

ROBERT W.B. LINN

Domoticz Home Automation Workbook

Exploring Domoticz Home Automation System

by Robert Linn, Pinneberg, Germany

13.04.2019

DISCLAIMER

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

Table of Contents

Table of Contents	1
Purposes	8
Components.....	10
Hardware	10
Software	11
External Services.....	11
Setup.....	12
Development Device	12
Raspberry Pi	12
Installation	12
WLAN Static IP	13
WLAN Power Save Mode.....	14
Raspian Check Version & Update	15
Samba	16
Persistent USB Devices.....	18
Domoticz	20
Installation	20
Settings	21
Folders.....	22
Events	22
Functions	23
Introduction	23
RFXCOM RFXtrx433E	24
Purpose	24
Prepare RFXtrx433E	24
Domoticz Configuration.....	26
Temperature & Humidity	27
Purpose	27
Device.....	27
Setup.....	27
Anemometer	28
Purpose	28
Device.....	28
Setup.....	28
Wind	29

Purpose	29
Solution	29
Event dzVents	30
Somfy Blinds	31
Purpose	31
Domoticz Configuration.....	31
Dashboard & Switches Widget	32
MQTT Messages	32
Air Pressure	33
Purpose	33
Circuit.....	33
I2C	34
Hardware & Device.....	35
Barometer Widget	36
Barometer Event (dzVents).....	36
Philips Hue	38
Purpose	38
Hue Bridge Configuration.....	38
Hue Devices	38
Hue Light Add New	39
Hue Control Tests.....	40
Hue Timed Switch Lights	42
Hue Lights Control Node-RED.....	44
Remote Switch	46
Purpose	46
Device.....	46
Setup.....	46
Waste Calendar	48
Purpose	48
Devices Configuration.....	49
Concept.....	49
CSV Input File	49
dzVents Lua Script	51
Python CSV File Generation Script	53
Days-To-Go.....	57
Purpose	57
Solution	57

Event Properties	57
Event Script dzVents	58
Raspberry Pi Monitoring.....	60
Purpose	60
Solution	60
Monitoring.....	61
Electric Usage	63
Purpose	63
Solution	64
dzEvents Script.....	65
volkszaehler.....	66
Coffee Machine Monitor.....	82
Purpose	82
Solution	82
dzVents Script	82
Alert Message.....	84
Purpose	84
Solution	84
dzEvents Script.....	85
Alert Message Reset.....	86
Database Tables.....	88
Info Message (Not Used)	89
Purpose	89
Solution	89
dzEvents Script.....	89
Web Frontend Customized.....	90
Purpose	90
Web Front End Sample	90
Concept.....	90
HTML & JavaScript.....	92
App atHome	95
Purpose	95
Solutions.....	95
Volumio Webradio	97
Purpose	97
Volumio Setup.....	98
Solution Options.....	98

Solution dzVents	100
Solution Node-RED	104
Favorites List.....	109
Stock Monitor.....	110
Purpose	110
Solution	111
Domoticz Configuration.....	112
Node-RED Flow.....	113
Notifications	115
Node-RED Dashboard UI	117
Enhancements.....	118
Ambient Light	119
Purpose	119
I2C	119
Parts Needed.....	119
Wiring.....	119
Domoticz Device Setup.....	119
ESP Easy Device Setup.....	120
Trigger Switch Light	121
User Variables	123
Syntax.....	123
User Variable List	123
Usage in Scripts.....	124
Usage in Browser Interactive	125
Usage in Node-RED	126
Usage in Node-RED Dashboard	127
Ambient Light Threshold	127
Custom Icons.....	129
Explore	130
Events System	130
Purpose	130
Event Execution Order	130
Event Scheduling & Trigger	131
Event Database Tables	131
Event Script Viewer	131
Event Development	132
Event Sample Basement Humidity Monitor	133

dzVents.....	143
Device Properties	143
Script	143
Property List.....	144
External Modules	144
Shared Helper Functions	148
API Interaction	149
Purpose	149
Option HTTP.....	149
Option MQTT	150
Recommendation.....	151
MQTT.....	152
Purpose	152
Install	152
Mosquitto	154
Python.....	156
SQL	162
Purpose	162
SQL Commands	162
Tables	162
Select	164
Delete	167
Node-RED	171
Purpose	171
Installation.....	171
Update	172
Access Flows & Dashboard UI.....	174
Start, Stop, Log.....	174
Manage Node Packages	175
Folder Locations.....	175
Flow Domoticz MQTT Messages	176
Flow Raspberry Pi CPU Usage	177
Node-RED as MQTT Publisher	178
Dashboard UI	178
Python Plug In	183
Purpose	183
ESPEasy	184

Purpose	184
Experiments.....	184
Flash NodeMCU	184
Update NodeMCU.....	185
Domoticz Controller.....	186
Rules.....	187
GPIO Commands	188
ESP8266	189
Purpose	189
Experiment	189
Prototype	189
Prepare Arduino IDE.....	189
Parts Needed.....	190
Wiring.....	190
Circuit.....	190
Source Code	191
Appendix Domoticz Hints.....	192
Start, Stop, Status	192
Lost Username and Password	192
Create Backup	193
Troubleshooting	194
Device Widget turns yellow/red.....	194
Appendix Domoticz Build Source	195
Appendix volkszaehler Setup.....	197
Hardware	197
Software	197
Setup Raspberry Pi	198
Option 1 WLAN static IP address	198
Option 2 Ethernet static IP address	198
volkszaehler.....	200
PowerMeter Setup	200
USB Configuration.....	201
Define Channels	202
vzlogger Setup	203
Appendix Tools	210
Domoticz Internal Script Viewer	210
Purpose	210

Appendix TODO	210
---------------------	-----

Purposes

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

As a Domoticz Beginner

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

Notes

- ❖ There might be better solutions for what is shared – solutions might change whilst evolving.
- ❖ This is a working document – idea's & todo's will never cease to exist.
- ❖ The code of scripts & flows is not fully shared or might have changed – check the GitHub scripts folder for the latest full source.
- ❖ The scripts are mainly developed using dzVents Lua scripts. Few exceptions using Python.
- ❖ There is quite a bit of functionality where Node-RED flows are used. Node-RED is one of my favorite tools – esp. as wanting to learn JavaScript.

Functions

- Display temperature & humidity measured in rooms.
- Charts for selective weather items, room temperature & humidity.
- Control Somfy roller shutters with RTS motors in rooms.
- Philips Hue Lighting System control via Hue Bridge for ZigBee devices.
- Security door & window wireless contact detectors.
- Information on key dates (Calendar type information).
- MQTT subscribe & publish messages to trigger actions or information.
- Raspberry Pi system information with charts and threshold email notification.
- Energy Power Consumption metering from “volkszaehler” with charts.
- Control Volumio Music Player whilst listening to Web Radio.
- Monitor stock quotes.
- Ambient Light (from an ESP8266 NodeMCU running ESP Easy) with threshold.
- Custom icons.

Explore How to Use

- Domoticz running on a Raspberry Pi (setup, configure).
- Scripting Python, Lua, dzVents, JavaScript.
- RFXCOM RFXtrx433E USB RF Transceiver for
 - Temperature & Humidity devices.
 - External Wind device (only for RFXCOM tests).
 - Other 433Mhz devices, i.e. door & window contacts.
- External services, i.e. OpenWeatherMap.
- MQTT messaging.
- Philips Hue Light Control.
- Node-RED as an alternative script engine and User Interface.
- ESP Easy for sensors.

- Domoticz Android App (native client, to be determined).
- Advanced User Interfaces, i.e. Node-RED, Bootstrap ...

Credits

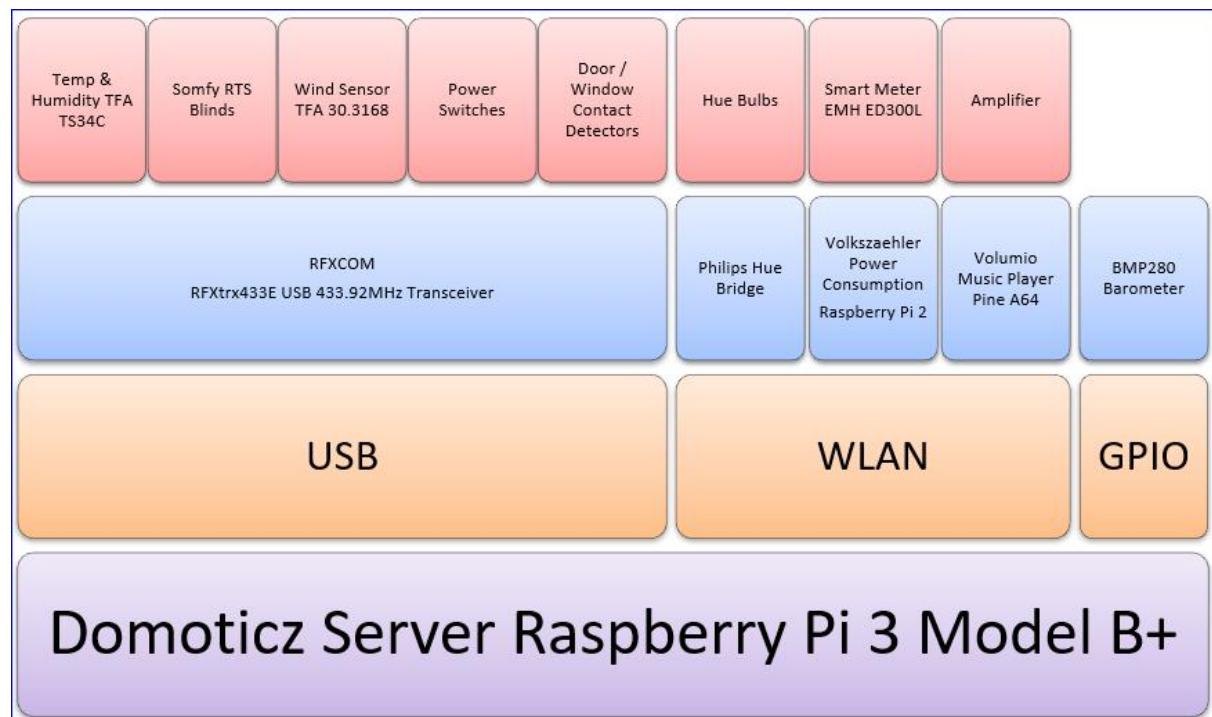
To the developers of Domoticz and to all sharing information about Domoticz. Without these, it would not be possible to build this project and write the workbook.

Components

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

Hardware

- Raspberry Pi 3 Model B+ v1.2
- RFXCOM RFXtrx433E USB 433.92MHz Transceiver
- TFA Dostmann TS34C Temperature & Humidity device
- TFA Dostmann 30.3168 Wind Meter (speed, direction, temperature)
- Somfy Blinds RTS Pure
- PB-62R Door & window wireless contact detector
- Philips Hue Bridge
- volkszaehler Raspberry Pi 2 with SmartMeter EMH ED300L
- Volumio Music Player Pine A64
- Power Switches
- BPM280 Barometer



Software

- Raspbian GNU/Linux 9 (Stretch) based upon NOOBS Version 2.8.2, 2018-06-27
- Domoticz V4.1 BETA
 - BETA version to ensure include latest fixes (i.e. Philips Hue)
- 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
- WinSCP - to exchange files between the Development Device and the Raspberry Pi's
- PuTTY - to run terminal commands
- Thonny - Python Script development
- The software versions mentioned have been used during the initial setup of the project. Recommend check out for newer versions available.

