Hue MakeLab changed to Off|2019-05-21 16:16:11|
… and many more …
sqlite>
```

Select Temperature Device idx=120

```
sqlite> select * from Temperature where devicerowid=120;

120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:40:00
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:45:00
… and many more …
```

Select Temperature,Name Device idx=120

Show the device Name and limit the number of entries to 5.

```
sqlite> select * from Temperature, (select Name from devicestatus where id=120) as Name where
devicerowid=120 limit 5;

120|16.3|0.0|91|0|14.83|0.0|2019-05-21 09:45:00|Aussen
120|16.2|0.0|91|0|14.73|0.0|2019-05-21 09:50:00|Aussen
120|16.2|0.0|91|0|14.73|0.0|2019-05-21 09:55:00|Aussen
120|16.3|0.0|91|0|14.83|0.0|2019-05-21 10:00:00|Aussen
120|16.4|0.0|91|0|14.93|0.0|2019-05-21 10:05:00|Aussen
```

Another way to get the result using mode csv and nested tables.

```
sqlite> .mode csv
sqlite> select d.name, t.* from devicestatus d, temperature t where d.ID=t.devicerowid and
t.devicerowid=120 limit 5;

Name,DeviceRowID,Temperature,Chill,Humidity,Barometer,DewPoint,SetPoint,Date
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:15:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:20:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:25:00"
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:30:00"
```

```
"Aussen Temperatur",120,5.1,0.0,89,0,3.44,0.0,"2019-12-16 11:35:00"
```

Select Temperature Device idx=120 specific day.

If using select with between datetime('2019-05-21') and datetime('2019-05-22');, the first entry of the last date (i.e. 2019-05-22) is also shown.

```
sqlite> select * from Temperature, (select Name from devicestatus where id=120) as Name where
devicerowid=120 and date >=datetime('2019-05-21') and date < datetime('2019-05-22');

120|17.5|0.0|84|0|14.77|0.0|2019-05-21 13:35:00|Aussen
120|17.5|0.0|84|0|14.77|0.0|2019-05-21 13:40:01|Aussen
…
```

Select Plans

The .header on and .column mode are set prior select command.

```
sqlite> .header on
sqlite> .mode column
sqlite> select * from plans;

ID          Order       Name             FloorplanID  Area
----------  ----------  ---------------  -----------  ----------
1           1           $Hidden Devices  0
2           9           Information      1
4           12          Termine          0
5           18          Volumio          3            123,118
8           17          Hardware Monito  0
11          7           Esszimmer (EZ)   0
12          13          Wohnzimmer (WZ)  0
13          4           Bad              0
14          5           Dusche           0
15          11          MakeLab          0
16          8           Haus             0
17          10          Keller           0
```

Select Roomplans with associated Devices

```
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan        PlanID      Device            Idx
----------  ----------  ----------------  ----------
Bad         13          Hzg Bad Batterie  209
Bad         13          Hzg Bad Sollwert  207
Bad         13          Hzg Bad Temperat  208
Dusche      14          Hzg Dusche Batte  200
Dusche      14          Hzg Dusche Sollw  198
Dusche      14          Hzg Dusche Tempe  199
… and more …
```

Another output in CSV format with header

Note

If the device name contains blanks, the name is put between "", i.e. “Device Name"

```
cd /home/pi/domoticz
sqlite3

sqlite> .open domoticz.db
sqlite> .header on
sqlite> .mode csv
sqlite> select p.Name as Plan,p.ID as PlanID,d.Name as Device,d.ID as Idx from devicestatus d,
devicetoplansmap m, plans p WHERE d.ID=m.DeviceRowID and p.ID=m.PlanID order by p.Name;

Plan,PlanID,Device,Idx
Bad,13,"Hzg Bad Batterie",209
Bad,13,"Hzg Bad Sollwert",207
Bad,13,"Hzg Bad Temperatur",208
Dusche,14,"Hzg Dusche Batterie",200
Dusche,14,"Hzg Dusche Sollwert",198
```

```
Dusche,14,"Hzg Dusche Temperatur",199
"Esszimmer (EZ)",11,"Hzg EZ Batterie",206
"Esszimmer (EZ)",11,"Hzg EZ Sollwert",204
"Esszimmer (EZ)",11,"Hzg EZ Temperatur",205
"Esszimmer (EZ)",11,"Hue EZ",111
… and more …
```

Select Setpoint Timers

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

```
sqlite> .header on
sqlite> .mode csv
sqlite> SELECT d.Name, t.* FROM DeviceStatus d, SetpointTimers t WHERE d.ID=t.DeviceRowID;

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

Delete LightingLog Rows Device idx=161

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

```
Check if there are entries:
sqlite> select * from lightinglog where devicerowid=161;
161|0|Security Main Door changed to Alarm + Tamper|2019-05-21 09:01:07|
161|0|Security Main Door changed to Normal|2019-05-21 09:05:00|
161|0|Hue MakeLab changed to Off|2019-05-21 16:16:11|
… and many more …

Delete the rows:
sqlite> delete from lightinglog where devicerowid=161;

Check again if there are entries = select returns no entries:
sqlite> select * from lightinglog where devicerowid=161;
sqlite>
```

Delete Temperature Rows Date < 2018-08-28

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

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

Check if select returns no entries:
sqlite> SELECT * FROM [Temperature] WHERE [DeviceRowID] = 28;
sqlite>
```

Delete all data device idx=44

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

```
sqlite> DELETE FROM Meter WHERE DeviceRowID=44
```

```
sqlite> SELECT * FROM Meter WHERE DeviceRowID=44

sqlite> DELETE FROM MultiMeter_Calendar WHERE DeviceRowID=44

sqlite> SELECT * FROM MultiMeter_Calendar WHERE DeviceRowID=44
```

Note
The SQL statements are DELETE followed by a SELECT Query to check if no result.

Updating Device

Be careful on using – this 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 Statements from File

Executing SQL statements from a text file, is an option to clean database entries (DELETE) or select multiple entries (SELECT).

```
.read FILE
```

If the path is not specified, the file read is taken from the current folder.
If the file contains multiple SQL statements, ensure to close the command with ";".

Example

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

Define
File: /home/pi/domoticz/test.sql

```
select * from lightinglog where devicerowid=97;
select * from lightinglog where devicerowid=98;
```

Execute Interactive

```
cd domoticz
sqlite3
sqlite> .open domoticz.db
sqlite> .header on
sqlite> .read test.sql
97|0|0 J 97 T|2019-05-21 01:00:00|
97|0|0 J 98 T|2019-05-22 01:00:00|
98|0|31-05-2019|2019-05-21 00:30:00|
98|0|31-05-2019|2019-05-22 00:30:00|
```

Execute Command-Line

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/test.sql"
```

Note
See next for more info on running from the command-line.

SQL Command Line

Using the sqlite3 tool with arguments from the command line

- Argument1: The full path to the Domoticz database
- Argument 2: The SQL dot command ".read path-to-sql-command-file"

The SQL statements are stored in files, located in folder

```
/home/pi/domoticz/scripts/dzVents/scripts/
```

and read by the SQL dot command

```
.read
```

If running multiple SQL statements, separate the SQL statements by ";".

The command-line can also be used from dzVents Lua scripts
(see below example dzVents Lua Execute SQLite).

Example SQLite3 Command-Line with the two arguments

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-
all.sql"
```

The SQL file contains two statements (taken from the function Time Control):

```
-- Domoticz Home Automation Workbook - Function Time Control
-- SQL-Command-File: time-control-all.sql
-- Table [LightingLog], Field [DeviceRowID] = 101;
-- Calculate for all rows the time difference in hours for the subsequent rows where nValue is 1
(switch is on)
-- Run from the command line with:
-- sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-
all.sql"

.header on
-- List the blocks
WITH rows AS
  (
  SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM    [lightinglog]
  WHERE   [devicerowid] = 101
  )
SELECT startdata.date AS Start,
   enddata.date AS End,
   ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
   -- Round(Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 As Real),2) AS
Minutes
   -- Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 * 60 As Integer) AS Secs
FROM rows startdata
JOIN rows enddata
ON   startdata.rownumber = enddata.rownumber - 1
WHERE  startdata.nValue = 1;

-- Show the total
WITH rows AS
  (
  SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM    [lightinglog]
  WHERE   [devicerowid] = 101
  )
```

```
SELECT ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),4 ) AS
Total
FROM rows startdata
JOIN rows enddata
ON   startdata.rownumber = enddata.rownumber - 1
WHERE  startdata.nValue = 1;
```

Examples Data Selection

The next examples are taken from the function Time Control and use:

- Domoticz database “domoticz.db”
- Table ”LightingLog”
- Fields ”DeviceRowID”, ”nValue” and ”Date” for queries
- Switch device with Idx=101 - stored in field ”DeviceRowID” and switch state in field ”nValue” with values 0 (=Off) and 1 (=On).

Select All Entries Switch Idx=101 (interactive)

```
sqlite> select * from [LightingLog] where [DeviceRowID] = 101;

DeviceRowID|nValue|sValue|User|Date
101|1|0|Admin|2020-02-19 19:10:57
101|0|0|Admin|2020-02-19 19:12:05
101|1|0|Admin|2020-02-19 19:14:17
101|0|0|Admin|2020-02-19 19:18:26
101|1|0|Admin|2020-02-20 10:12:52
101|0|0|Admin|2020-02-20 10:16:16
```

Select Today Entries Switch Idx=101 (interactive)

The entries are listed with a subsequent row number (window function ROW_NUMBER(), which is used in next examples to calculate the time difference between switch On & Off state. The state is subsequent, which can be used to determine the time difference between on and off. To get the time difference, the subsequent state must be 1,0,1,0..., therefor ensure the switch is turned off.

```
sqlite> select ROW_NUMBER() OVER () Nr,* from [LightingLog] where [DeviceRowID] = 101 and [Date] >=
date('now');

Nr|DeviceRowID|nValue|sValue|User|Date
1|101|1|0|Admin|2020-02-20 10:12:52
2|101|0|0|Admin|2020-02-20 10:16:16
3|101|1|0|Admin|2020-02-20 10:25:55
4|101|0|0|Admin|2020-02-20 11:14:20
```

Select All Activity Blocks

This is a more complex SQL statement to get

a) the difference in time (hours, minutes or seconds) between switch state On (nValue=1) and Off (nValue=0). The time between on & off is a block.
b) the total time difference.

There are two SQL statements used for a) and b).

The SQL statement are

- separated by ";" to enable executing multiple SQL statements.

- stored in a file called time-control-all.sql
  (located in folder /home/pi/domoticz/scripts/dzVents/scripts/) and read by sql dot command .read.

Define SQL Statements

```
-- SQL-Command-File: time-control-all.sql
.header on
WITH rows AS
  (
  SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM    [lightinglog]
  WHERE   [devicerowid] = 101
  )
SELECT startdata.date AS Start,
   enddata.date AS End,
   ROUND(CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real),3) AS Hours
   -- Round(Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 As Real),2) AS
Minutes
   -- Cast( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 * 60 * 60 As Integer) AS Secs
FROM rows startdata
JOIN rows enddata
ON   startdata.rownumber = enddata.rownumber - 1
WHERE  startdata.nValue = 1;

WITH rows AS
  (
  SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber
  FROM    [lightinglog]
  WHERE   [devicerowid] = 101
  )
SELECT ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),4 ) AS
Total
FROM rows startdata
JOIN rows enddata
ON   startdata.rownumber = enddata.rownumber - 1
WHERE  startdata.nValue = 1;
```

Execute SQL Statements

Run from the command-line:

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-
all.sql"

Start|End|Hours
2020-02-20 10:12:52|2020-02-20 10:16:16|0.057
2020-02-20 10:25:55|2020-02-20 11:14:20|0.807
2020-02-20 15:55:30|2020-02-20 16:22:14|0.446
2020-02-20 16:22:37|2020-02-20 16:45:08|0.375
2020-02-20 17:06:55|2020-02-20 17:06:57|0.001
2020-02-21 09:39:23|2020-02-21 09:50:17|0.182
Total
1.8667
```

Select Today Activity Blocks

Basically as previous, but with an additional WHERE clause "[date] >= date('now')".