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
- [Thonny](#) - Python IDE for script development
- [Notepad++](#) - Text and source code editor

Raspberry Pi

Installation

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

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

NOOBS

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

NOOBS Version 2.8.2, 2018-06-27, Zip archive NOOBS_v2_8_2.zip.

On the Development PC

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

On the Raspberry Pi

- insert the SD card (ensure no power connected)
- connect power to the Raspberry Pi
- from the initial installation menu
 - select Raspbian
 - 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.

WLAN Static IP

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

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

Wireless & Wired Network Settings.

```
Select Configure > Interface > wlan0:  
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.N.NNN
```

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 and check the WLAN address

```
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  
...
```

Note

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

WLAN Power Save Mode

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

Check the Raspberry Pi WLAN power save mode

```
iw wlan0 get power_save
```

Output

```
Power save: on
```

Note

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

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

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

Note

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

Check again the Power Save Mode:

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

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

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

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

Reboot the Raspberry Pi and check power save mode

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

Raspian Check Version & Update

Raspian Check version installed.

```
cat /etc/os-release
PRETTY_NAME=" "
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

Raspian Update to the latest version.

Perform regular updates (doing this once a week):

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

Or run alternative from a Terminal:

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

Samba

Start by reading [here](#).

Hints: Do not forget to set the password and restart Samba.

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

Domoticz Home Folder as a share

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

Prework Raspberry Pi Domoticz Server

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

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

Content

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

Details Samba settings:

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

After configuration: Perform following steps after the Samba configuration.

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

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

Restart samba:

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

Restart domoticz:

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

Prework Windows

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

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

To connect from the Windows explorer:

Network drives:

Add new drive with url <\\rpi-domoticz-ip\DoProDomoticz>

Username: pi, Password: *****

Example Music Shared Folder

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

Create the folder music:

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

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

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

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

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

Restart samba:

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

Restart domoticz:

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

Connect from Windows explorer using
URL: <\\rpi-domoticz-ip\MyMusic>
Username: Pi
Password: *****

Persistent USB Devices

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

USB Symlink for RFXCOM RFXtrx344E

Create symlink:

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

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

Get USB device information

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

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

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

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

Create the symlink

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

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

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

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

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

Domoticz

Installation

The Domoticz installation is initiated by running terminal command:

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

Answers to the questions:

Services:

http, Port: 8080.

Domoticz Folder:

/home/pi/domoticz

Domoticz URL:

<http://rpi-domoticz-ip:8080> or

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

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

Now READY to configure the Domoticz Home Automation Solution further.

Settings

Changes made to the default Domoticz settings:

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

Folders

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

Events

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

The Domoticz internal event editor is used.

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

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

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

Functions

Introduction

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

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

Function examples are

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

Just a few functions to mention:

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

RFXCOM RFxTrx433E

Purpose

The [RFXCOM RFxTrx433E](#) Transceiver is a 433Mhz transmitter and receiver (transceiver) to control or respond to 433Mhz devices (sensors).
More Product information [here](#) (including successor products).

Prepare RFxTrx433E

Install the utility programs RFXflash and RFXmngr on the Development Device.

Note

For upgrading the firmware, the RFX Flash Programmer is required. Always download the latest version prior flashing.

Flash latest firmware (RFxTrx433_Ext2_Firmware) as described in [RFxTrx User Guide](#).

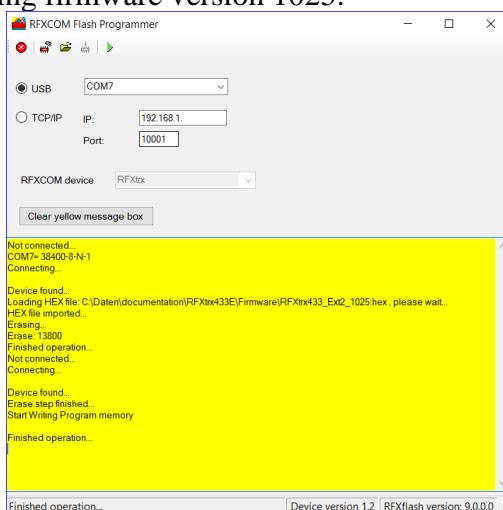
Note

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

Steps to install/upgrade

- Shutdown the Domoticz server (i.e. Raspberry Pi)
- Connect the RFXCOM device to USB port of development device (used COM7 on a notebook)
- Download and unzip the latest firmware
- Download and start the RFXCOM Flash Programmer
- Select the latest hex file
- Write to the device
- Connect the RFXCOM device to the Domoticz server
- Reboot the Domoticz server
- Open Domoticz in web browser and check the log (Setup > Log) and if the hardware device (Setup > Hardware) RFxTrx433e shows the correct version (in this case Version: Ext2/1025)
- In case issues, restart Domoticz from terminal:
sudo service domoticz.sh restart

Example screenshot flashing firmware version 1025.



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

Example RFXmngr Log

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

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

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

Note

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

Domoticz Configuration

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

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

Note

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

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

After adding the device, restart Domoticz (Settings > More Options > Restart) and check the Domoticz Log (Settings > Log), i.e.

```
2018-08-30 11:27:51.659 Status: RFXCOM: Using serial port: /dev/ttyUSB-RFX433E-A
2018-08-30 11:27:52.311 subtype = Interface Response
2018-08-30 11:27:52.311 Sequence nbr = 2
2018-08-30 11:27:52.311 response on 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/Cartelecronic 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)
```

Temperature & Humidity

Purpose

To measure the temperature (°C) & humidity (%RH) in various rooms.

Device

Device (433MHz): Temperature TFA Dostmann / Wertheim TS34C.

The devices are connected to the RFXCOM RFXtrx433E.

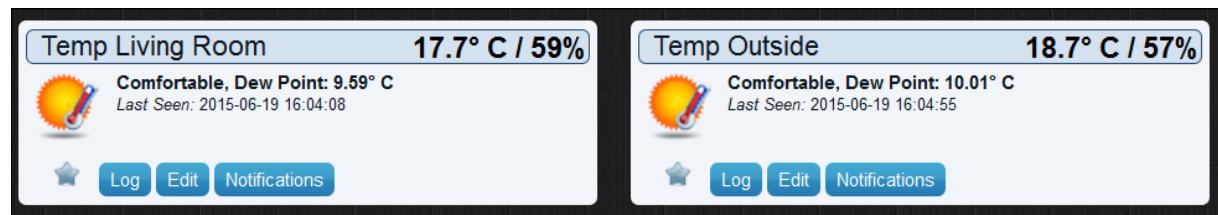
Setup

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

Device List Entries (idx=14,11)

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	Tail	Last Seen
14	RFXCom	450E	2	Temp Outside	Temp + Humidity	Cresta, TFA TS34C	19.0 C, 55 %	4 100	2015-06-19 15:55:10
11	RFXCom	2A0E	1	Temp Living Room	Temp + Humidity	Cresta, TFA TS34C	17.6 C, 59 %	6 100	2015-06-19 15:54:06
6	GPIO Port	18		LED GPIO18	Lighting 1	Impuls	Off	7 -	2015-06-19 15:20:16
12	RFXCom	1F1F1F	1	Blind Living Room	RFY	RFY	Stopped	7 -	2015-06-19 15:18:55
13	RFXCom	1E1E1E	2	Blind Bed Room	RFY	RFY	Stopped	7 -	2015-06-19 15:06:34

Temperature & Humidity Widgets



Anemometer

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)



Wind

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

Event dzVents

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 script using the Domoticz Event Editor.

Event Name	Interpreter	Event Type	Event Active
wind_update.lua	dzVents	Device	Enabled
<pre>--[[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. Project: atHome Interpreter: dzVents, Device See: athome.pdf Author: Robert W.B. Linn Version: 20180910]]-- -- 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 } </pre>			
Domoticz Log Entry			
<pre>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</pre>			

Somfy Blinds

Purpose

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

Domoticz Configuration

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

Do NOT press Add Device

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

Note

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

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

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

The screenshot shows the Domoticz interface with two main windows:

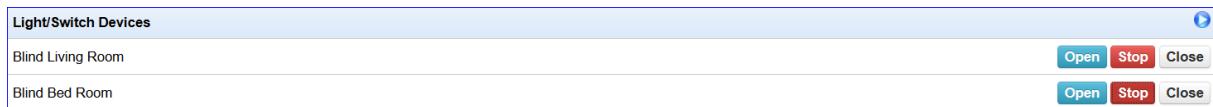
- Add Manual Light/Switch Device:** A configuration dialog where "Hardware" is set to "RFXtrx433e", "Device name" is "Blind Bed Room", "Switch Type" is "Blinds", "Type" is "RFY", and "ID" is "1E 1E 1E". The "Unit Code" dropdown is set to "2". Below the form are buttons for "Test", "As: Main Device" (radio button), and "Sub/Slave Device" (radio button). At the bottom are "Add Device" and "Cancel" buttons.
- Blind Bed Room:** A device card showing the device is "Stopped". It includes a thumbnail of a window with a red "STOP" button, the last seen time (2015-06-19 15:06:34), and type (RFY, RFY, Blinds). Below the card are buttons for "Log", "Edit", "Timers", and "Notifications".

Below these windows is a table of devices:

Idx	Hardware	ID	Unit	Name	Type	SubType	Data	U	T	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

Dashboard & Switches Widget

Dashboard



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

Air Pressure

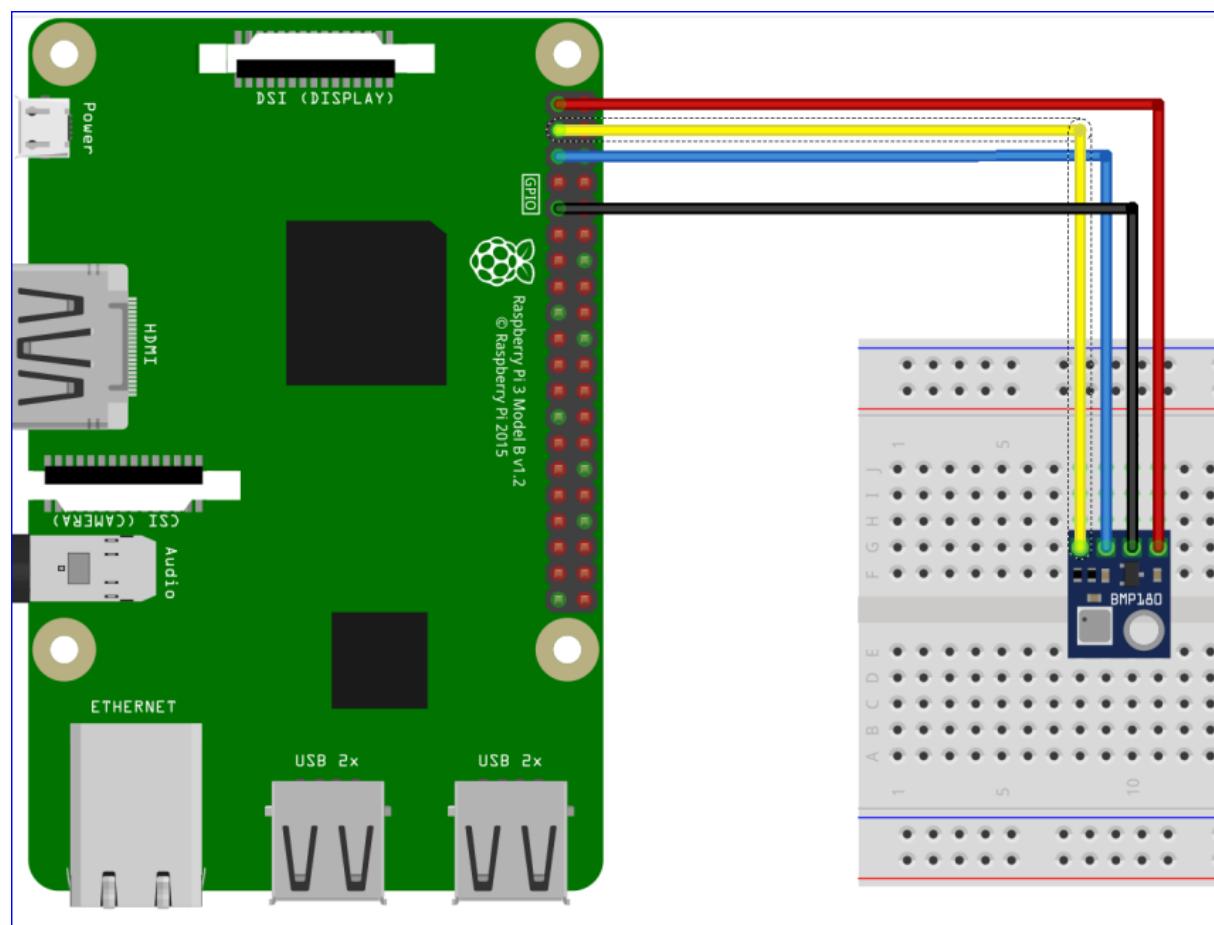
Purpose

To measure the Air Pressure using a BMP280 sensor connected to the GPIO of the Raspberry Pi.

Circuit

The BMP280 sensor (Type Sunfounder), is connected to the Raspberry Pi SDA & SCL pins.

BMP180	Raspberry Pi 3B+ (wirecolor)
3.3v	3.3v (red)
SDA	SDA = I2C1 = Pin #3 (yellow)
SCL	SCL = I2C2 = Pin#5 (blue)
GND	GND (black)
I2C Address	76 Use command <i>i2cdetect -y 1</i>



Note

Drawing created with Fritzing. Adding multiple I2C sensors, requires a unique address for each sensor and two 1.k resistors between Vcc and SDA, SCL.

I2C

Check if the I2C module (i2c_bcm2835) is loaded.

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

Below list is only an extract...

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

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

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

Hardware & Device

(Setup > Hardware)

Enabled:	<input checked="" type="checkbox"/>
Name:	BMP280
Type:	I2C sensors
Data Timeout:	Disabled
Specifying a Data Timeout will restart the hardware device if no data is received for the specified time. Do not enable this option for devices that do not receive data!	
SubType:	I2C sensor BME280 Temp+Hum+Baro
Path to I2C bus:	/dev/i2c-1 (etc. /dev/i2c-1, or empty for autodetection)

Note

The path to the I2C Bus is taken from previous 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 list

Idx	Name	Enabled	Type	Address	Port
13	BMP280	Yes	I2C sensor BME280 Temp+Hum+Baro	I2C	/dev/i2c-1

Check the log entry after adding the hardware

```
2018-08-31 11:34:50.758 Status: I2C Start HW with 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 the BMP280 Device

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

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

115	BMP280	0001	1	TempHumBaro	Temp + Humidity + Baro	Weather Station	20.2 C, 0 %, 1021.0 hPa
-----	--------	------	---	-------------	------------------------------	--------------------	-------------------------

Add the device, named BMP280 and check the widgets.

Weather Widget	Temperature Widget
<div style="border: 1px solid #ccc; padding: 5px;"> <p>BMP280 1021 hPa</p>  <p>Prediction: Sunny Last Seen: 2018-08-31 11:42:51</p> <p>Log Edit Notifications</p> </div>	<div style="border: 1px solid #ccc; padding: 5px;"> <p>BMP280 19.7° C / 0%</p>  <p>Dry, Barometer: 1027.8 hPa, Prediction: Sunny Dew Point: 19.70° C Last Seen: 2018-08-31 16:00:48</p> <p>Log Edit Notifications</p> </div>

Barometer Widget

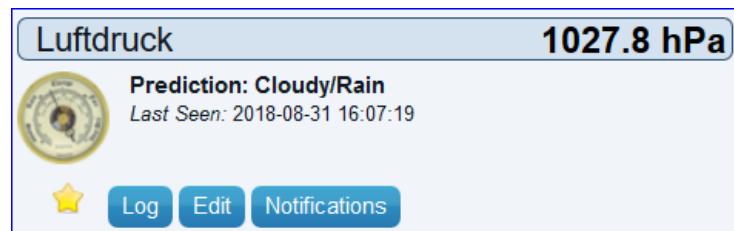
The BMP280 measures pressure, temperature and humidity.
 The temperature and the humidity are not of interest because these are measured at the Raspberry Pi location and not outside.
 In the Dashboard would like to have a Barometer Widget with value taken from the BMP280.

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



Note

Default (=wrong) data as the event to obtain date is developed next.

Barometer Event (dzEvents)

Created an event to update the pressure and pressure forecast from the BMP280 data.
 This is done via a Lua script using the Domoticz Event Editor.

115	BMP280	0001	1	BMP280	Temp + Humidity + Baro	Weather Station	19.5 C, 0 %, 1028.0 hPa
-----	--------	------	---	--------	------------------------------	--------------------	-------------------------

- TEMP = Temperature
- HUM = Humidity
- BAR = Barometric pressure

Event Editor

Event properties: luftdruck_update, dzEvents, Device

dzVents Script

```
--[[
luftdruck_update.lua
If the a value of the BMP280 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_BMP280 = 115;
local IDX_LUFTDRUCK = 116;

-- Event handling changes of the BMP280 device
return {
  on = {
    devices = {
      IDX_BMP280
    }
  },
  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 BMP device to the Barometer device
    -- See source code RFXNames.cpp, BMP_Forecast_Desc
    -- Map the BMP_Forecast_Desc string to dzEvents Forecast value:
    -- domoticz.BARO_STABLE, BARO_SUNNY, BARO_CLOUDY, BARO_UNSTABLE, BARO_THUNDERSTORM

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

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

Domoticz Log Entry

```
2018-09-11 13:51:36.606 Status: dzVents: Info: Handling events for: "BMP280",
value: "19.9;0;2;1024.0;4"
2018-09-11 13:51:36.606 Status: dzVents: Info: ----- Start internal script:
luftdruck_update: Device: "BMP280 (BMP280)", Index: 115
2018-09-11 13:51:36.606 Status: dzVents: Info: Barometer: 1024/4/Cloudy
2018-09-11 13:51:36.607 Status: dzVents: Info: ----- Finished luftdruck_update
```

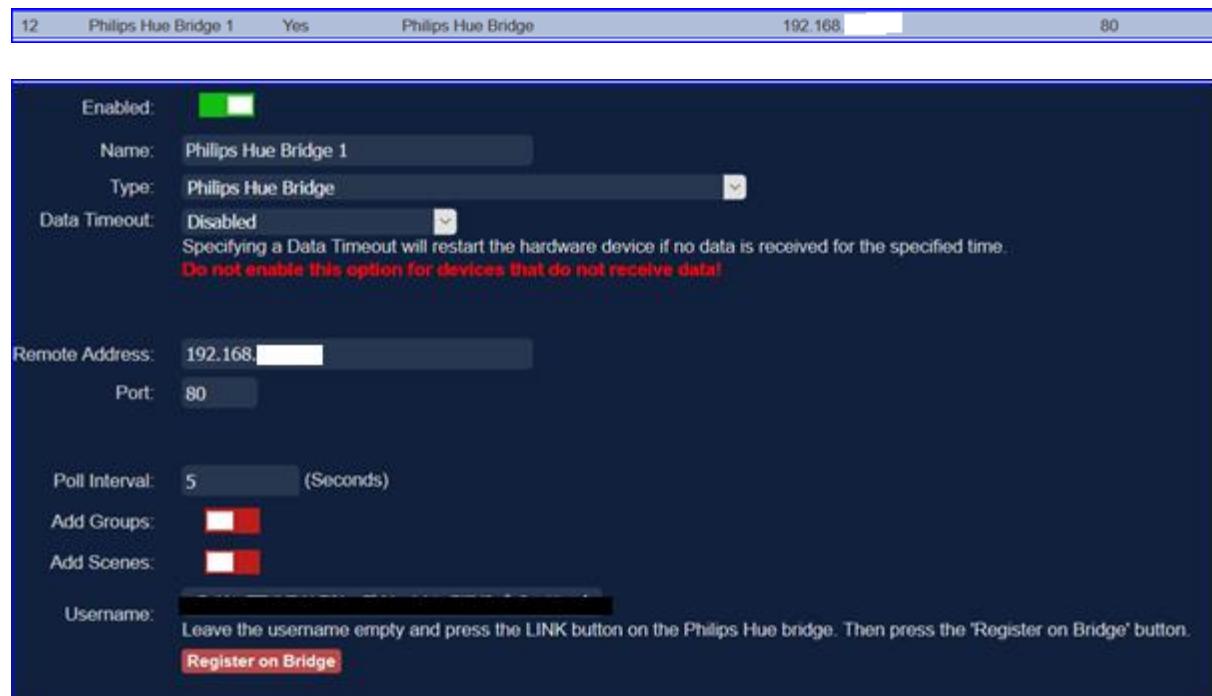
Philips Hue

Purpose

To control devices connected to the Philips Hue Bridge.

Hue Bridge Configuration

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



Note

Initially Groups & Scenes are not used.

Hue Devices

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

Idx	Hardware	ID	Unit	Name	Type	SubType
110	Philips Hue Bridge 1	0000001	1	Hue Haustür	Lighting 2	AC
111	Philips Hue Bridge 1	0000002	1	Hue WZ Ablage	Color Switch	WW
112	Philips Hue Bridge 1	0000003	1	Hue WZ TV	Lighting 2	AC
113	Philips Hue Bridge 1	0000004	1	Hue WZ Robs Ecke	Lighting 2	AC
118	Philips Hue Bridge 1	0000005	1	Hue MakeLab	Lighting 2	AC

Hue Light Add New

Example adding a Hue white light.

First the new Hue Lamp is added to the Hue Bridge. This is done by using the Hue App (Android).

After the Hue Lamp has been added to the Hue Bridge, Domoticz adds the device (idx=118) to the list (after Setup > Settings > Hardware allow for 5 minutes).

118	Philips Hue Bridge 1	0000005	1	Hue white lamp 4	Lighting 2	AC
-----	----------------------	---------	---	------------------	------------	----

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
---	---	--

Script 1 – Device Trigger

If the Hue Lamp is switched ON, then switch it OFF again.

dzEvents Script