Define SQL Statements (truncated)

```
-- SQL-Command-File: time-control-today.sql
.header on
WITH rows AS (SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber   FROM [lightinglog] WHERE
[devicerowid] = 101 AND [date] >= date('now'))
SELECT startdata.date AS Start, enddata.date AS End, ROUND(CAST( (JulianDay(enddata.date) -
JulianDay(startdata.date)) * 24 As Real),3) AS Hours
FROM rows startdata JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1 WHERE
  startdata.nValue = 1;
```

```
WITH rows AS (SELECT  *, ROW_NUMBER() OVER (ORDER BY Date) AS rownumber FROM [lightinglog] WHERE
[devicerowid] = 101 AND [date] >= date('now'))
SELECT ROUND( SUM( CAST( (JulianDay(enddata.date) - JulianDay(startdata.date)) * 24 As Real) ),3 ) AS
Total
FROM rows startdata JOIN rows enddata ON startdata.rownumber = enddata.rownumber - 1 WHERE
startdata.nValue = 1;
```

Execute SQLite Command Line with result

```
sqlite3 /home/pi/domoticz/domoticz.db ".read /home/pi/domoticz/scripts/dzVents/scripts/time-control-
today.sql"

Start|End|Hours
2020-02-20 10:12:52|2020-02-20 10:16:16|0.057
2020-02-20 10:25:55|2020-02-20 11:14:20|0.807
2020-02-20 15:55:30|2020-02-20 16:22:14|0.446
2020-02-20 16:22:37|2020-02-20 16:45:08|0.375
2020-02-20 17:06:55|2020-02-20 17:06:57|0.001
Total
1.685
```

Notes
To display the time only, use the function strftime('%H:%M',startdata.date) and strftime('%H:%M',enddata.date). See next.

dzVents Lua Execute SQLite

An example of a dzVents Lua script executing the sql-command-file "time-control-today".
The script is triggered by a switch (Switch Type Push On Button, idx=102) and executes the sql-input-file.
The script creates some additional Domoticz log entries to explore how to use further.

The output of the sql-input-file (as shown previous) is captured and used to update a text device (idx=106) by switching the push-on-button.
The format of the output can be set by using the sqlite dot command ".mode".

Text Device
Some examples with various .mode commands - the first without mode. Try out which suits best.

Mode command not used (default)	.mode tabs	.mode colums	.mode line
2020-02-21 Start|End|Hours 09:39|09:50|0.182 10:44|10:48|0.06 10:56|11:00|0.069 11:00|11:02|0.033 Total 0.344	2020-02-21 Start End Hours 09:39 09:50 0.182 10:44 10:48 0.06 10:56 11:00 0.069 11:00 11:02 0.033 Total 0.344	2020-02-21 Start End Hours ---------- ---------- - --------- 09:39 09:50 0.182 10:44 10:48 0.06 10:56 11:00 0.069 11:00 11:02 0.033 Total ---------- 0.344	2020-02-21 Start = 09:39 End = 09:50 Hours = 0.182 Start = 10:44 End = 10:48 Hours = 0.06 Start = 10:56 End = 11:00 Hours = 0.069 Start = 11:00 End = 11:02 Hours = 0.033 Total = 0.344
.mode list .separator " - "			
2020-02-21			

Start - End - Hours 09:39 - 09:50 - 0.182 10:44 - 10:48 - 0.06 10:56 - 11:00 - 0.069 11:00 - 11:02 - 0.033 Total 0.344			

dzVents Lua Script

Source: time_control_today.lua

```
-- Script triggered by switch (Switch Type Push On Button, idx=102) and executes sql-command-file.
-- Output of sql-command-file is captured and used to update a text device (idx=106).
-- Devices
-- Light/Switch,Switch,On/Off,Time Control Today,Idx: 102
-- General,Text,Time Control Status,Idx: 106.

IDXSWITCH = 102
IDXSTATUS = 106
SQLCMD = "sqlite3 /home/pi/domoticz/domoticz.db '.read /home/pi/domoticz/scripts/dzVents/scripts/time-
control-today.sql'"

-- split a string by delimiter
-- return array, i.e. array[1], array[2]
function splitstring(s, delimiter)
    local result = {};
    for match in (s..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match);
    end
    return result;
end

local function osExecute(cmd)
    local fileHandle     = assert(io.popen(cmd, 'r'))
    local commandOutput  = assert(fileHandle:read('*a'))
    local returnTable    = {fileHandle:close()}
    -- rc[3] contains returnCode
    return commandOutput,returnTable[3]
end

return {
  on = {
    devices = { IDXSWITCH },
    },
  execute = function(domoticz, item)
      if (item.isDevice) then
        if (domoticz.devices(IDXSWITCH).state == "On") then
            output,rt = osExecute(SQLCMD)
                 -- local handle = os.execute(sqlcmd)
                 rawdate = domoticz.time.rawDate .. " "
                 -- add the date to the output
                 output = string.format("%s\n%s",rawdate,output)
                 domoticz.log(output, domoticz.LOG_INFO)
                 -- update the status device
                  domoticz.devices(IDXSTATUS).updateText(output)
                 -- Returncode: 0=OK
                 domoticz.log(tostring(rt), domoticz.LOG_INFO)
                 -- convert the output to a table. Use #outputtable to get the number of entries.
                 outputtable = splitstring(output, "\n")
                 domoticz.log(outputtable, domoticz.LOG_INFO)
             end
    end
     end
}
```

Domoticz Log

```
2020-02-21 11:43:40.343 (VirtualDevices) Light/Switch (Time Control Today)
2020-02-21 11:43:40.327 Status: User: Admin initiated a switch command (102/Time Control Today/On)
2020-02-21 11:43:40.481 Status: dzVents: Info: Handling events for: "Time Control Today", value: "On"
2020-02-21 11:43:40.481 Status: dzVents: Info: ------ Start internal script: time_control_today:
Device: "Time Control Today (VirtualDevices)", Index: 102
```

```
2020-02-21 11:43:40.507 Status: dzVents: Info: 2020-02-21
2020-02-21 11:43:40.507 Start|End|Hours
2020-02-21 11:43:40.507 09:39|09:50|0.182
2020-02-21 11:43:40.507 10:44|10:48|0.06
2020-02-21 11:43:40.507 10:56|11:00|0.069
2020-02-21 11:43:40.507 11:00|11:02|0.033
2020-02-21 11:43:40.507 Total
2020-02-21 11:43:40.507 0.344
2020-02-21 11:43:40.507
2020-02-21 11:43:40.508 Status: dzVents: Info: 0
2020-02-21 11:43:40.508 Status: dzVents: Info: {"2020-02-21 ", "Start|End|Hours", "09:39|09:50|0.182",
"10:44|10:48|0.06", "10:56|11:00|0.069", "11:00|11:02|0.033", "Total", "0.344", ""}
2020-02-21 11:43:40.509 Status: dzVents: Info: ------ Finished time_control_today
```

Python Scripts

Various examples of Python scripts to perform SQL operations.
The lastest version of the scripts can be found in the scripts folder (file names sql-*.py), plus some additional scripts not documented here.

Select Device Status

Select the device status for all devices from the Domoticz database table [DeviceStatus].
Scrip name: sql-select-devicestatus.py

```
…
#!/usr/bin/env python
## File: sql-select-devicestatus.py
## Project: Domoticz-Homeautomation-Workbook
## Purpose: Test select device status for all devices; select temperatures.
## Author:  Robert W.B. Linn
## Version: 20191218

## Table(s) used
## CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT NULL, [DeviceID]
VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(100) DEFAULT Unknown, [Used] INTEGER
DEFAULT 0, [Type] INTEGER NOT NULL, [SubType] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0,
[Favorite] INTEGER DEFAULT 0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0,
[nValue] INTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFAULT
(datetime('now','localtime')),[Order] INTEGER BIGINT(10) default 0, [AddjValue] FLOAT DEFAULT 0,
[AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0, [AddjMulti2] FLOAT DEFAULT 1, [StrParam1]
VARCHAR(200) DEFAULT '', [StrParam2] VARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0,
[Protected] INTEGER DEFAULT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '',
[Options] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL);

import sqlite3
from sqlite3 import Error
import time
import os

# Define globals
version="SQL Test v20180828"
# Domoticz database path
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

###
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_device_status(conn):
    """
    Query all rows in the tasks DeviceStatus
    :param conn: the Connection object
    :return:
    """
    print("Select Device Status")
```

```
    cur = conn.cursor()
    cur.execute("SELECT * FROM DeviceStatus")

    rows = cur.fetchall()

    for row in rows:
        print(row)

def main():
    print(version)

    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection
        conn = create_connection(database)
        # select data
        with conn:
            select_all_device_status(conn)

        conn.close()

if __name__ == '__main__':
    main()
```

```
python3 sql-select-devicestatus.py

SQL Test v20180828
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Device Status
(1, 1, '0000044C', 1, 'RPi RAM Usage', 1, 243, 6, 0, 1, 12, 255, 0, '32.35', '2019-12-18 09:21:47',
104, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, 'RPi Memory Usage increases over time and needs to be
watched.', None, None)
(2, 1, '0000044E', 1, 'RPi HDD /boot', 0, 243, 6, 0, 0, 12, 255, 0, '1.50', '2019-12-18 09:19:16', 1,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, '', None, None)
(3, 1, '0000044F', 1, 'RPi Disc Usage', 1, 243, 6, 0, 1, 12, 255, 0, '32.86', '2019-12-18 09:19:16',
105, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, '', None, None)
(4, 1, '1', 1, 'RPi CPU Temp', 1, 80, 5, 0, 0, 12, 255, 0, '41.9', '2019-12-18 09:21:47', 117, 0.0,
1.0, 0.0, 1.0, '', '', 0, 0, 0, '', None, None)
(5, 1, '0000044D', 1, 'RPi CPU Usage', 1, 243, 6, 0, 0, 12, 255, 0, '3.55', '2019-12-18 09:21:36', 11,
0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, '', None, None)
...
and many more
```

Select Device Status idx=201

Script name: sql-select-idx201.py

```
#!/usr/bin/env python
## File: sql-select-idx201.py
## Project:Domoticz-Homeautomation-Workbook
## Purpose:Test selecting device status for device with idx=201
## Author: Robert W.B. Linn
## Version:20191218
import sqlite3
from sqlite3 import Error
import time
import os

version="SQL Test v20191218"
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: Database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_device_status(conn,idx):
    """
    Query rows table DeviceStatus for a specific device ID (=Idx)
    :param conn: connection object
    :param idx: idx of the device
    :return:
    """
    print("Select Device Status for Idx=%s " % (idx))
    cur = conn.cursor()
    table = "DeviceStatus"

    sql = "SELECT group_concat(name, ', ') FROM pragma_table_info(?)"
    args = (table,)
    cur.execute(sql, args)
    colnames = cur.fetchone()
    print(colnames[0])

    sql = "SELECT * FROM {tn} WHERE ID={idx}"
    cur.execute(sql.format(tn=table, idx=str(idx)))
    devicedata = cur.fetchall()
    print(devicedata)
    print("***")
    for row in devicedata:
        print(row)                      #all fields as tupels
        print("-----")
        print("ID (=Idx)=%s" % row[0])  #id with value 201
        print("-----")
        print(row[0],row[1],row[2])     #more fields
        print("***")

def main():
    print(version)
    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection
```

```
        conn = create_connection(database)
        with conn:
            select_device_status(conn, 201)

        conn.close()

if __name__ == '__main__':
    main()
```

```
python3 sql-select-idx201.py

SQL Test v20191217
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Device Status for Idx=201
ID, HardwareID, DeviceID, Unit, Name, Used, Type, SubType, SwitchType, Favorite, SignalLevel,
BatteryLevel, nValue, sValue, LastUpdate, Order, AddjValue, AddjMulti, AddjValue2, AddjMulti2,
StrParam1, StrParam2, LastLevel, Protected, CustomImage, Description, Options, Color
[(201, 20, '00140001', 1, 'Hzg WZ Sollwert', 1, 242, 1, 0, 0, 12, 255, 1, '21.0', '2019-12-17
06:15:02', 181, 0.0, 1.0, 0.0, 1.0, '', '', 0, 0, 0, '', None, '')]
***
201
201 20 00140001
***
```

Delete Temperature Rows

<TODO> Add function, i.e. delete_temperature_by_date(conn, date).
Script name: sql-delete-temperature-threshold.py

```
#!/usr/bin/env python
## File: sql-delete-temperature-threshold.py
## Project: Domoticz-Homeautomation-Workbook
## Purpose: Test deleting records from the SQLite database domoticz.db table Temperature
## Author: Robert W.B. Linn
## Version: 20180828

## Table(s) used
## CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill]
FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT
0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));

import sqlite3
from sqlite3 import Error
import time
import os

# Define globals
version="SQL Test v20180828"
# Domoticz database path
# Tests
database="/home/pi/python/sql/domoticz.db"
# Production
#database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

###
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_temperatures(conn):
    """
    Query all rows in the tasks Temperature
    :param conn: the Connection object
    :return:
    """
    print("Select all Temperatures")
    cur = conn.cursor()
    cur.execute("SELECT * FROM Temperature")

    rows = cur.fetchall()

    for row in rows:
        print(row)


def select_temperature_by_threshold(conn, threshold):
    """
    Query temperatures by threshold
    :param conn: the Connection object
    :param threshold:
```

```
    :return:
    """
    print("Select Temperature by Threshold %s" % (threshold))
    cur = conn.cursor()
    cur.execute("SELECT * FROM Temperature WHERE Temperature>?", (threshold,))

    rows = cur.fetchall()

    for row in rows:
        print(row)

def delete_temperature_by_threshold(conn, threshold):
    """
    Delete all temperatures by threshold
    :param conn: the Connection object
    :param threshold:
    :return:
    """
    print("Delete Temperature by Threshold %s" % (threshold))
    cur = conn.cursor()
    cur.execute("DELETE FROM Temperature WHERE Temperature>?", (threshold,))
    conn.commit()

def main():
    print(version)

    if DatabaseExists(database):
        print("Connecting to the database...",database)
        # create a database connection
        conn = create_connection(database)
        with conn:
            threshold = 40.0

            select_temperature_by_threshold(conn, threshold)

            delete_temperature_by_threshold(conn, threshold)

            # Check if zero
            select_temperature_by_threshold(conn, threshold)

        conn.close()

if __name__ == '__main__':
    main()
```

```
python3 sql-delete-test.py

SQL Test v20180828
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Temperature by Threshold 40.0
(2, 40.8, 0.0, 0, 0, 0.0, 0.0, '2018-08-28 19:20:00')
Delete Temperature by Threshold 40.0
Select Temperature by Threshold 40.0
```

Tinkerforge

Purpose

- To explore how to interact between Domoticz and the Tinkerforge Building Blocks.
- To explore how to write Python scripts with the Tinkerforge Python API bindings triggered by dzVents.
- To learn how to write Domoticz Hardware Plugins with the Domoticz Python Framework using the Tinkerforge Python API bindings.

TinkerForge is an open source hardware platform of stackable microcontroller building blocks (Bricks) that can control different modules (Bricklets).
There is wide range of bricklets enabling to build all kind of solutions.
The building blocks can be controlled using a variety of high-level programming languages through API bindings. The focus is on using the Python API bindings - unfortunate Lua is not supported (author's favorite Domoticz scripting language through dzVents).

Recommend to visit the Tinkerforge website to explore more indepth whats possible with the Building Blocks.

The setup to explore Tinkerforge uses following hard- and software enabling to test the various concepts.

Hardware

- Tinkerforge Master Brick 2.1 connected, via USB, to the Raspberry Pi 3B+ running the Domoticz Development System v4.x
- Ambient Light Bricklet 2.0
- IO4 Bricklet 2.0
- LCD 20x4 Bricklet 1.2
- RGB LED Bricklet 2.0

Software

- Raspberry Pi Raspian Debian Linux Buster 4.19.93-v7+ #1290
- Domoticz Home Automation System 4.11696 (BETA)
- Tinkerforge Python API-Binding 2.1.24
- Python 3.7.3, GCC 8.2.0
- *Note:* The versions are subject to change.

Installed the Tinkerforge Brick Daemon (brickd) and the Brick Viewer (brickv) on the Domoticz Development System.
The Brick Daemon acts as a bridge between the Bricks/Bricklets and the application.
The Brick Viewer provides a graphical interface for testing Bricks and Bricklets.
The Python API binding is used to control (via TCP/IP connection) the Bricks and Bricklets from the application.

Concepts

Several concepts explored to interact between Domoticz and Tinkerforge.

Interaction means:

- **Get** data from a Tinkerforge Bricklet which is taken by Domoticz to action accordingly
- **Set** data from Domoticz to the Tinkerforge Bricklet
- **Listen via Callbacks** to Tinkerforge Bricklet changes which is taken by Domoticz to action accordingly

Tinkerforge Script
To interact via dzVents by executing Python scripts using the Tinkerforge Python API binding for a bricklet (getter, setter).

Domoticz Plugin
To interact using the Domoticz Python Framework with the Tinkerforge Python API Binding (getter, setter, callbacks).

Tinkerforge Service
To interact between Domoticz and Tinkerforge v.v. using a service running a Tinkerforge Python Script for one or more bricklets depending functionality (getter, setter, callbacks).

Tinkerforge Script

To interact between Domoticz & Tinkerforge using dzVents by executing Python scripts with the Tinkerforge Python API binding for the selected bricklet.

This concept is basically a setter or getter with following sequence:

1. dzVents calls the Tinkerforge Python script with parameter JSON format
2. Tinkerforge Python script
    a. makes a Tinkerforge IP connection
    b. sets or gets data, i.e. bricklet configuration or value
    c. disconnects the Tinkerforge IP connection
    d. returns data and status in JSON format
3. dzVents handles returned data & status, i.e. update Domoticz devices or control action

There is no ongoing connection between Domoticz and Tinkerforge nor is callback used.
It is not required to create or use dedicated Domoticz device(s) – depends on the actions.
To use callback, it is recommended to create a Domoticz Python Plugin or a Service.
As an extension it is also possible to create a combined Tinkerforge Python script for multiple bricklets, with logic integrated or controlled via arguments.

Decided this concept, i.e. running external Python scripts doing the Tinkerforge work, as it is easier to test & maintain (API changes, updates).

Example GET Illuminance (Lux) from Ambient Light Bricklet 2.0 (snippet)

dzVents script “tfalv2_get_lux.lua” executes, triggered by a Timer (every minute), the Tinkerforge Python script “tfalv2_get_lux.py” with the Ambient Light Bricklet 2.0 UID (obtained using the Tinkerforge Brick Viewer):

```
python3 tfalv2_get_lux.py '{"UID":"yyc"}'
```

Tinkerforge Python Script returns if ok, the Lux value:

```
{"status": "OK", "title": "", "lux":"146.21"}
```

OR in case of an error

```
{"status": "ERROR", "title": "Error message"}
```

dzVents to decode returned JSON text and update the Domoticz Device with the Lux value:

```
local IDXLUX = 107
…
local result = JSON:decode(output)
domoticz.devices(IDXLUX).updateLux(result.lux)
```

Example SET Color of RGB LED Bricklet 2.0 (snippet)

dzVents script “tfrgbledv2_set_color.lua” executes, triggered by a change of the Domoticz Device Selector Switch value (0=OFF,10=RED,20=YELLOW,30=GREEN), the Tinkerforge Python script “tfrgbledv2_set_color.py” with the RGB LED bricklet 2.0 UID (obtained using the Tinkerforge Brick Viewer) and the R,G,B color values – example color yellow 255-255-0.

```
python3 tf_rgbledv2_set_color.py '{"UID":"Jng","R":255,"G":255,"B":0}'
```

Tinkerforge Python Script returns if ok:

```
{"status": "OK", "title": "255,255,0"}
```

OR in case of an error

```
{"status": "ERROR", "title": "Error message"}
```

Examples (with reference to the dzVents & Python scripts)

Tinkerforge Bricklet	Example	dzVents Script (.lua)	Trigger	Tinkerforge Script (.py)	Domoticz Device (idx, hardware, name, type, subtype, switchtype, data)	
Ambient Light 2.0 [yyc]	Get illuminance and update Domoticz device	tfalv2_get_lux	Timer Every Minute	tfalv2_get_lux	107, VirtualDevices, MakeLab Lux, Lux, Lux, n/a 2974 Lux	
RGB LED 2.0 [Jng]	Set RGB color triggered by Domoticz selector switch	tfrgbledv2_set_color	Device	tfrgbledv2_set_color	12, VirtualDevices, RGBLED - Traffic Light Switch, Light/Switch, Switch Selector Set Level: 10 %	
LCD 20x4 2.0 [BHN]	Set LCD text HH:mm triggered by dzVents very minute	tflcd20x4_set_clock	Timer Every Minute	tflcd20x4_set_text	Not used. Tinkerforge script writes direct to the LCD Bricklet.	
LCD 20x4 2.0 [BHN]	Set LCD text Raspberry Pi info triggered by dzVents every minute	tflcd20x4_set_rpi_monitor	Timer Every Minute	tflcd20x4_set_text	112, VirtualDevices, LCD Text General, Text Note: Text and display taken different times.	

The latest version of the scripts can be found in the file tf<bricklet>_set|get_action.py.