```
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
}
```

Script 2 – Timer Trigger

Switch the Hue Lamp ON at ‘HH:MM’ and switch it OFF again one minute later.
This type of script could be used, to switch

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

dzEvents Script

```
-- 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()

    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.

dzEvents Script

```
--[[[  
    hue_wz_timer.lua  
    Turn WZ Hue lamps 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.  
]]--  
  
-- 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  
    end  
}
```

```
end

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

-- switch the hue lights on & off in the morning
-- NOT USED
-- if (timer.trigger == 'at 07:00') then hueswitchon(domoticz) end
-- if (timer.trigger == 'at 08:00') then hueswitchoff(domoticz) end

end
}
```

Domoticz Log Entry

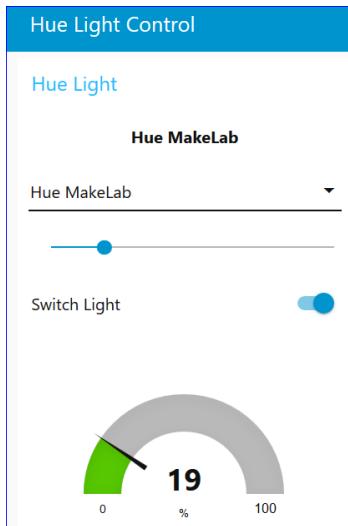
```
2018-09-05 20:02:00.394 (Philips Hue Bridge 1) Color Switch (Hue WZ Ablage)
2018-09-05 20:02:00.417 (Philips Hue Bridge 1) Lighting 2 (Hue WZ TV)
2018-09-05 20:02:00.259 Status: dzVents: Info: ----- Start internal script: hue_wz_timer:, trigger: at
sunset
2018-09-05 20:02:00.276 Status: dzVents: Info: ----- Finished hue_wz_timer
2018-09-05 20:02:00.357 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

Hue Lights Control Node-RED

An example of controlling Hue lights via Node-RED Dashboard UI in a webbrowser.
Select a Hue and switch the light ON|Off or set Level.

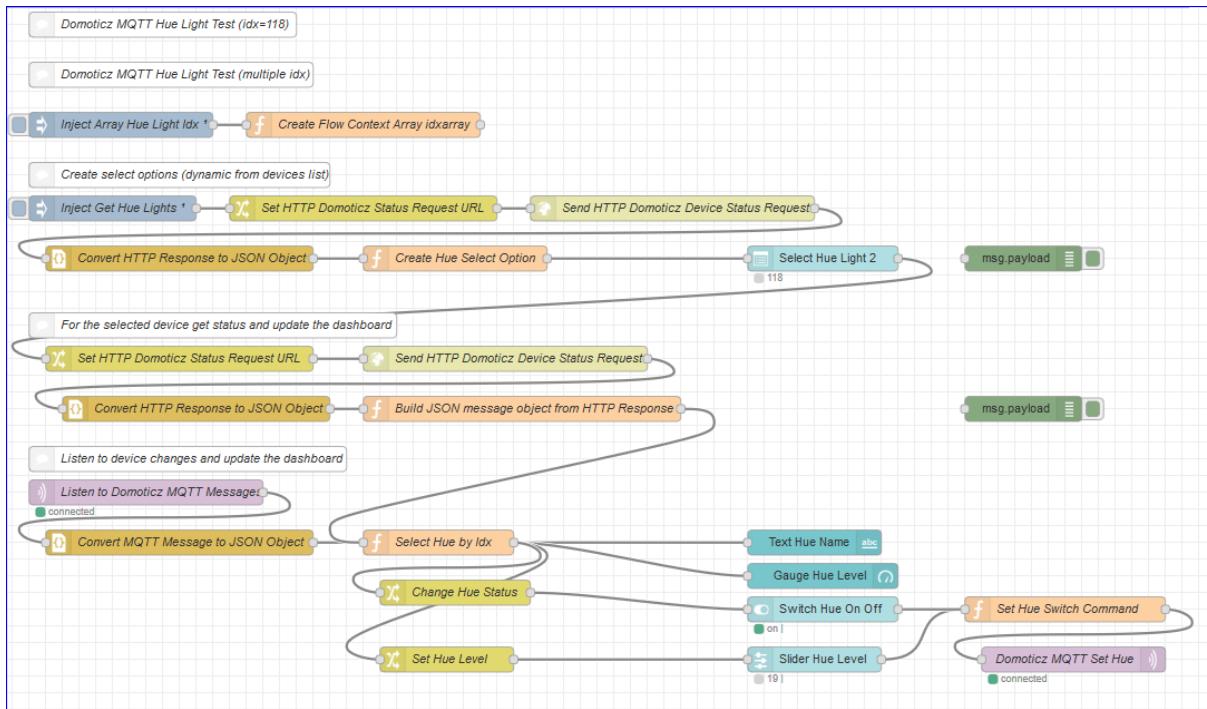
Read more about Node-RED [Appendix Node-RED](#).

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 by the Domoticz Web UI, then the Node-RED dashboard is updated.

Flow



Note

When the flow starts, a flow context array “idxarray” is defined with the Hue deviceidx’s, which are used for the dropdown with the Hue’s. 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 Domoticz Web UI), the MQTT listener node recognized this and the message payload is handled also by the node selecting the Hue with the connected nodes.

So, the „Select Hue by Idx“ node has two input, one from a converted HTTP request and the other from MQTT listener.

Remote Switch

Purpose

To remote switch devices, i.e. lamps etc.

Device

The remote switch used is a Mumbai Wireless Switch Module.
The remote switch is connected to the RFXCOM RFXtrx433E.

Setup

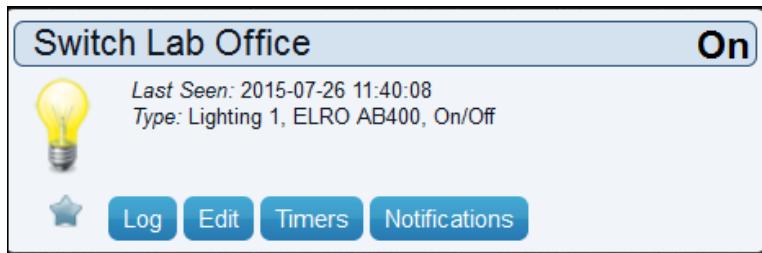
1. Select Switches > Manual Light Switch
2. Add the Properties > See Picture right
 - On the Remote Control of the Switch, the Type is displayed (AB440R which is compatible with AB400D)
 - To determine the House Code and the Unit Code, open the Switch and note the state of the dip switches and lookup the corresponding RFXCom values (see below)
3. Press Add Device
4. Select Setup > Devices
5. Select Switch Lab, click green V to add

House Code				Unit Code										Add Manual Light/Switch Device			
	1	2	3	4	5	A	B	C	D	E	5	A	B	C	D	Hardware:	RFXtrx433e
HC	=====				5	6	7	8	9	10	5	6	7	8	9	Device name:	Switch Lab
A	0	0	0	0	1	0	0	1	0	0	33	0	0	0	0	Switch Type:	On/Off
B	0	0	0	1	1	0	0	0	0	0	34	0	0	0	1	Type:	AB400D
C	0	0	1	0	1	0	0	0	0	0	35	0	0	1	0	House Code:	I
D	0	0	1	1	1	0	0	0	0	0	36	0	0	1	0	Unit Code:	3
E	0	1	0	0	0	1	0	0	0	0	37	0	1	0	0		
F	0	1	0	1	0	1	0	0	0	0	38	0	1	0	1		
G	0	1	1	0	0	1	1	0	0	0	39	0	1	1	0		
H	0	1	1	1	0	1	1	0	0	0	40	0	1	1	0		
I	1	0	0	0	0	0	0	0	0	0	41	1	0	0	0		
J	1	0	0	1	0	0	1	0	0	0	42	1	0	0	1		
K	1	0	1	0	0	1	0	0	0	0	43	1	0	1	0		
L	1	0	1	1	0	1	0	0	0	0	44	1	0	1	1		
M	1	1	0	0	0	0	1	0	0	0	45	1	1	0	0		
N	1	1	0	1	0	0	0	1	0	0	46	1	1	0	1		
O	1	1	1	0	0	1	0	0	0	0	47	1	1	1	0		
P	1	1	1	1	0	1	1	0	0	0	48	1	1	1	0		
											49	0	0	0	0		
											50	0	0	0	1		
											51	0	0	1	0		
											52	0	0	1	1		
											53	0	1	0	0		
											54	0	1	0	1		
											55	0	1	0	0		
											56	0	1	1	0		
											57	1	0	0	0		
											58	1	0	0	1		
											59	1	0	1	0		
											60	1	0	1	1		
											61	1	1	0	0		
											62	1	1	0	1		
											63	1	1	1	0		
											64	1	1	1	1		

Dip Switches for this device translated into HC and UC

1234 5ABCDE
1000 001000
HC=I UC=3

Remote Switch Widget



Waste Calendar

Purpose

To display waste calendar dates (“Abfall Termine”) for a city (location) and waste type.
The waste types are (in German): Rest, Bio, Gelb, Papier, Schad.
The dates are provided by the respective city waste management (“Abfallwirtschaft”) for the cities Pinneberg and Hohenfelde.
A room plan “Termine” is defined (Tab Setup > More Options > Plans > Roomplan).

Example Room Termine

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

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

Devices Configuration

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

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

Concept

A dzVents Lua script, “waste_calendar_update” executed dzVents time trigger reads a CSV file, which contains per line the Device Idx, Interval, waste dates.

The device is updated based upon the difference in days between now and the waste date within a given interval (i.e. 16 or 32 days).

The script also notifies via email (recipients defined in the Domoticz Settings Email and Notification) if the waste target date is 1 days from actual date.

CSV Input File

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

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

CSV Line Syntax: idx, interval, date1, dateN

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

Example CSV file line entry

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

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


CSV File name & location

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

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

dzVents Lua Script

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

```
-- [[
    waste_calendar_update.lua
    Read a csv file with wastecalendar dates per device.
    Update the device text with the date if the date is within interval specified.
    The wastecalender file "wastecal.csv" is located in the dzVents scripts folder.
    Line entry csv file: idx,interval,date1,dateN.
    Project: atHome
    Interpreter: dzVents, Timer
    See: athome.pdf
    Author: Robert W.B. Linn
    Version: 20180930
]]-- 

-- External modules: /home/pi/domoticz/scripts/dzVents/scripts
local utils = require('utils')
local msgbox = require('msgbox')

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

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

-- the wastecalender file is updated once per year
local wastecalfile = '/home/pi/domoticz/scripts/dzVents/scripts/wastecal.csv'
local sep = ","