To get started, two templates are available for a dzVents & Python script.
Source: tfbricklet_set_get.lua

```
-- Domoticz Home Automation Workbook - Function Tinkerforge
-- dzVents Automation Script: tfbricklet_set_get
-- Template for setting or getting bricklet values.
-- The dzVents Lua script controlles the functionality and run a Python script executing Tinkerforge
API functions for the bricklet.
-- The logic is handled by the Domoticz script whereas the Python script sets or gets bricklet vaue(s)
only.
-- Python Script
-- The script is triggered every minute or by a device (i.e. switch) and executes python script.
-- The python script expects a string argument (enclosed in '') with key:value pairs:
-- '{"HOST":"IP-Address","PORT":4223,"UID":"yyc"}'
-- The UID argument is mandatory, HOST and PORT are optional. The keys must in UPPERCASE!
-- In addition key:value pairs can be defined, depending on the functionality and if getter or setter.
-- Examples Bricklets:
-- RGB LED 2.0: set color = '{"HOST":"IP-Address","PORT":4223,"UID":"Jng","R":255,"G":255,"B":0}'
-- LCD 20x4: set text 4 lines =
'{"UID":"BHN","LCDLINES":[{"line":0,"position":0,"clear":1,"text":"R=255"},{"line":1,"position":5,"clea
r":1,"text":"G=0"},{"line":2,"position":10,"clear":1,"text":"B=0"},{"line":3,"position":15,"clear":1,"t
ext":"Color"}]}'
-- Result
-- The result of the command is a JSON formatted string containing the key:value pairs:
-- {"status": "OK", "title": ""} or {"status": "ERROR", "title": "Error message"}
-- For a getter, the requested values are added to the result.
-- Examples Bricklets:
-- Ambient Light 2.0: get lux = {"status": "OK", "title": "", "lux":"146.21"}
--
-- 20200226 by rwbl

-- json library
local JSON = (loadfile "/home/pi/domoticz/scripts/lua/JSON.lua")()  -- For Linux
-- define the path to the python script
local CMD = "python3 /home/pi/domoticz/scripts/python/tfbricklet_set_get.py"

-- Idx of the devices used, i.e. a switch (set) or a device (get) to be updated
local IDXDEVICE = NNN
-- TF Bricklet parameter
local TFUID = "UID"

-- run external command
local function osExecute(cmd)
```

```
    local fileHandle     = assert(io.popen(cmd, 'r'))
    local commandOutput  = assert(fileHandle:read('*a'))
    local returnTable    = {fileHandle:close()}
    -- rc[3] contains returnCode
    return commandOutput,returnTable[3]
end

return {
  on = {
    -- select the function
    timer = { 'every minute' },
    devices = { IDXDEVICE },
     },
  -- select the function
  execute = function(domoticz, timer)
    domoticz.log('Timer event was triggered by ' .. timer.trigger, domoticz.LOG_INFO)

  execute = function(domoticz, device)
    domoticz.log(string.format('Device changed:%s to %s.',device.name, device.state),
domoticz.LOG_INFO)

        -- define the python command with json argument
        -- example: python3 tfbricklet_set_get.py '{"UID":"yyc"}'
        local pycmd = string.format('%s \'{\"UID\":"%s"}\'',CMD,TFUID)
    domoticz.log(pycmd, domoticz.LOG_INFO)

    -- run the external python scripts ad handle return code
    local output,rt = osExecute(pycmd)
        -- domoticz.log(string.format('%d:%s',rt,output), domoticz.LOG_INFO)
        -- decode the result string which is json format, i.e.
        -- {"status": "OK", "title": "", "lux":"146.21"}
        -- {"status": "ERROR", "title": "Error message"}
        local result = JSON:decode(output);
        -- domoticz.log(string.format('TEST Status:%s, Title:%s,
Lux:%s',result.status,result.title,result.lux), domoticz.LOG_INFO)
    -- handle the result of the python command: 0=OK,>0=ERROR
    if rt == 0 then
        -- handle the result of the tinkerforge scripts
        if result.status == "OK" then
       -- setter uses status and result only
                domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_INFO)

       -- getter to get the value and update domoticz deice
                domoticz.log(string.format('Status:%s, Title:%s,
Value:%s',result.status,result.title,result.value), domoticz.LOG_INFO)
                domoticz.devices(IDXDEVICE).updateLux(result.value)
            -- tinkerforge script returns an error
            else
                domoticz.log(string.format('Status:%s, Title:%s',result.status,result.title),
domoticz.LOG_ERROR)
            end
    -- handle the result of the python command
        else
            domoticz.log(string.format('Python command: RT:%d, Output:%s',rt,output),
domoticz.LOG_ERROR)
        end

    end
}
```

Source: tfbricklet_set_get.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
File:       tfbricklet_get_set.py
Project:    Domoticz-Home-Automation-Workbook
Purpose:    Get or set the value of a bricklet.
Depends on: Python API Binding
Description:JSON formatted string as argument with key:value pairs enclosed in ''.
IMPORTANT: all keys must be in uppercase.
Key:Value
"HOST":"String" - IP address of the host running the brick deamon. Default: "localhost" (optional)
"PORT":integer - Port number. Default: 4223 (optional)
```

```
"UID":"String" - UID of the bricklet. Get from the Brick Viewer, i.e. "yyc" (mandatory)
The UID argument is mandatory, HOST and PORT are optional. The keys must in UPPERCASE!

Define additional key:value pairs depending bricklet:
Examples Bricklets:
RGB LED 2.0: set color = '{"HOST":"IP-Address","PORT":4223,"UID":"Jng","R":255,"G":255,"B":0}'
LCD 20x4: set text 4 lines =
'{"UID":"BHN","LCDLINES":[{"line":0,"position":0,"clear":1,"text":"R=255"},{"line":1,"position":5,"clea
r":1,"text":"G=0"},{"line":2,"position":10,"clear":1,"text":"B=0"},{"line":3,"position":15,"clear":1,"t
ext":"Color"}]}'
Result
The result of the command is a JSON formatted string containing the key:value pairs:
{"status": "OK", "title": ""} or {"status": "ERROR", "title": "Error message"}
For a getter, the requested values are added to the result.
Examples Bricklets:
Ambient Light 2.0: get lux = {"status": "OK", "title": "", "lux":"146.21"}

Tests:
Recommend during script development, to run (iterate) the script for a terminal command line without
using Domoticz (dzVents, Lua).
Example: Run to get the lux value of the bricklet connected to the master brick which is connected via
USB with Raspberry Pi:
    cd domoticz/scripts/python
    python3 /home/pi/domoticz/scripts/python/tfalv2_get_lux.py '{"UID":"yyc"}'
    Output:
    {"status": "OK", "title": "", "lux": "177.92"}

Version:    20200226 by rwbl
"""
import sys
import json

# Tinkerforge
from tinkerforge.ip_connection import IPConnection
# Bricklet api - see Tinkerforge documentation
from tinkerforge.bricklet_<BRICKLETAPINAME> import Bricklet<BRICKLETAPINAME>
## Set defaults for connection
HOST = "localhost"
PORT = 4223
UID = "<UID>"

# Arguments Length
ARGVLEN = 2

if __name__ == "__main__":
    status = 1
    # Get the command line argurments
    # print("Arguments: {}".format(str(sys.argv)))
    if len(sys.argv) == ARGVLEN:
        # Parse the command line argument 1 from string to json
        args = json.loads(sys.argv[1])
        if "HOST" in args:
            HOST = args["HOST"]
        if "PORT" in args:
            PORT = int(args["PORT"])
        if "UID" in args:
            UID = args["UID"]
        # Add additional arguments depending bricklet
        if "VALUE" in args:
            VALUE = args["VALUE"]
        # Examples:
        # RGB LED 2.0 set RGB color: '{"UID":"Jng", "R":255,"G":255,"B":0}'
        # LCD 20x4 set text lines: '{"UID":"BHN",
"LCDLINES":[{"line":1,"position":1,"clear":1,"text":"Text"},{"line":2,"position":1,"clear":1,"text":"Te
xt2"}]}'
        #
        # Check arguments
        # print("JSON:H={},P={},U={}".format(HOST,PORT,UID))
    else:
        status = 0
        result = { "status" : "ERROR", "title" : "Wrong number of arguments." }

    if status == 1:
        try:
            # Create IP connection
```

```
        ipcon = IPConnection()
        # Create device object
        devobj = BrickletBRICKLETAPINAME(UID, ipcon)
        # Connect to brickd
        ipcon.connect(HOST, PORT)
        # Check if the bricklet is reachable - else error is triggered
        # Depending the version of the bricklet the UID uses different function:
        # Older bricklets .get_identity() and newer .read_uid()
        # In case the bricklet is not reachable, it takes 1-2 seconds befor a response is given
        devobj.get_identity()
        devobj.read_uid()
        # GET
        value = devobj.get_<value>()
        # define the result containing the mandatory status, title and the value
        result = { "status" : "OK", "title" : "", "value" : "{}".format(value) }

        # SET
        devobj.set_value(<VALUE>)
        result = { "status" : "OK", "title" : "{}".format(<VALUE>) }

        # Disconnect
        ipcon.disconnect()
    except:
        result = { "status" : "ERROR", "title" : "get_value failed. Check master & bricklet." }
# Print the result which is used by the dzVents Lua script
print(json.dumps(result))
```

Domoticz Plugin

To interact using the Domoticz Python Framework with the Tinkerforge Python API Binding. This solution can be applied to dedicated bricklets or solutions containing multiple bricklets with logic control.

More to be found in section Plugin Development.

A selection of Domoticz Tinkerforge Plugins can be found in the repository folder plugins or on GitHub:

- Air Quality Solution (domoticz-plugin-indoor-air-quality-monitor)
- Ambient Light Bricklet 2.0 (domoticz-plugin-tinkerforge-ambientlightv2)
- IO4 Bricklet 2.0 with input callback (domoticz-plugin-tinkerforge-io4v2)
- LCD 20x4 Bricklet Full with buttons callback & configuration options (domoticz-plugin-tinkerforge-lcd20x4)
- LCD 20x4 Bricklet Lite display text only (domoticz-plugin-tinkerforge-lcd20x4-lite)
- RGB LED Bricklet 2.0 (domoticz-plugin-tinkerforge-rgbledv2)
- Soil Moisture Solution (domoticz-plugin-soil-moisture-monitor)
- Tinkerforge Traffic Light with RGB LED Bricklet 2.0 (domoticz-plugin-tinkerforge-trafficlight)

Tinkerforge Service

To interact between Domoticz and Tinkerforge v.v. using a service running a Tinkerforge Python Script for one or more bricklets depending functionality.

The communication between Domoticz and the Service is via MQTT (using the mosquitto broker and the Domoticz MQTT Client Gateway Hardware).

Example IO4 Bricklet 2.0

To interact between Domoticz and Tinkerforge v.v. using a service "tfio4v2-mqtt.service" running the Python script "tfio4v2-mqtt.py".

The example shows how to use a Tinkerforge callback by subscribing to a MQTT topic and how to publish MQTT messages to a MQTT topic.
The Python script uses the paho mqtt client library and the Tinkerforge Python API bindings.

The IO4 Bricklet 2.0 configuration (channel, sensor/actor connected, direction, initial value):

- IO4 Channel 0: LED connected. Output, Low.
- IO4 Channel 1: Push-button connected. Input, Low.

There are 2 Domoticz devices used (Idx,Name,Type,SubType,SwitchType):

- 108,TFIO4 Status,General,Text
- 109,TFIO4 LED Switch,Light/Switch,Switch,On/Off

Outline
By using dzVents, Domoticz can control & take actions. Just some simple examples:

- if the push-button gets pushed, a light can be switched on by dzVents

- if the Raspberry Pi Domoticz system CPU temperature gets to high, switch the RED LED connected to IO4 channel 0 to On

MQTT Communication

The Python script:

Subscribes to topic "domoticz/out" and listens to messages published by Domoticz having Idx=109 (switch).

If the Domoticz switch is set to On/Off, the LED connected to IO4 channel 0 is turned On/Off.

Domoticz Log sample switching On/Off

```
2020-03-01 11:37:51.740 (VirtualDevices) Light/Switch (TF IO4 LED Switch)
2020-03-01 11:37:51.732 Status: User: Admin initiated a switch command (109/TF IO4 LED Switch/On)
2020-03-01 11:38:03.716 (VirtualDevices) Light/Switch (TF IO4 LED Switch)
2020-03-01 11:38:03.708 Status: User: Admin initiated a switch command (109/TF IO4 LED Switch/Off)
```

Publishes, using topic "domoticz/in", a text to Domoticz which updates the text device with Idx=108.

If the push-button connected to IO4 channel 0 is pressed, a message is published to Domoticz. Domoticz updates the text of the device.

Sample text displayed on the text device:

```
IO4 changed:True, value:True, time:2020-03-01 11:38:58
IO4 changed:True, value:False, time:2020-03-01 11:39:05
```

Service File (tfio4v2-mqtt.service)
Login as user Pi and create the service file in folder /home/pi/domoticz/scripts/python:

```
[Unit]
Description=Tinkerforge Domoticz Service IO4V2
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u tfio4v2-mqtt.py
WorkingDirectory=/home/pi/domoticz/scripts/python
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

Copy the service file to the systemd folder

```
sudo cp tfio4v2-mqtt.service /etc/systemd/system/tfio4v2-mqtt.service
```

Start the service

```
sudo systemctl start tfio4v2-mqtt.service
```

Check if the service is running

```
ps ax | grep tfio4v2-mqtt

 4745 ?        Ssl    0:00 /usr/bin/python3 -u tfio4v2-mqtt.py
```

Stop the service

```
sudo systemctl stop tfio4v2-mqtt.service
```

and check again if the service has stopped = nothing to be retuned from the command.

Start Service at Boot
Start automatically on reboot:

```
sudo systemctl enable tfio4v2-mqtt.service
```

The systemctl command can also be used to restart the service or to disable.
Example with output:

```
sudo systemctl enable tfio4v2-mqtt.service
Created symlink /etc/systemd/system/multi-user.target.wants/tfio4v2-mqtt.service →
/etc/systemd/system/tfio4v2-mqtt.service.
```

Bash: tfio4v2-mqtt-start.sh

```
#!/bin/bash
## File:tfio4v2-mqtt-start.sh
## Attributes: make the script executable: sudo chmod +x tfio4v2-mqtt-start.sh
## 20200301 rwbl

cd /home/pi/domoticz/scripts/python
# start the service
echo Starting...
sudo systemctl start tfio4v2-mqtt.service
# check if running
echo check if running...
echo entry should be listed line: pi 14621 1  0 19:16 ? 00:00:00 /usr/bin/python3 -u tfio4v2-mqtt
ps -ef | grep tfio4v2-mqtt
echo done
```

Bash: tfio4v2-mqtt-stop.sh

```
#!/bin/bash
## File:tfio4v2-mqtt-stop.sh
## Attributes: make the script executable: sudo chmod +x tfio4v2-mqtt-stop.sh
## 20200301 rwbl

cd /home/pi/domoticz/scripts/python
# stop the service
echo service stopping...
sudo systemctl stop tfio4v2-mqtt.service
# check if stopped
echo check if stopped...
echo no entry should be listed line: pi 14621 1  0 19:16 ? 00:00:00 /usr/bin/python3 -u tfio4v2-mqtt
ps -ef | grep tfio4v2-mqtt
echo done
```

Service: tfio4v2-mqtt-stop.service

```
[Unit]
Description=Tinkerforge Domoticz Service IO4V2
After=network.target

[Service]
ExecStart=/usr/bin/python3 -u tfio4v2-mqtt.py
```

```
WorkingDirectory=/home/pi/domoticz/scripts/python
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

Python: tfio4v2-mqtt.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
File:       tfio4v2-mqtt.py
Project:    Domoticz-Home-Automation-Workbook
Purpose:    To test mqtt communication between Tinkerforge IO4 Bricklet 2.0 and Domoticz devices or
scripts.
The IO4 Bricklet 2.0 configuration:
IO4 Channel 0: LED connected. Output, High.
IO4 Channel 1: Push-button connected. Input, Low.
There are 2 Domoticz devices used (Idx,Name,Type,SubType,SwitchType):
108,TFIO4 Status,General,Text
109,TFIO4 LED Switch,Light/Switch,Switch,On/Off

To configure the IO4 bricklet, a JSON formatted string with key:value pairs enclosed in '' is required.
* IMPORTANT: all keys must be in uppercase except for the CHANNELS which must be in lowercase.
(Key:Value)
"HOST":"String" - IP address of the host running the brick deamon. Default: "localhost" (optional)
"PORT":integer - Port number. Default: 4223 (optional)
"UID":"String" - UID of the bricklet. Get from the Brick Viewer, i.e. "yyc" (mandatory)
STATUSLED:integer - Set the status LED configuration (see Bricklet documentation)
The UID argument is mandatory, HOST and PORT are optional. The keys must in UPPERCASE!
Additional key:value pairs for the io4v2 bricklet:
"CHANNELS":json array - Channels 0 to 3 with value& direction, ie.
[{"channel":0,"value":1,"direction":"o"},{"channel":1,"value":0,"direction":"i"}]}'
"channel":integer - Channel number 0-3 (mandatory)
"value":integer - State of the channel 0=low (off) 1=high (on)
"direction":String - Direction output "o" or input "i"

Tests
1: Switch the IO4 Channel 0 LED on/off triggered by the Domoticz Switch device (idx=109) (MQTT
subscribe)
2: Set the text of the Domoticz Text device (idx=108) when pushing or release the IO4 Channel 1 push-
button (MQTT publish)
Example running python script from folder /home/pi/domoticz/scripts/python
# IO4 Channel 0: LED connected. Output, High.
# IO4 Channel 1: Push-button connected. Input, Low.
python3 tfio4v2-mqtt.py

The config is set below in the global CONFIG:
'{"UID":"G4d","STATUSLED":2,"CHANNELS":[{"channel":0,"value":0,"direction":"o"},{"channel":1,"value":0,
"direction":"i"}]}'

Recommend during script development, to run (iterate) the script for a terminal command line without
using Domoticz (dzVents, Lua).
Depends on: Tinkerforge Python API Binding, MQTT Python paho client
Version:    20200229 rwbl
"""
# Imports
import sys
import time
from datetime import datetime
## json to handle incoming mqtt messages from domoticz
import json
## mqtt client library for subscribe & publish mqtt messages
import paho.mqtt.client as mqtt
# Tinkerforge
from tinkerforge.ip_connection import IPConnection
## Bricklet api - see Tinkerforge documentation
from tinkerforge.bricklet_io4_v2 import BrickletIO4V2

# Info
```

```
VERSION = "DoTfIO4V2"

# Debug
# DEBUG = True
DEBUG = False

# Configuration
CONFIG =
'{"UID":"G4d","STATUSLED":2,"CHANNELS":[{"channel":0,"value":0,"direction":"o"},{"channel":1,"value":0,
"direction":"i"}]}'

## MQTT connection parameter defaults and domoticz topics publish & subscribe
MQTT_BROKER = "localhost"
MQTT_CLIENT = "tfdomoticzio4"
MQTT_TOPIC_SUB = "domoticz/out"
MQTT_TOPIC_PUB = "domoticz/in"

# Tinkerforge
## Connection parameter defaults
TF_HOST = "localhost"
TF_PORT = 4223
## IO4
IO4_UID = "G4d"
## Set the input channel callback period to 0.25s (250ms) or 500 ...
IO4_CALLBACK_PERIOD = 500
## Set the status LED to heartbeat (see bricklet api doc for the values to be set)
IO4_STATUS_LED = 2
## Channels
IO4_CHANNEL_0 = 0
IO4_CHANNEL_1 = 1

# Domoticz
## (mqtt subscribe) Domoticz switch to turn the io4 led on&off (connected to channel 0)
IDX_SWITCH = 109
## (mqtt publish)Domoticz text device gets updated when io4 button (connected to channel 1) is pressed.
IDX_STATUS = 108

##
## INIT with globals having initial values
##

##
## MQTT functions
##

# mqtt subscribe to topic(s) after successful connection to the mqtt broker
# for this solution only the domoticz mqtt topic "domoticz/out" is subscribed to
def on_connect(client, userdata, flags, rc):
    mqttConnected = True
    if DEBUG == True:
        print("MQTT on_connect: client={},userdata={},flags={},rc={},subscribe topic={}".format(client,
userdata, flags, rc, mqttTopicSub))
    client.subscribe(mqttTopicSub)
    print("MQTT Connected. Waiting for messages...".format())

# mqtt handle incoming messages from domoticz topic "domoticz/in"
# for this topic all domoticz mqtt messages are received (json formatted string)
# the mqtt message is json parsed to get the device idx and take action.
# for this example, the io4 channel 0 is set to high or low, i.e. led on / off
def on_message(client, userdata, message):
    if DEBUG == True:
        print("MQTT on_message: topic={}".format(message.topic))
        print("MQTT on_message: payload={}".format(message.payload.decode("utf-8")))
        print("MQTT on_message: qos={},retail=".format(message.qos,message.retain))
    # decode the message payload to be able to parse
    json_string = str(message.payload.decode("utf-8"))
    # Handle changes of the domoticz switch to turn io4 led channel 0 on or off
    set_io4_channel_value(json_string, IDX_SWITCH, IO4_CHANNEL_0)

# mqtt log any messages
def on_log(client, userdata, level, buf):
    if DEBUG == True:
        print("MQTT on_log:client={},userdata={},level=={},buf={}".format(client, userdata, level,
buf))
```

```
# mqtt create a new instance and start listening.
# prior string to listen to mqtt messages, connect to tinkerforge brick deamon with callback on io4
input channels
def create_mqtt_instance():
    global mqttBroker,mqttClient,mqttTopicSub,mqttTopicPub,mqttConnected

    mqttBroker = MQTT_BROKER
    mqttClient = MQTT_CLIENT
    mqttTopicSub = MQTT_TOPIC_SUB
    mqttTopicPub = MQTT_TOPIC_PUB
    mqttConnected = False

    print("Connecting to Tinkerforge & MQTT...")

    if DEBUG == True:
        print("MQTT create new instance:broker={}client={},".format(mqttBroker,mqttClient))
    mqttClient = mqtt.Client(mqttClient)
    mqttClient.on_connect = on_connect
    mqttClient.on_message = on_message
    mqttClient.on_log = on_log
    if DEBUG == True:
        print("Connecting to the broker", mqttBroker)
    mqttClient.connect(mqttBroker, 1883, 60)
    time.sleep(4)
    # Connect to the tinkerforge brick daemon with json string holding the configuration
    # Instead of constant CONFIG, an argument sys.argv can be used
    if connect_tinkerforge(CONFIG) == True:
        # Loop the matt client
        mqttClient.loop_forever()
    else:
        print("[ERROR] Can not connect to Tinkerforge. Check configuration.")
    return

# mqtt publish new message to domoticz
## Examples payload:
## Text device: '{"command": "udevice", "idx": %s, "svalue": "Hello World at %s"}' %
(IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
def publish_mqtt_message(payload):
    if DEBUG == True:
        print("MQTT publish_mqtt_message:topic={},payload={}".format(mqttTopicSub,payload))
    mqttClient.publish(mqttTopicPub,payload)

def disconnect_mqtt_broker():
    mqttClient.loop_stop()
    if DEBUG == True:
        print("MQTT disconnected mqtt broker={}".format(mqttBroker))

##
## JSON Parsing
##
def get_domoticz_device_property(json_string, idx, property):
    result = ""
    if DEBUG == True:
        print("get_domoticz_device_property: json={},idx={},property={}".format(json_string, idx,
property))
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        name = parsed_json['name']
        value = parsed_json[property]
        result = value
        if DEBUG == True:
            print("get_domoticz_device_property: idx={},name={},property={},value={}".format(idx, name,
property, value))
    return result

## Text device: '{"command": "udevice", "idx": %s, "svalue": "Hello World at %s"}' %
(IDX,datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
def set_domoticz_text_device(idx, svalue):
    if DEBUG == True:
        print("set_domoticz_text_device: idx={},svalue={}".format(idx, svalue))
    payload = '{{"command":"udevice","idx":{},"svalue":"{}"}}'.format(idx,svalue)
    # payload = '{"command":"udevice","idx":%d,"svalue":"%s"}' % (idx, svalue)
    print("set_domoticz_text_device: payload={}".format(payload))
    publish_mqtt_message(payload)
```

```
    if DEBUG == True:
        print("set_domoticz_text_device: idx={},svalue={}".format(idx, svalue))

# set the value of an io4 input channel depending domoticz witch on/off device nvalue
def set_io4_channel_value(json_string, idx, channel):
    # get all idx properties
    # if DEBUG == True:
    #     print("set_io4_channel_value: json={},idx={}".format(json_string, idx))
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        # get the properties of the selected idx
        if DEBUG == True:
            print("set_io4_channel_value: json={},idx={}".format(json_string, idx))
        name = parsed_json["name"]
        value = int(parsed_json["nvalue"])
        io4Dev.set_selected_value(channel,value)
        print("set_io4_channel_value: channel={},idx={},name={},nvalue={}".format(channel,idx, name,
value))

##
## Tinkerforge functions
##

# Connect to the tinkerforge brick deamon, create io4 bricklet device object and set bricklet parameter
# Parameter:
# config - json string with settings
def connect_tinkerforge(config):
    result = False
    # Define the globals for he tinkerforge connection and io4 bricklet
    global tfHost,tfPort
    global io4Uid,io4CallbackPeriod,io4StatusLED,io4Dev

    # Get the config parameter
    if DEBUG == True:
        print("Config=:{}".format(config))

    # Parse the config string
    args = json.loads(config)
    # Assign to the vars
    tfHost = args["HOST"] if "HOST" in args else TF_HOST
    tfPort = int(args["PORT"]) if "PORT" in args else TF_PORT
    io4Uid = args["UID"] if "UID" in args else IO4_UID
    io4StatusLED = int(args["STATUSLED"]) if "STATUSLED" in args else IO4_STATUS_LED
    io4CallbackPeriod = IO4_CALLBACK_PERIOD
    io4Channels = args["CHANNELS"] if "CHANNELS" in args else ""

    # Check arguments
    if DEBUG == True:
        print("connect tinkerforge:
host={},port={},uid={},callbackperiod={},channels={}".format(tfHost,tfPort,io4Uid,io4CallbackPeriod,io4
Channels))

    try:
        # Create IP connection
        ipCon = IPConnection()
        # Create device object
        io4Dev = BrickletIO4V2(io4Uid, ipCon)
        # Connect to brickd
        ipCon.connect(tfHost,tfPort)
        # Check if the bricklet is reachable - else error is triggered
        # In case the bricklet is not reachable, it takes 1-2 seconds befor a response is given
        io4Dev.read_uid()
        # Set the status led
        io4Dev.set_status_led_config(io4StatusLED)
        # Set one or more channels
        if io4Channels != None:
            title = ""
            for item in io4Channels:
                channel = int(item["channel"])
                value = int(item["value"])
                direction = item["direction"]
                # set configuration is used to enable the of getting the value using get_configuration
                io4Dev.set_configuration(channel, direction, value)
                # Register input value callback to function cb_input_value
```

```
                if direction == "i":
                    io4Dev.register_callback(io4Dev.CALLBACK_INPUT_VALUE, on_io4_callback)
                    # Set period for input value for the channel with callback
                    io4Dev.set_input_value_callback_configuration(channel, io4CallbackPeriod, False)
                title = "channel:{} value:{} direction:{}".format(channel,value,direction)
            result = True
            # result = { "status" : "OK", "title" : "{}".format(title) }
        #
        print("Tinkerforge Connected: host={},port={},uid={}".format(tfHost,tfPort,io4Uid))
        # Disconnect
        # ipCon.disconnect()
    except:
        result = False
        # result = { "status" : "ERROR", "title" : "io4 failed. Check master & bricklet." }
    # Print the result which is used by the dzVents Lua script
    if DEBUG == True:
        print(json.dumps(result))

    return result

    # input("Press key to exit\n") # Use raw_input() in Python 2
    # ipCon.disconnect()

# Callback function for the io4 channel(s) configured as input
# Updates the domoticz text device:
# Button pushed (example log entry & device text):
# 2020-02-29 12:09:08.468 MQTT: Topic: domoticz/in, Message:
{"command":"udevice","idx":108,"svalue":"IO4 changed:True, value:False, time:2020-02-29
# TF IO4 Status IO4 changed:True, value:True, time:2020-02-29 12:13:13
def on_io4_callback(channel, changed, value):
    if DEBUG == True:
        print("on_io4_callback:channel:{},changed:{},value:{}".format(channel,changed,value))
    # Select the channel
    if (channel == IO4_CHANNEL_1) and (changed == True):
        svalue = "IO4 changed:{}, value:{}, time:{}".format(changed,value,datetime.now().strftime('%Y-
%m-%d %H:%M:%S'))
        set_domoticz_text_device(IDX_STATUS, svalue)

# Main
if __name__ == "__main__":
    print("{} starting...".format(VERSION))
    create_mqtt_instance()
```