-- read the csv file
function readandupdate(domoticz)
    -- read the file line by line.
    -- each line contains an array with device[1], wasteinterval[2], dates[2...]
    for line in io.lines(wastecalfile) do
        -- split the line by , and assign to a table
        local wastetable = line:split(sep)
        local idx = tonumber(wastetable[1])
        local interval = tonumber(wastetable[2])
        -- flag to check if a wastedate has been found
        local datefound = 0

        -- check the dates, start at 3 because 1=idx, 2=interval and number of table entries
        for i = 3, #wastetable do

            -- get the wastedate array entry
            local wastedate = wastetable[i]
```

```

-- split the wastedate into an array with 3 entries day, month, year
local wastedatesplit = wastedate:split("-")
-- calculate the daysdiff between now and the wastedate
local daysdiff = utils.datediffnow(wastedatesplit[1], wastedatesplit[2], wastedatesplit[3])
--
local month = tonumber(wastedatesplit[2])

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

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

    -- check number of days left compared to the threshold
    -- i.e. if 1 day left, then notify using device name & text.
    -- the device text contains the date
    -- NOT USED = handled by the dzVents script waste_calendar_notifier
    if daysdiff == TH_WASTECAL then
        -- print ('Device ' .. idx .. ' notifying...')
        message = '[ACHTUNG] ' .. domoticz.devices(idx).name .. ' ' ..
domoticz.devices(idx).text
            -- update alert message
            -- msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_RED, message)
            domoticz.log(message, domoticz.LOG_INFO)
            domoticz.notify(message)
    end
end

    -- check if for the last entry daysdiff less 0, then no new date available
    -- this is used f.e. for hohenfelde as last wastedate is sep or oct
if i == #wastetable and daysdiff < 0 then
    datefound = 1
    print(idx .. '=' .. wastedate .. ', days=' .. daysdiff .. ', interval=' .. interval .. ',
KEIN TERMIN')

    -- update the device text with the info that there is no date (yet) available
    domoticz.devices(idx).updateText('Kein neuer Termin')
end

end

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

    -- JUST SOME LEARNERS
    -- print the content
    -- for i = 1, #wastetable do
    --     print( wastetable[i] )
    -- end
end
end

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

```

Python CSV File Generation Script

A Python 2 script is used to create the wastecal.csv file.
The script reads a iCal file.

Script

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

## Notes:
## The generated csv file wastecal.csv must be placed in
/home/pi/domoticz/scripts/dzVents/scripts
## Domoticz Device Idx and Interval
## Biotonne Pi=98,16
## Papier Pi=99,32
## Restmuell Pi=100,16
## Gelber Sack Pi=101,16
## Schadstoff Pi=102,32
## Biotonne Ho=103,16
## Papier Ho =104,32
## Restmuell Ho=105,16
## Gelber Sack Ho=106,16

## The generated file has to be merged with the wastecal.csv file located on the
Domoticz server:
## /home/pi/domoticz/scripts/dzVents/scripts

import csv, io, sys
import datetime
from datetime import date

wastecalendarics = 'wastecal-pi-2019.ics'

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

# Check the python version to set the file open arguments
if sys.version_info[0] == 2: # Not named on 2.6
    icsaccess = 'rb'
    icskargs = {}
    csvaccess = 'wb'
    csvkargs = {}
else:
    icsaccess = 'rt'
    icskargs = {'newline':''}
    csvaccess = 'a'
    csvkargs = {}
    #csvkargs = {'newline':''}
```

```
# Define the ics tokens
TOKEN_EVENT_BEGIN = "BEGIN:VEVENT"
TOKEN_EVENT_END = "END:VEVENT"
TOKEN_EVENT_DESCRIPTION = "DESCRIPTION:"
TOKEN_EVENT_START = "DTSTART:"
TOKEN_REST = "Restabfall 2"
TOKEN_BIO = "Bio"
TOKEN_SCHAD = "Schad"
TOKEN_GELB = "Gelb"
TOKEN_PAPIER = "Papier"

# Define the lists holding the dates per waste types. The lists are used to create
# the csv file
pirestlist = []
pibiolist = []
pischadlist = []
pigelblist = []
pipapierlist = []
horestlist = []
hobiolist = []
hogelblist = []
hopapierlist = []

# Define helpers
eventfound = 0
eventdescription = ""
eventstart = 0
eventyear = 0
eventmonth = 0
eventday = 0

def csv2string(data):
    si = io.BytesIO()
    cw = csv.writer(si)
    cw.writerow(data)
    return si.getvalue().strip('\r\n')

# Get the actual year month day
s = datetime.date.today().strftime("%d-%m-%Y")
# Get the date now in format i.e. 09-08-2018
datenow = datetime.datetime.strptime(s, "%d-%m-%Y")

# Set the waste type as the first entry in the waste lists
## Biotonne Pi=98,16
## Papier Pi=99,32
## Restmuell Pi=100,16
## Gelber Sack Pi=101,16
## Schadstoff Pi=102,32
## Biotonne Ho=103,16
## Papier Ho =104,32
## Restmuell Ho=105,16
## Gelber Sack Ho=106,16

pibiolist.append('98,16')
pipapierlist.append('99,32')
pirestlist.append('100,16')
pigelblist.append('101,16')
pischadlist.append('102,32')
hobiolist.append('103,16')
hopapierlist.append('104,32')
```

```
horestlist.append('105,16')
hogelblist.append('106,16')

# Open the wastecalendar ics file in read access
with open(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

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

# Write the lists to the CSV file. Each row contains a waste type with dates.
# Sample: 98,16,12-01-2018,09-02-2018, ...
with open(wastecalendarcsv, csvaccess, **csvkwargs) as csvfile:
    # Pinneberg
    csvfile.write(",".join(pipapierlist))
    csvfile.write("\n")
    csvfile.write(",".join(pirestlist))
    csvfile.write("\n")
    csvfile.write(",".join(pibiolist))
    csvfile.write("\n")
    csvfile.write(",".join(pigelblist))
    csvfile.write("\n")
    csvfile.write(",".join(pischadlist))
    csvfile.write("\n")
    # Hohenfelde - the dates needs to be added manually
    csvfile.write(",".join(hopapierlist))
    csvfile.write("\n")
    csvfile.write(",".join(horestlist))
    csvfile.write("\n")
    csvfile.write(",".join(hobiolist))
    csvfile.write("\n")
    csvfile.write(",".join(hogelblist))
    csvfile.write("\n")
    # Close with message
    csvfile.close()
    print "Created CSV file", wastecalendarcsv

# Exit the script
icsfile.close()
sys.exit()
```

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 script**, once a day at 00:15, to calculate the date difference in days and update the devices.

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

Event Properties

Event name: days_to_go_update.lua

Interpreter: dzVents

Event type: Timer

Event active

Event Script dzVents

(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
```

Raspberry Pi Monitoring

Purpose

To measure & show specific Raspberry Pi system information and trigger alarm if a value is above a certain threshold.

Monitored are: Memory Usage, Hard Disc Space, Temperature.

Solution

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

Hardware Motherboard

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

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

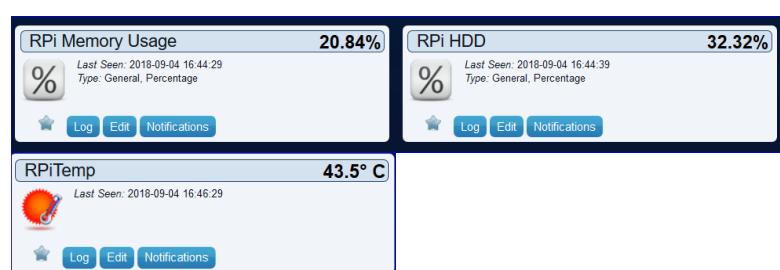
Devices derived from Hardware Motherboard

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

Note

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

Widgets (Tabs Utility & Temperature)



Monitoring

To monitor selective devices and notify if above threshold.

The notification is done using the Virtual Device Alert Message (idx=55).



An email notification is 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

dzVents Script

```
--[[[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 messagebox: /home/pi/domoticz/scripts/dzVents/scripts
local messagebox = require('messagebox')

-- 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 .. ')'
        messagebox.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
}
```

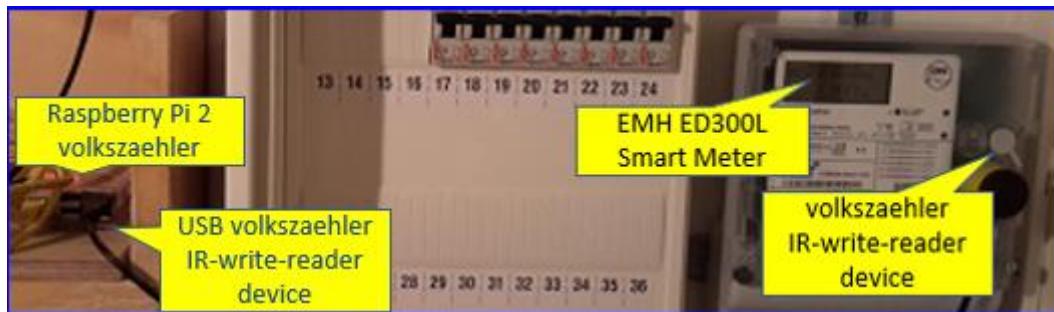
Electric Usage

Purpose

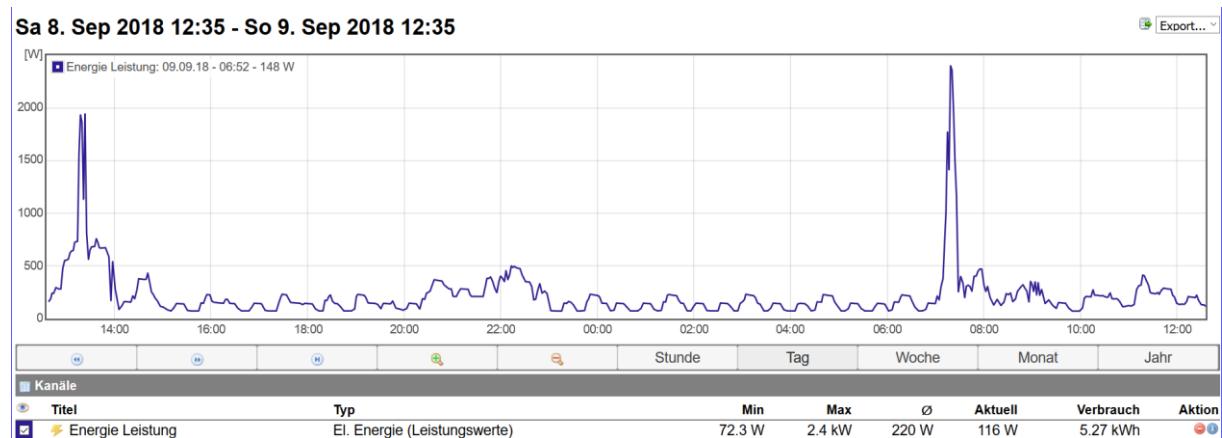
To measure the Electric Usage (Watt) obtained from a [volkszaehler](#) - Open Source Smart Meter - connected to a Raspberry Pi 2.

Detailed setup see [Appendix volkszaehler Setup](#).

Hardware Setup



Volkzaehler Web Frontend



Solution

To get the electric usage from the volkszaehler and update the Domoticz device, a dzVents script is executed:

1. Send, every 5 minutes, a 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.
3. Parse the JSON string to get the electric usage from key **data.average**. (which calculates the average electric usage over the last 5 minutes)
4. Update the Domoticz Electric Usage device:
idx=44, VirtualSensors, Name=Stromverbrauch, Type=Usage, SubType=Electric.

Domoticz Device

44	VirtualSensors	82043	1	Stromverbrauch	Usage	Electric	140 Watt
----	----------------	-------	---	----------------	-------	----------	----------

Stromverbrauch
113 Watt

Last Seen: 2018-09-08 11:46:00
Type: Usage, Electric

★
Log
Edit
Notifications

The HTTP GET request uses the channel (UUID: 958bce60-b342-11e8-b54f-bbec0573e1f4) and timestamp parameter:

- **from=5+minutes+ago**
- **to=now**

Request Examples

(see also the volkszaehler reference)

Data last 5 minutes (ensure to replace, in the URL request, the space by a +)