Appendix Domoticz Hints

Start, Stop, Status

Terminal commands

```
// Start, Stop, Status
sudo service domoticz.sh start
sudo service domoticz.sh stop
sudo service domoticz.sh status

// Enable Domoticz service at boot
sudo systemctl daemon-reload
sudo systemctl enable domoticz.service

// Disable Domoticz Service at boot
sudo systemctl disable domoticz.service

// Start, Stop Domoticz service using systemctl
sudo systemctl start domoticz.service
sudo systemctl stop domoticz.service
```

Output Status Sample

```
domoticz.service - LSB: Home Automation System
   Loaded: loaded (/etc/init.d/domoticz.sh)
   Active: active (running) since Mon 2018-08-27 09:32:39 CEST; 16min ago
  Process: 31803 ExecStop=/etc/init.d/domoticz.sh stop (code=exited, status=0/SUCCESS)
  Process: 32093 ExecStart=/etc/init.d/domoticz.sh start (code=exited, status=0/SUCCESS)
   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 a browser run:

http://NNN.NNN.1.60:8080/json.htm?type=devices&rid=28

Result JSON string (snippet)


```
{
   "app_version" : "4.9999",
   "result" : [
      {
         "BatteryLevel" : 0,
         "Data" : "22.00;NNE;5;0;26.5;26.5",
         "HardwareName" : "RFXtrx433e",
         "HardwareType" : "RFXCOM - RFXtrx433 USB 433.92MHz Transceiver",
         "SignalLevel" : 6,
         "Used" : 1,
         "idx" : "28"
      }
   ],
   "status" : "OK",
   "title" : "Devices"
}
```


From the returned result, the BatteryLevel is 0.
It seems that the rechargeable battery is defect for this device.
The battery is recharged by a small solar panel on top of the device.

Check Domoticz Status

Command

```
sudo service domoticz.sh status
```

Output Example
The log shows Domoticz is running OK.

```
domoticz.service - LSB: Home Automation System
   Loaded: loaded (/etc/init.d/domoticz.sh; generated)
   Active: active (running) since Fri 2020-01-31 13:04:15 CET; 3h 7min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 19 (limit: 2077)
   Memory: 43.7M
   CGroup: /system.slice/domoticz.service
           └─503 /home/pi/domoticz/domoticz -daemon -www 8080 -sslwww 443

Jan 31 13:04:13 dodev systemd[1]: Starting LSB: Home Automation System...
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.463  Status: Domoticz V4.11666 (c)2012-2020
GizMoCuz
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.463  Status: Build Hash: 3630122d7, Date:
2020-01-31 08:45:14
Jan 31 13:04:15 dodev domoticz.sh[431]: 2020-01-31 13:04:15.464  Status: Startup Path:
/home/pi/domoticz/
Jan 31 13:04:15 dodev domoticz.sh[431]: domoticz: Domoticz is starting up....
Jan 31 13:04:15 dodev domoticz[457]: Domoticz is starting up....
Jan 31 13:04:15 dodev domoticz[503]: Domoticz running...
Jan 31 13:04:15 dodev systemd[1]: Started LSB: Home Automation System.
```

GLIBC Version not found

After updating a beta version on Raspberry Pi running Raspian Debian Strech, the Domoticz server was not reachable via webbrowser domoticz-ip:port.
A possible cause could be the GLIBC version is not right.

How to find out?

1. Stop Domoticz
2. cd /home/pi/domoticz
3. sudo chmod +x ./domoticz.sh
4. sudo ./domoticz.sh start
5. If a message like below occurs then Domoticz requires newer GLIBC version.

```
/home/pi/domoticz/domoticz: /lib/arm-linux-gnueabihf/libm.so.6: version `GLIBC_2.27' not found
(required by /home/pi/domoticz/domoticz)
/home/pi/domoticz/domoticz: /lib/arm-linux-gnueabihf/libc.so.6: version `GLIBC_2.28' not found
(required by /home/pi/domoticz/domoticz)
```

How to fix?
As the Raspberry Pi was not running the latest Raspbian Debian Buster version, decided to reinstall the system from scratch.
After installation, verified GLIBC version by running the command:

```
ldd --version | grep -i glibc
```

Output

```
ldd (Debian GLIBC 2.28-10+rpi1) 2.28
```

Note
Installation steps are explained in chapter Setup, but in a nutshell to get going:

1. Create new Raspberry Pi Image Raspbian Debian Linux Buster NOOBS v3.2.1
2. Reboot the Raspberry Pi and followed finalization steps
3. Set a fixed WLAN IP address (this is a MUST)
4. Created folders under /home/pi for later use, i.e. b4j,domoticz, java,python,tinkerforge
5. Installed Domoticz: curl -L install.domoticz.com | sudo bash
6. Installed mosquito: sudo apt-get install mosquitto mosquitto-clients
7. Reboot the Raspberry Pi
8. Access Domoticz GUI from browser

Various

Some useful command line interface (CLI) commands.

Check the Ports in Use

```
sudo netstat -tulpn | grep LISTEN
```

Output Example
The list shows the Domoticz ports 8080 & 443 are in use.