```
http://NNN.NNN.1.139/middleware.php/data/958bce60-b342-11e8-b54f-
bbec0573e1f4.json?from=5+minutes+ago&to=now
```

Data now

```
http://NNN.NNN.1.139/middleware.php/data/958bce60-b342-11e8-b54f-
bbec0573e1f4.json?from=now&to=now
```

Request Result JSON string

```
{"version": "0.3", "data": {"tuples": [[[1536243170351, 926.045, 1]]], "uuid": " 958bce60-
b342-11e8-b54f-
bbec0573e1f4", "from": 1536243168796, "to": 1536243170351, "min": [1536243170351, 926.045
0147835], "max": [1536243170351, 926.0450147835], "average": 926.045, "consumption": 0.4,
"rows": 2}}
```

Obtain the power consumption from key

`data.average`

which is 926.045

dzEvents Script

The event uses a callback to handle the volkszaehler server response. The response is the JSON string from which the average electric usage is parsed.

```
-- HTTP GET request url with parameter from=5+minutes+ago, to=now
local requesturl = 'http://NNN.NNN.1.139/middleware.php/data/958bce60-b342-11e8-b54f-
bbec0573e1f4.json?from=5+minutes+ago&to=now'

-- Idx of the devices
local IDX_POWERCONSUMPTION = 44

return {
    -- active = true,
    on = {
        timer = {
            'every 5 minutes'
            -- 'every minute'
        },
        httpResponses = {
            'powerData'
        }
    },
    execute = function(domoticz, item)
        if (item.isTimer) then
            domoticz.openURL({
                url = requesturl,
                method = 'GET',
                callback = 'powerData'
            })
        end
        if (item.isHTTPResponse and item.ok) then
            -- parse the json string to get the average from key data.average
            local powerconsumption = tonumber(item.json.data.average)

            -- update the powerconsumption device
            domoticz.devices(IDX_POWERCONSUMPTION).updateEnergy(math.floor(powerconsumption))
            domoticz.log('Powerconsumption updated: ' .. powerconsumption)
        end
    end
}
```

Domoticz Log Entry

The log shows the HHTP request and the callback few seconds later

HTTP Request

```
2018-09-06 17:00:00.285 Status: dzVents: Info: ----- Start internal script:
powerconsumption_update:, trigger: every 5 minutes
2018-09-06 17:00:00.286 Status: dzVents: Info: ----- Finished
powerconsumption_update
2018-09-06 17:00:00.371 Status: EventSystem: Script event triggered:
/home/pi/domoticz/dzVents/runtime/dzVents.lua
```

Callback

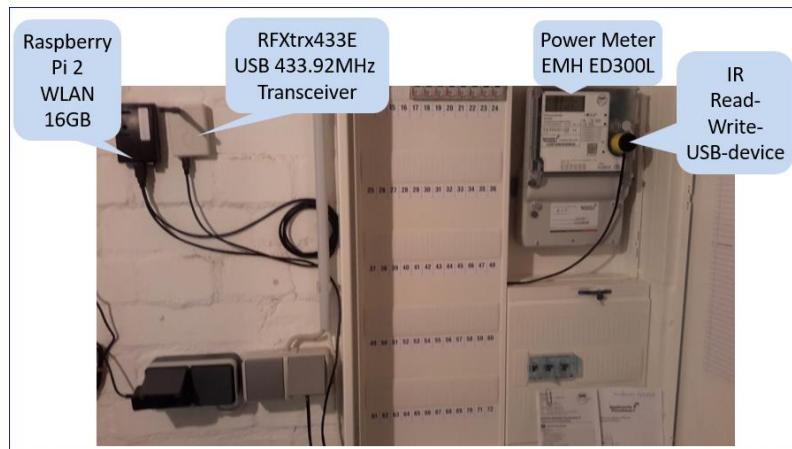
```
2018-09-06 17:00:05.817 Status: dzVents: Info: Handling httpResponse-events for:
"powerData"
```

```
2018-09-06 17:00:05.818 Status: dzVents: Info: ----- Start internal script:  
powerconsumption_update: HTTPResponse: "powerData"  
2018-09-06 17:00:05.844 Status: dzVents: Info: Powerconsumption updated: 150.206  
2018-09-06 17:00:05.844 Status: dzVents: Info: ----- Finished  
powerconsumption_update
```

volkszaehler

Overview

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



Note

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

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

Setup Raspberry Pi

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

Fix IP Address

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

/etc/dhcpcd.conf

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

/etc/wpa_supplicant/wpa_supplicant.conf

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

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

USB Port Symlink

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

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

Check

```
ls /dev/ttyU*
```

Serial Link

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

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

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

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

Must read: https://wiki.volkszaehler.org/howto/emh_pv-anlage?redirect=1

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

IMPORTANT

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

These are NOT S0-Impulses, but counters, i.e. „Zählerstände, Leistungswerte“ depending Obis-Code.

There are two channels defined, from which channel 2 - Energy Consumption (16.7.0 Leistung) is used for Domoticz.

Channel 1: Haus 1.8.0 (Bezug +A), Typ El. Energie (Zählerstände), Auflösung 1000
UUID:5b8ea2a0-b342-11e8-bec6-15c040e6d041

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	Haus 16.7.0 (Leistung), Typ El. Energie (Leistungswerte)
UUID 5b8ea2a0-b342-11e8-bec6-15c040e6d041	UUID 958bce60-b342-11e8-b54f-bbec0573e1f4
<pre>// /etc/vzlogger.conf { // Zählerstand "uuid" : "5b8ea2a0-b342-11e8-bec6-15c040e6d041", "middleware" : "http://localhost/middleware.php", "identifier" : "1-0:1.8.0" }</pre>	<pre>/etc/vzlogger.conf { // Leistungswert "uuid" : "958bce60-b342-11e8-b54f-bbec0573e1f4", "middleware" : "http://localhost/middleware.php", "identifier" : "1-0:16.7.0" }</pre>

Web Frontend Settings	Web Frontend Settings
{ "version":"0.3", "entity": {"uuid":"5b8ea2a0-b342-11e8-bec6- 15c040e6d041", "type":"electric meter", "color":"aqua", "fillstyle":0, "public":true, "resolution":1000, "style":"steps", "title":"Energie Z\u00fchlerstand", "yaxis":"auto"} }	{ "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"} }

vzlogger

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

Configuration File

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

Ensure no empty lines are spaces after the last }.

IMPORTANT:

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

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

/etc/vzlogger.conf

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

Start Stop Status

Terminal commands

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

LogFile

Check

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

Delete

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

vz Web Frontend

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

Read more [here](#).



Database Management

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

User vz-admin

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

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

Login phpMyAdmin

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

Table	Action	Rows	Type	Collation	Size	Overhead
aggregate	Browse Structure Search Insert Empty Drop	2,612,489	InnoDB	latin1_swedish_ci	190.3 Mib	-
data	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 Kib	-
entities	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8_unicode_ci	32 Kib	-
entities_in_aggregator	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8_unicode_ci	48 Kib	-
properties	Browse Structure Search Insert Empty Drop	127	InnoDB	utf8_unicode_ci	48 Kib	-
5 tables	Sum	2,612,632	InnoDB	latin1_swedish_ci	190.5 Mib	0 B

Check Database Size

URL

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

Result

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

Size of database and access_log

Limit logging

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

Delete Data

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

Option 1

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

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

```
mysql --user=root -praspberry

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

mysql> use volkszaehler

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

Note

This operation might take long...

Check the database size after completion

Option 2

Use terminal command

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

Option 3

Use URL

Prepare mysql

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

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

URL deleting data

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

Result JSON

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

URL Check database size

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

Result JSON

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

vzclean

Purpose

To delete, every day at midnight, data which is older 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://rpi-vz-ip/middleware.php/capabilities/database.json?
```

Python

vzcleanconfig.py

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

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

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

    # print(db)
    return db
```

vzclean.ini

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

vzclean.py

```
#!/usr/bin/env python
"""
File: vzclean.py
Author: Robert W.B. Linn, 20180909
Project: AtHome
Purpose: Delete all channel data prior last 24 h from database volkszaehler, table data.
Run once per day via crontab.
The scripts logs to a file. If the script is executed with parameter 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 COUNT()
    :return: Number of records
    """
    cursor = conn.cursor()
    cursor.execute(qry)
    result = cursor.fetchone()
    return result[0]

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

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

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

    try:
        conn = MySQLConnection(**db_config)

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

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

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

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

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

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

Bash

vzclean.sh

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

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

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

Crontab

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

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

Coffee Machine Monitor

Purpose

To monitor the start & end time of the Coffee Machine.

The Coffee Machine switches OFF, two hours after being switched ON.

Solution

Monitor the Electric Usage Device (idx=44) value if above certain threshold (> 1500 Watt) between 07:00 – 08:00. If above threshold, set the Alert Message.

Monitoring is handled by a dzVents Script for device changes and timer.

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

dzVents Script

```
local msgbox = require('msgbox')
local IDX_STROMVERBRAUCH = 44
local TH_STROMVERBRAUCH = 1500
local MONITOR_START = 'at 07:00'
local MONITOR_START_END = 'at 07:01-07:59'
local MONITOR_END = 'at 08:00'

return {
  on = {
    devices = {
      IDX_STROMVERBRAUCH
    },
    timer = {
      MONITOR_START,
      MONITOR_START_END,
      MONITOR_END,
    },
  },
  data = {
    notified = { initial = 0 }
  },
  execute = function(domoticz, item)
    if (item.isTimer) then
      if (item.trigger == MONITOR_START) then
        domoticz.data.notified = 0
      end
      if (item.trigger == MONITOR_END) then
        timeoff = os.date("*t")
        message = ('Kaffee Machine AUS %02d:%02d'):format(timeoff.hour, timeoff.min)
        msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREY, message)
      end
    end
    if (item.isDevice) then
      device = item
      if (device.WhActual > TH_STROMVERBRAUCH) and (domoticz.data.notified == 0) then
        timeon = os.date("*t")
        message = ('Kaffee Machine AN %02d:%02d'):format(timeon.hour, timeon.min)
        -- 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 = ('Kaffee Machine aus um %02d:%02d'):format(timeoff.hour, timeoff.min)
      end
    end
  end
}
```