```
tcp        0      0 0.0.0.0:4240            0.0.0.0:*               LISTEN
8175/brickd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
465/sshd
tcp        0      0 0.0.0.0:1880            0.0.0.0:*               LISTEN
345/node-red
tcp        0      0 0.0.0.0:1883            0.0.0.0:*               LISTEN
428/mosquitto
tcp        0      0 0.0.0.0:4223            0.0.0.0:*               LISTEN
8175/brickd
tcp6       0      0 :::8080                 :::*                    LISTEN
503/domoticz
tcp6       0      0 :::22                   :::*                    LISTEN
465/sshd
tcp6       0      0 :::443                  :::*                    LISTEN
503/domoticz
tcp6       0      0 :::1883                 :::*                    LISTEN
428/mosquitto
tcp6       0      0 :::6144                 :::*                    LISTEN
503/domoticz
```

Remove ambitious CR characters from bash scripts.

To avoid the message:

```
/bin/bash^M: bad interpreter: No such file or directory…
```

The ^M has probably been created by editing the bash script under Wondows. Linux does not like ^M. It can not find /bin/bash^M as it expects /bin/bash.

Remove>

```
sed -i -e 's/\r$//' *.sh
```

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: Initalizing event module.
  2018-09-12 16:18:13.481  Status: EventSystem: Started
  2018-09-12 16:18:13.481  Status: EventSystem: Queue thread started...
  2018-09-12 16:18:42.447  Status: Incoming connection from: NNN.NNN.1.3

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

Manual update:
Stable: cd domoticz, ./updaterelease
Beta: cd domoticz, ./updatebeta
Source Update: cd domoticz, git pull, make -j 4 OR make
```

Appendix volkszaehler Setup

Description of the **volkszaehler** with an **IR device**, connected to a Raspberry Pi.
volkszaehler.org 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 domoticz-ip
netmask 255.255.255.0
broadcast NNN.NNN.0.255
gateway domoticz-ip
```

Edit sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
          ssid="YOURSSID"
          proto=RSN
          key_mgmt=WPA-PSK
          pairwise=CCMP
          group=CCMP
          psk="YOURPSK"
  }
```

Option 2 Ethernet static IP address

The Raspberry Pi volkszahler and the Raspberry Pi Domoticz production system are direct connected via an Ethernet cable. A standard cable is used, no twisted wires required.
The communication uses ETH0 with static network addresses on both sides.
Important: The ETH0 network address must be different from the WLAN network address.

Raspberry Pi volkszahler **[Raspbian GNU/Linux 8 (jessie)]** **Linux 39 4.1.19-v7+ #858**	**Raspberry Pi Domoticz production** **[Raspbian GNU/Linux 8 (jessie)]** **Linux openHABianPi 4.9.35-v7+ #1014**
eth0 inet addr:169.254.87.85 Bcast:169.254.255.255 Mask:255.255.0.0	eth0 inet addr:169.254.87.84 Bcast:169.254.255.255 Mask:255.255.0.0
Set the static Ethernet network Address sudo nano /etc/dhcpcd.conf	sudo nano /etc/dhcpcd.conf
Add the lines	
interface eth0 static ip_address=169.254.87.85	interface eth0 static ip_address=169.254.87.84

static routers=169.254.0.1	static routers=169.254.0.1

Note

- Determine the Linux version: uname –a to
- Determine the Raspberry Pi release: cat /etc/os-release to

Used option direct communication with ethernet cable.

volkszaehler

PowerMeter Setup

Powermeter: EMH ED300L
The powerpuls is captured using an IR-write-reader device.
This device is connected via USB to the Raspberry Pi 2 Port /dev/ttyUSB22.

Note

- Check the connected USB devices with command **ls /dev/tty***
- Ensure when changing the Raspberry Pi or adding USB devices, to use the same port for the IR-write-reader device (read).

Persistent USB devices are defined using:

```
sudo nano /etc/udev/rules.d/99-usb-serial.rules
```

Add Line

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="00F2E620",
SYMLINK+="ttyUSB-VZ", MODE="0666"
```

Check USB device information

```
sudo lsusb -v
```

To check detailed USB port information, use the command:

```
udevadm info --query=all --name=/dev/ttyUSB-VZ
```

Output Example

```
P: /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
N: ttyUSB0
S: serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0
S: serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0
S: ttyUSB-VZ
E: DEVLINKS=/dev/serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0 /dev/serial/by-id/usb-
Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0 /dev/ttyUSB-VZ
E: DEVNAME=/dev/ttyUSB0
E: DEVPATH=/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
E: ID_BUS=usb
E: ID_MODEL=CP2104_USB_to_UART_Bridge_Controller
E: ID_MODEL_ENC=CP2104\x20USB\x20to\x20UART\x20Bridge\x20Controller
E: ID_MODEL_FROM_DATABASE=CP210x UART Bridge / myAVR mySmartUSB light
E: ID_MODEL_ID=ea60
E: ID_PATH=platform-3f980000.usb-usb-0:1.5:1.0
E: ID_PATH_TAG=platform-3f980000_usb-usb-0_1_5_1_0
E: ID_REVISION=0100
E: ID_SERIAL=Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620
E: ID_SERIAL_SHORT=00F2E620
E: ID_TYPE=generic
E: ID_USB_DRIVER=cp210x
E: ID_USB_INTERFACES=:ff0000:
E: ID_USB_INTERFACE_NUM=00
E: ID_VENDOR=Silicon_Labs
E: ID_VENDOR_ENC=Silicon\x20Labs
E: ID_VENDOR_FROM_DATABASE=Cygnal Integrated Products, Inc.
E: ID_VENDOR_ID=10c4
E: MAJOR=188
E: MINOR=0
E: SUBSYSTEM=tty
E: TAGS=:systemd:
E: USEC_INITIALIZED=5005222
```

USB Configuration

The USB port settings are 9600 Baud 8 n.

Set via command stty:

```
stty -F /dev/ttyUSB1
1:0:8bd:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
```

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)

```
stty -F /dev/ttyUSB-VZ
1:0:8bd:0:3:1c:7f:15:4:5:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
```

Check Data coming from the USB device (option 1):

```
cat /dev/ttyUSB-VZ | od -tx1
```

Output Example:

```
 0000000 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 a browser the volkszaehler Frontend using its ip address: http://vz-ip

Define Channels using the volkszaehler FrontEnd

```
Channel 1 =
{"version":"0.3","entity":{"uuid":"d14e9a80-8894-11e5-9dc8-
298a9a1001ac","type":"current","active":true,"color":"aqua","description":"Strom
Direktverbrauch","fillstyle":0,"public":true,"resolution":10000,"style":"steps","title":"Direktverbrauc
h","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":"Gesamtverbrauc
h","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","titl
e":"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":"Wirkleisting","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 Leisting"** 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_finest)
"log": "/var/log/vzlogger.log",   // log file, optional
"retry": 30,                      // http retry delay in seconds

// Build-in HTTP server
"local": {
  "enabled": false, // enable local HTTPd for serving live readings
  "port": 8080,   // TCP port for local HTTPd
  "index": true,  // provide index listing of available channels if no UUID was requested
  "timeout": 30,  // timeout for long polling comet requests in seconds (0 disables comet)
  "buffer": 600   // HTTPd buffer configuration for serving readings, default -1
                 //   >0: number of seconds of readings to serve
                         //   <0: number of tuples to server per channel (e.g. -3 will serve 3 tuples)
  },

"push": [],
"meters" : [
  {
    "enabled" : true,
    "protocol" : "sml",
    "device" : "/dev/ttyUSB-VZ",
    "baudrate": 9600,
    "parity": "8n1",
    "channels":
    [
      {
        // Leistungswert
        "uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4",
        "middleware" : "http://localhost/middleware.php",
        "identifier" : "1-0:16.7.0"
      },
      {
        // Zaehlerstand
        "uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041",
        "middleware" : "http://localhost/middleware.php",
        "identifier" : "1-0:1.8.0"
      }
    ]
  }]
}
```

volkszaehler Frontend http://vz-ip showing the two channels defined.

Clicking on the Aktion (i) icon provides the details for the two channels defined:

API Channel Properties

Energie Leistung (Channel 1)

```
http://vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```


```
{"version":"0.3","entity":{"uuid":"958bce60-b342-11e8-b54f-
bbec0573e1f4","type":"powersensor","color":"#311b92","fillstyle":0,"public":true,"style":"lines","title
":"Energie Leistung","yaxis":"auto"}}
```


Energie Zaehlerstand (Channel 2)

```
http://vz-ip/middleware.php/channel/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json
```


```
{"version":"0.3","entity":{"uuid":"5b8ea2a0-b342-11e8-bec6-15c040e6d041","type":"electric
meter","color":"aqua","fillstyle":0,"public":true,"resolution":1000,"style":"steps","title":"Energie
Z\u00e4hlerstand","yaxis":"auto"}}
```

vzlogger Service

Ensure the vzlogger service is correct 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 service vzlogger start
vzlogger Stop	systemctl stop vzlogger OR service vzlogger stop
vzlogger Status	systemctl status vzlogger Example output:
vzlogger Daemon reload	systemctl daemon-reload
vzlogger Debug	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://vz-ip/middleware.php/entity/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

Result


```
{"version":"0.3","entity":{"uuid":"958bce60-b342-11e8-b54f-
bbec0573e1f4","type":"powersensor","color":"#311b92","fillstyle":0,"public":true,"style":"lines","title
":"Energie Leistung","yaxis":"auto"}}
```


API Channel Properties

```
http://vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

Result


```
{"version":"0.3","entity":{"uuid":"958bce60-b342-11e8-b54f-
bbec0573e1f4","type":"powersensor","color":"#311b92","fillstyle":0,"public":true,"style":"lines","title
":"Energie Leistung","yaxis":"auto"}}
```


API Entities

```
http://vz-ip/middleware.php/capabilities/definitions/entities.json
```

API Read Data Requests

Note

Relative Dateformats = http://de.php.net/manual/en/datetime.formats.relative.php

Get All Data

```
http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json
```

Get One Record

```
http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?tuples=1
```

Result


```
{"version":"0.3","data":{"tuples":[[1447316802613,246.491,19046]],"uuid":"23844c00-8895-11e5-a642-
6d8ce36c9a44","from":1447262000365,"to":1447316802613,"min":[1447316802613,246.49134831111],"max":[1447
316802613,246.49134831111],"average":246.491,"consumption":3752.3,"rows":1}}
```


Get the last value

```
http://vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?from=now&to=now
```

Result


```
{"version":"0.3","data":{"tuples":[[1447317119885,201.681,1],[1447317121700,198.347,1]],"uuid":"23844c0
0-8895-11e5-a642-
6d8ce36c9a44","from":1447317118100,"to":1447317121700,"min":[1447317121700,198.34710743513],"max":[1447
317119885,201.68067227331],"average":200,"consumption":0.2,"rows":3}}
```

API Logging

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

Using the volkzaehler Frontend to display data
http://vz-ip/frontend/?uuid=23844c00-8895-11e5-a642-6d8ce36c9a44

Appendix Tools

In progress developing some tools for managing Domoticz.

Domoticz Internal Script Viewer

Purpose

- To view offline the Domoticz internal scripts (focus on dzVents scripts) stored in the Domoticz database (domoticz.db), which are created using the Domoticz GUI Event editor (Setup > More Options > Events – My automation scripts)
- To export the scripts to the clipboard, text file or HTML file

Download the application from GitHub.

Screenshot

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”.

The MQTT logger

- logs (in reverse order) the last 5 MQTT message payloads from the Domoticz Production oe Development server.
- displays in a separate group the last entry Idx, Name and Payload.
- enables to analyze MQTT message in detail:
  - check the values from properties nvalue, svalue, svalue1 etc.
  - explore the properties of a new device, to use in scripts or apps.

More information about MQTT.

Solution

The solution is based on a Node-RED flow, subscribing to Domoticz MQTT messages published by topic “domoticz/out” using the Domoticz Hardware “MQTT Client Gateway with LAN interface” (see GUI > Settings > Hardware).

Node-RED Dashboard UI

The dashboard has its own tab MQTT Logger with the three groups Log Entries, Last Entry and Settings.

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

Notes
It is easy to build flows in Node-RED for these kind of utility functions.
Scribbled first the concept on paper, some trial-and-error flows on payload handling and lastly use the Dashboard UI nodes to build the UI.

Description

Group Log Entries

Display the last 5 messages logged - rolling entries (the number of displayed messages is defined in the function node “Add msg to flow.log”).

The flow uses data stored as flow context for the

- log = an array for the messages displayed (flow.log)
- idx filter content = comma separated string to filter device idx (flow.idxfilter)
- use the idxfilter = flag 0 or 1 (flow.usefilter)

Each entry has a timestamp - taken from the msg.payload property msgreceived which is added to the Domoticz payload after receiving the message.
The msg.payload is converted to a string to enable word wrap - after a comma a space is added.

Hold Logging
Hold the logging - useful if want to view in detail or copy the listed messages.

Clear Log
Clear the listed log entries.

Group Last Entry
Display the Idx, Name and Payload of the last message received.
The msg.payload is converted to a string to enable word wrap - after a comma a space is added.

Button Copy Message
Copy the content of the message field to the clipboard. Example content:


```
{"Battery":100,"RSSI":5,"description":"","dtype":"Temp + Humidity","id":"10254","idx":40,"name":"WZ
Temperatur","nvalue":0,"stype":"Cresta, TFA
TS34C","svalue1":"21.6","svalue2":"43","svalue3":"1","unit":1,"msgreceived":"17:56:24"}
```


Group Settings
Broker
Select the MQTT broker. Two options: Domoticz Production (DoPro) or Domoticz Development (DoDev) broker.

Idx Filter
Set a filter containing 1 or more devices idx. The input must end with a comma.

Use Filter
Use the filter defined. If not switched on, then all messages are listed.

Node-RED Flow

Source: mqtt-logger.flow

Node-RED Flow Brief Description

The flow makes use of the Node-RED Dashboard Module.

The mqtt-in node is the entry point from which the message payload is routed to Node-RED Dashboard nodes. The MQTT broker used is the Domoticz Production server.
And subscribes to the topic " domoticz/out".
The function "Add Msg to flow.log", which is the core of this solution, manages the new MQTT incoming message entries by creating an array with max. 5 msg.payload entries. The array is used by the ui_template node "MQTT Log" to display in reverse order the messages – using form. The function also sends as a seconds output, the last message which is displayed in a form by the ui_template "Last Entry". Both templates uses javaScript scope function to create the field content.

The setting options are setting flow context which are used by the previous described core function.

If want to explore more in detail, lookup at the source of the flow or import the flow.

Alternative Logger

An alternative, rather simple Nod-RED flow, to see Domoticz MQTT messages in JSON format, is to create a flow with three nodes:

Mqtt in	Json	Debug
Listen to topic “domoticz/out” from the Domoticz system	Convert the message payload to a Javascript object	Log the full message object (i.e. msg.payload) or selective property (i.e. msg.payload.idx or msg.payload.svalue1)

Output Debug Node (as full message payload)
(see Browser > Node-RED development tab)

Example Log Entry


```
{
"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"00070006","idx":33,"name":"IAQM
Pressure","nvalue":0,"stype":"Barometer","svalue1":"1011","svalue2":"0","unit":6
}
```

RaspberryMatic Statelist

Purpose

To obtain the list of devices and datapoints , with selected properties, from the RaspberryMatic system.
The focus is on getting the id of the devices and datapoints configured in the CCU.
The id's are used by RaspberryMatic scripts and Domoticz Automation Events (dzVents).

Solution

The RaspberryMatic addon XML-API is used to get the data.
Send a HTTP XML-API request to the Raspberrymatic system, with the statelist script (statelist.cgi) as parameter.
Example HTTP API Request:

```
http://CCU-IP/config/xmlapi/statelist.cgi
```

The HTTP response is a XML tree (as plain text) with all devices and datapoints.
The XML tree is parsed to get the device name(s) & id plus the according datapoint(s) id, type and value.

Node-RED

Versions: Node-RED v1.0.4 with the add-on node-red-dasboard 2.19.4.

Dashboard UI and Flow

Source: raspmatic_statelist.flow

In Brief
The dashboard ui has a device dropdown list which populates a html list with selected device datapoints. The button refresh obtains the latest statelist. The input field CCU IP sets the ip address of the CCU (running on the RaspberryMatic system).

The http request node sends an HTTP XML-API request to the CCU to obtain the statelist in XML format.
The XML tree is converted to a JavaScript object which is parsed to arrays holding JSON formated entries for the devices and datapoints.
These arrays are used for the UI.
The UI is based on the Node-RED Dashboard [Info](https://github.com/node-red/node-red-dashboard).
The devices are added, from the devices array, to a ui_dropdown node.
Selecting a device, lists the assigned datapoints, from the datapoints array, in a ui_template node as list.
A button Refresh requests a new statelist from the CCU.

HTTP XML-API Response

The response contains the statelist with devices, channels and datapoints.
Example with data extract.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stateList>
<device config_pending="false" unreach="false" ise_id="2530" name="Briefkasten Status">
  <channel ise_id="2531" name="Briefkasten Status:0" operate="true" visible="true" index="0">
    <datapoint ise_id="2532" name="HmIP-RF.0000DA498D5859:0.CONFIG_PENDING" operations="5"
timestamp="1583826706" valueunit="" valuetype="2" value="false" type="CONFIG_PENDING"/>
    <datapoint ise_id="2536" name="HmIP-RF.0000DA498D5859:0.DUTY_CYCLE" operations="5"
timestamp="1583826706" valueunit="" valuetype="2" value="false" type="DUTY_CYCLE"/>
    <datapoint ise_id="2537" name="HmIP-RF.0000DA498D5859:0.ERROR_CODE" operations="5"
timestamp="1583826706" valueunit="" valuetype="8" value="0" type="ERROR_CODE"/>
    <datapoint ise_id="2539" name="HmIP-RF.0000DA498D5859:0.LOW_BAT" operations="5"
timestamp="1583826706" valueunit="" valuetype="2" value="false" type="LOW_BAT"/>
    <datapoint ise_id="2543" name="HmIP-RF.0000DA498D5859:0.OPERATING_VOLTAGE" operations="5"
timestamp="1583826706" valueunit="" valuetype="4" value="1.200000" type="OPERATING_VOLTAGE"/>
    <datapoint ise_id="2544" name="HmIP-RF.0000DA498D5859:0.OPERATING_VOLTAGE_STATUS" operations="5"
timestamp="1583826706" valueunit="" valuetype="16" value="0" type="OPERATING_VOLTAGE_STATUS"/>
    <datapoint ise_id="2545" name="HmIP-RF.0000DA498D5859:0.RSSI_DEVICE" operations="5"
timestamp="1583826706" valueunit="" valuetype="8" value="180" type="RSSI_DEVICE"/>
    <datapoint ise_id="2546" name="HmIP-RF.0000DA498D5859:0.RSSI_PEER" operations="5" timestamp="0"
valueunit="" valuetype="8" value="0" type="RSSI_PEER"/>
    <datapoint ise_id="2547" name="HmIP-RF.0000DA498D5859:0.SABOTAGE" operations="5"
timestamp="1583826706" valueunit="" valuetype="2" value="false" type="SABOTAGE"/>
    <datapoint ise_id="2551" name="HmIP-RF.0000DA498D5859:0.UNREACH" operations="5"
timestamp="1583826706" valueunit="" valuetype="2" value="false" type="UNREACH"/>
    <datapoint ise_id="2555" name="HmIP-RF.0000DA498D5859:0.UPDATE_PENDING" operations="5"
timestamp="1582992737" valueunit="" valuetype="2" value="false" type="UPDATE_PENDING"/>
  </channel>
  ... more channels
</device>
<device config_pending="false" unreach="true" ise_id="3597" name="Eingang Monitor">
  ... channels & datapoints
</device>
... more devices
</stateList>
```

Function Node Get Devices and Datapoints

The purpose of this function is to

- return a json array with devices used for the ui_dropdown
- set flow context "datapoints" for the ui_template to display the datapoints for the selected device id

Notes

- Example device entry: deviceobj["Briefkasten Status"]=2530;
- Example datapoint entry: device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-RF.0000DA498D5859:0.CONFIG_PENDING"}
- the second array below is not used as replaced by the flow context "datapoints" - but is kept in case any enhancements planned

```
statelist = msg.payload.stateList;
devices = statelist.device
arrdevices = [];
arrdatapoints = [];
devices.forEach(function(device) {
    devicename = device.$.name;
    deviceise_id = device.$.ise_id;
    deviceobj = {}
    deviceobj[devicename]=deviceise_id;
    arrdevices.push(deviceobj);
    channels = device.channel;
    channels.forEach(function(channel){
        channeldatapoints = channel.datapoint;
        if (channeldatapoints !== undefined) {
            channeldatapoints.forEach(function(datapoint){
                obj = datapoint.$;
                if (obj !== undefined) {
                  dpobj = {
                    "device":deviceise_id,
                    "datapoint": {
                      "id":obj.ise_id,"type":obj.type,"value":obj.value,"name":obj.name
                    }
                  };
                arrdatapoints.push(dpobj);
                }
            });
        }
    });
});
flow.set("datapoints",arrdatapoints);

msgdevices = {};
msgdevices.options = arrdevices;

// not used
msgdatapoints = {};
msgdatapoints.payload = arrdatapoints;

return [msgdevices,msgdatapoints];
```

Function Node Select Device Data

The purpose of this function is to

- get all datapoints for the selected device using the device id (ise_id)
- return an array with datapoint entries: id: type=value

Notes

- The type is the short name of the datapoint
- The array is used for the ui_template html list

```
const DEBUG = false;
// get the device id, i.e. 2530 for the selected device
var deviceid = msg.payload;
if (DEBUG) node.warn("Selected: " + deviceid);

if (deviceid === null) return [];

// get all the datapoints by device
// device:"2530",datapoint:{"id":"2532","type":"CONFIG_PENDING","value":"false","name":"HmIP-
RF.0000DA498D5859:0.CONFIG_PENDING"}
datapoints = flow.get("datapoints");

devicedatapoints = [];
datapoints.forEach(function(datapoint){
    if (datapoint.device == deviceid) {
        dp = datapoint.datapoint;
        var entry = dp.id + ": " + dp.type + "=" + dp.value;
        devicedatapoints.push(entry);
        if (DEBUG) node.warn("Added: " +  entry);
    }
});

msg.payload = devicedatapoints;
return msg;
```

Last page initially left blank