```
-- update alert message
msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_ORANGE, message)
-- set notified flag
domoticz.data.notified = 1
end
end
end
}
```

Alert Message

Purpose

To display an Alert Message on the UI tabs Dashboard and Utility.

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

- Raspberry Pi monitor (rpi_moonitor.lua)
- Hue lamps living room timed switch (hue_wz_timer.lua)
- More see the dedicated functions

Solution

Add a Virtual Devices named “Info Meldung”, Type General, SubType Alert, Idx=55.

	55	VirtualSensors	82054	1	Alert Message	General	Alert	Hue MakeLab switched OFF 18:56:43
--	----	----------------	-------	---	---------------	---------	-------	-----------------------------------

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

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

Dashboard

	Info Meldung	
--	--------------	--

Widget

Info Meldung



RPi Temperature above threshold 45 (48.3)
Last Seen: 2018-09-10 07:40:47
Type: General, Alert

[Log](#) [Edit](#) [Notifications](#)

Log Entries

Info Meldung	
Show	entries
Date	Data
2018-09-10 07:40:47	RPi Temperature above threshold 45 (48.3)
2018-09-09 22:30:00	WZ Hue Lampen AUS 22:30:00
2018-09-09 19:53:00	Hue Lampen WZ AN19:53:00
2018-09-09 17:20:09	Keine
Showing 1 to 4 of 4 entries	
First Previous 1 Next Last	
Clear	

dzEvents Script

Simple example: Switch the Hue MakeLab ON, then the event will turn the lamp OFF and updates the alert sensor with a message.

```
local IDX_HUE_MAKELAB = 118;

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

local function setalertmsg(domoticz, level, msg)
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)
end

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

For all events use the function:

```
-- Update the alert message with level.
local function setalertmsg(domoticz, level, msg)
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)
end
```

Example

```
local msg = device.name .. ' > ' .. threshold .. ' (' .. device.state .. ')'
setalertmsg(domoticz, domoticz.ALERTLEVEL_YELLOW, msg)
```

Alert Message Reset

Purpose

To reset the Alert Message using a Switch.

Solution

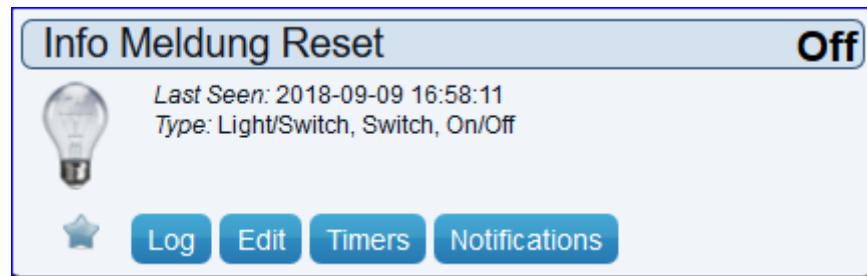
Create a virtual device type: Light/Switch, SubType: Switch.

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

119	VirtualSensors	000140C7	1	Info Meldung Reset	Light/Switch	Switch	Off
-----	----------------	----------	---	--------------------	--------------	--------	-----

Widget



User Variable

Idx	Variable name	Variable type	Current value
7	DEF_ALERTMSG	String	Keine

dzVents Script

```
--[[
    alertmsg_reset.lua
    If the switch is turned ON,
    the text of the alert sensor (idx=55) is set to the value of the User Variable
DEF_ALERTMSG and
    the switch is turned OFF again.
]]--
```

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

-- Idx of the devices
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
                msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREY, message)
                domoticz.log(message)
            end
            device.switchOff()
        end
    end
}
```

Database Tables

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

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

The log entries are stored in table LightingLog.

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

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

Info Message (Not Used)

Purpose

NOT USED – Replaced by the Alert Message.

To display an Info Message on the Dashboard.

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

- Raspberry Pi monitor (rpi_moonitor.lua)
- Hue lamps living room timed switch (hue_wz_timer.lua)
- 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 dzEvents, the device is updated.

Control Message	Hue MakeLab switched OFF 17:18:05	Last Seen: 2018-09-05 17:18:05	Type: General, Text	Hue MakeLab switched OFF 17:18:05
Text	Log	Edit		

dzEvents Script

```
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
}
```

Web Frontend Customized

Purpose

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

Web Front End Sample

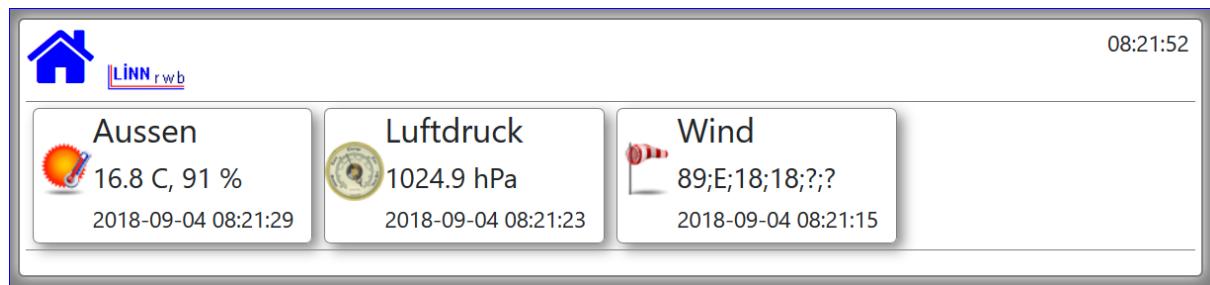
Display weather data from 3 devices and update every 5 seconds. A clock is running every second.

Devices Used

```
Temperature & Humidity
(Name="Aussen", idx=14, Hardware=RFXtrx433e, Type=Temp + Humidity, SubType=Cresta,
TFA TS34C)

Luftdruck
(Name="Luftdruck", idx=116, Hardware=VirtualSensors, Type=General,
SubType=Barometer)

Wind
(Name="Wind", idx=117, Hardware=VirtualSensors, Type=Wind, SubType=WTGR800)
```



Concept

The device data is requested (URL) via the Domoticz engine using JSON.

The Domoticz engine allows to interact with all devices using JSON (nice solution).

The UI uses jQuery, Bootstrap and Domoticz icons.

The data for the favorite (=Dashboard) devices is requested from the Domoticz server via URL.

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

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

JSON Result Snippet

The URL returns the data in JSON format.

This snippet shows some of the properties for the devices.

The key **result[]** is an array with the device information.

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

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

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

JavaScript Snippet

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

HTML & JavaScript

```

<!DOCTYPE html>
<html>
<head>
    <title>AtHome Weather Frontend</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8"/>
    <!-- Reference to the external CSS -->
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
    awesome/4.7.0/css/font-awesome.min.css">
    <!-- Reference to the external JS -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script
        src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
    <!-- Styling -->
    <style>
        body{background-color: grey; padding: 0px 1% 0% 1%;}

        .container{padding: 0px 20px 20px 20px; border-radius: 5px; background-color: white;
        border-style: solid; border-color: grey; border-width: 2px; box-shadow: 0px 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 5px 10px
        #888888; border-style: solid; border-color: grey; border-width: 1px; border-radius: 5px; }

        .sensor-icon{vertical-align: 80%; margin: 0px; }

        .sensor-data {display: inline-block; margin: 0px; }

    </style>
</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-sm-12" id="top-bar">
                <i class="fa fa-home" style="font-size:60px; color:blue;"></i>
                
                <div id="clock">
                </div>
            </div>
        </div>
        <div class="row" id="sensor-container">
        </div>
    </div>

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

```

```

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

// Returns the image of the sensor that is stored in the image folder on the Domoticz
server
// /home/pi/domoticz/www/images
function getSensorImage(TypeImg){
    var sensorURL = "";
    switch(TypeImg) {
        case "dimmer":
            return domoticzURL + "images/Light48_On.png";
            break;
        case "temperature":
            return domoticzURL + "images/temp-20-25.png";
            break;
        case "wind":
            return domoticzURL + "images/wind48.png";
            break;
        case "gauge":
            return domoticzURL + "images/baro48.png";
            break;
        default:
            return domoticzURL + "images/Light48on.png";
    }
}

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

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

function updateSensors() {
    // Get data for the favorite (=Dashboard) devices and select the idx=14,116,117
    // URL: domoticzurl:8080/json.htm?type=devices&used=true&filter=all&favorite=1
    url = domoticzURL + "json.htm?type=devices&used=true&filter=all&favorite=1"
    $.getJSON(url, function(result){
        var idx = 0;
        var sensorInfo = "";

```

```
        for (i = 0; i < result.result.length; i++) {
            idx = parseInt(result.result[i].idx);
            if (arrayIdx.indexOf(idx) > -1) {
                sensorInfo += setSensorInfo(result,i);
            }
        }
        document.getElementById("sensor-container").innerHTML = sensorInfo;
        //Finally set a timer to repeatedly poll the status of the sensors
        setTimeout(updateSensors,intervalTimer);
    });
}
});
</script>
</body>
</html>
```

App atHome

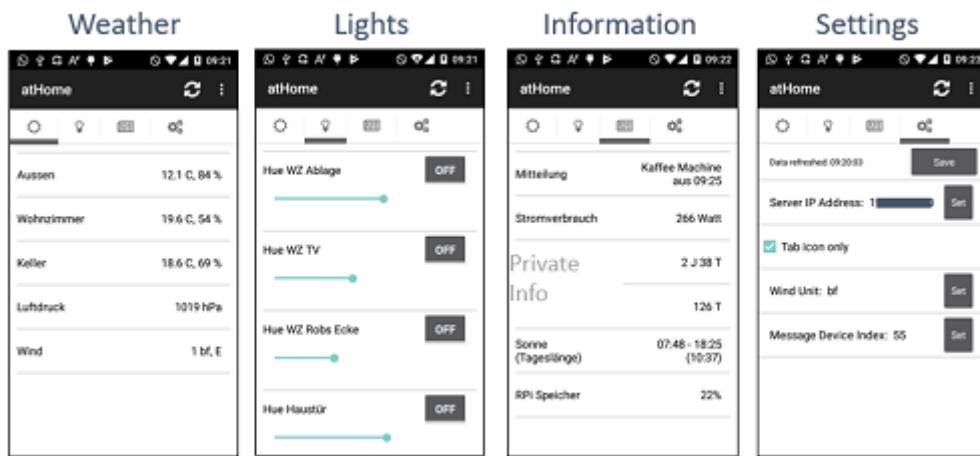
Purpose

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

Solutions

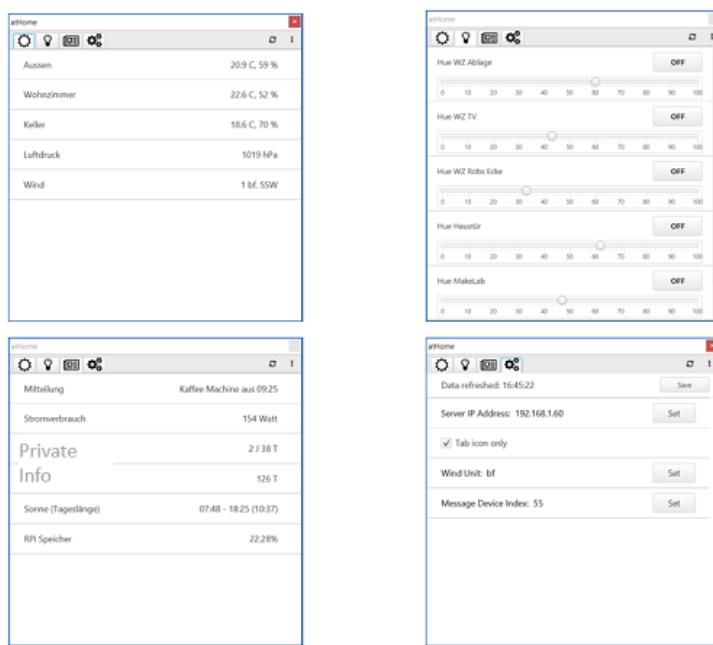
Android app developed with [B4A](#) v8.5.

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



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

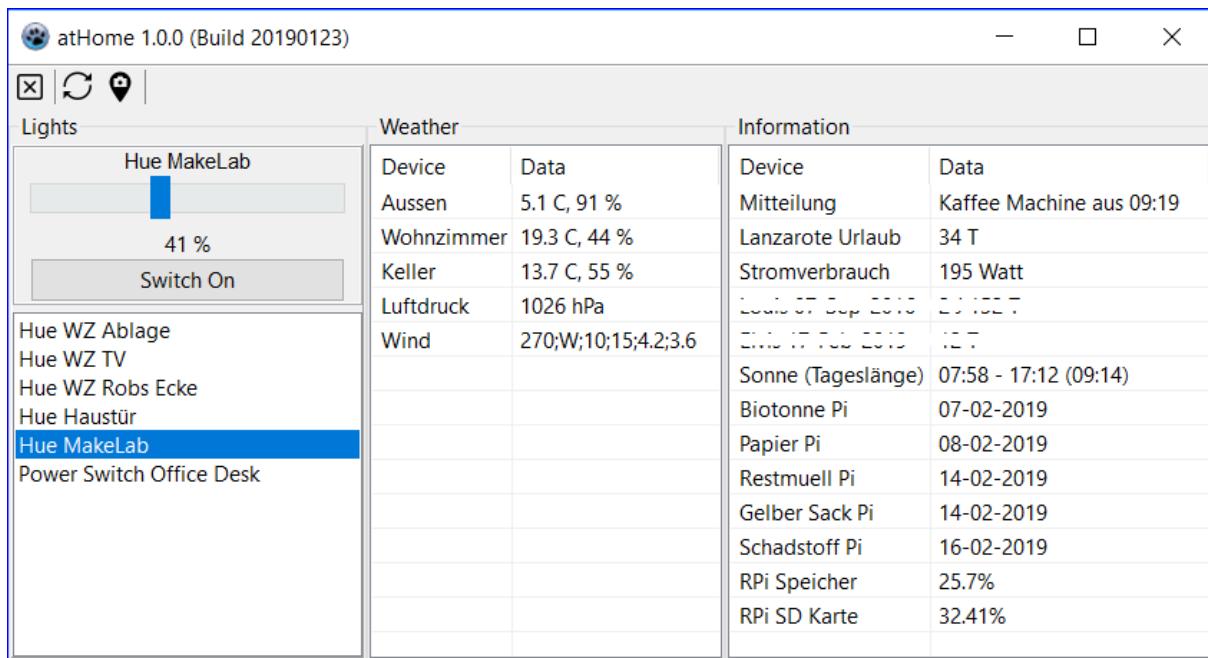
Read the B4J Forum *Share My Creations* post [here](#).



Note: current version in use has many more devices.

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

Not published yet.



Volumio Webradio

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

Light/Switch Devices

- Volumio Play (On)
- Volumio Refresh (On)
- Volumio Volume (Slider from 0 to 100, set to ~85)
- Volumio Bookmark (On)

Utility Sensors

Icon	Label	Value
Bluetooth icon	Volumio Current Song	This is the song title
Bluetooth icon	Volumio Favorites	This is the last bookmarked song title
Bluetooth icon	Volumio Status	PLAY

Volumio Current Song List sample	Volumio Favorites List sample
Show 25 entries Date 2019-02-10 12:18:25 song title 2019-02-10 12:14:04 song title 2019-02-10 12:08:38 song title 2019-02-10 12:04:18 song title	Show 25 entries Date 2019-02-10 11:45:05 song title 2019-02-10 11:33:13 song title 2019-02-10 11:18:31 song title 2019-02-10 11:15:55 song title
Volumio Status PLAY = switch Volumio Play set to On	
Volumio Status	PLAY
Volumio Status STOP = if the server is started or if switch Volumio Play is set to Off	
Volumio Status	STOP
Volumio Status OFF = server not reachable (network) or server turned off (shutdown)	
Volumio Status	OFF

Volumio Setup

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

Download the Volumio 2 Pine A64 SD Image from [here](#) (see additional [info](#)).

Image version: Volumio Digital Audio Player [2.383-2018-03-17].

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

Get Started

To get started, read the [Volumio manual](#) first.

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

Solution Options

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

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

Domoticz dzVents Scripts

Identified two challenges to resolve:

1. The dzVents timer updates every 1 minute which means that if a new song is playing, the displayed current song is still the previous and could also result in bookmarking the wrong song.
Resolved: Adding a switch enabling to update current state.
2. If the Volumio is not reachable, the Domoticz Error Log is getting spoiled with messages.
Resolved: If Volumio state is OFF, set play mode to OFF which does not trigger the timer to request Volumio status update.

Node-RED

Option 1

with the libraries [MPD](#) and [Advanced Ping](#) went well until handling volume changes – did not work via the `setvol` command. There might be more issues with MPD commands.

Option 2

Decided to use Node-RED to control Volumio via REST API commands as documented [here](#), thus to stick to what's defined (supported) by Volumio.

The available subset of Volumio REST API commands are suitable for this solution.

The communication between Volumio and Domoticz is based on MQTT messaging.

To know

This solution does not update the status of Domoticz Devices if Volumio is controlled via the Volumio Web Interface or any other (like an App).

<TODO>For future solutions, consider implementing in Volumio MQTT to be handled by Domoticz.

Solution dzVents

Domoticz Configuration

Devices

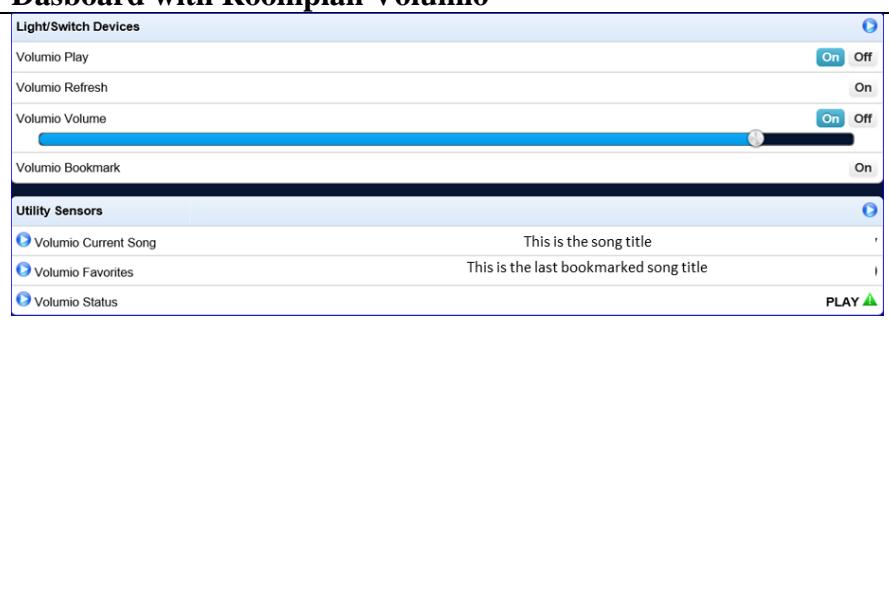
Idx	Name	Type	SubType	Function
145	Volumio Bookmark	Light/Switch	Push On Button	Switch ON adds (bookmarks) the current song playing to Favorites device (List)
146	Volumio Favorites	General	Text	Favorite songs
148	Volumio Volume	Light/Switch	Switch	Set the volume between 0- 100
149	Volumio Play	Light/Switch	Switch	Set Play (switch On) or Stop (switch Off)
150	Volumio Shutdown	Light/Switch	Switch	NOT USED <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	Push On Button	Updates the current song and the status.

Roomplan

Created a roomplan *Volumio* with the previous defined devices

(Domoticz Web UI Setup > More Options > Plans > Roomplan).

The roomplan (idx=5) is used as the Domoticz Dashboard to control Volumio.

Roomplan Volumio	Dashboard with Roomplan Volumio																		
 <p>Showing 1 to 5 of 5 entries</p> <p>Edit Delete</p> <p>Devices (Select Plan first to Edit...)</p> <p>Show [dropdown] entries</p> <table border="1"> <thead> <tr> <th>Idx</th> <th>Name</th> </tr> </thead> <tbody> <tr><td>149</td><td>Volumio Play</td></tr> <tr><td>155</td><td>Volumio Refresh</td></tr> <tr><td>148</td><td>Volumio Volume</td></tr> <tr><td>144</td><td>Volumio Current Song</td></tr> <tr><td>145</td><td>Volumio Bookmark</td></tr> <tr><td>146</td><td>Volumio Favorites</td></tr> <tr><td>151</td><td>Volumio Status</td></tr> <tr><td>150</td><td>Volumio Shutdown</td></tr> </tbody> </table> <p>Showing 1 to 8 of 8 entries</p>	Idx	Name	149	Volumio Play	155	Volumio Refresh	148	Volumio Volume	144	Volumio Current Song	145	Volumio Bookmark	146	Volumio Favorites	151	Volumio Status	150	Volumio Shutdown	 <p>Light/Switch Devices</p> <ul style="list-style-type: none"> Volumio Play Volumio Refresh Volumio Volume Volumio Bookmark <p>Utility Sensors</p> <ul style="list-style-type: none"> Volumio Current Song: This is the song title Volumio Favorites: This is the last bookmarked song title Volumio Status: PLAY ▲
Idx	Name																		
149	Volumio Play																		
155	Volumio Refresh																		
148	Volumio Volume																		
144	Volumio Current Song																		
145	Volumio Bookmark																		
146	Volumio Favorites																		
151	Volumio Status																		
150	Volumio Shutdown																		

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

dzVents Scripts

The dzVents Lua scripts are created and maintained via Domoticz Web UI (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= (extract from the script *volumio_play*):

```
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

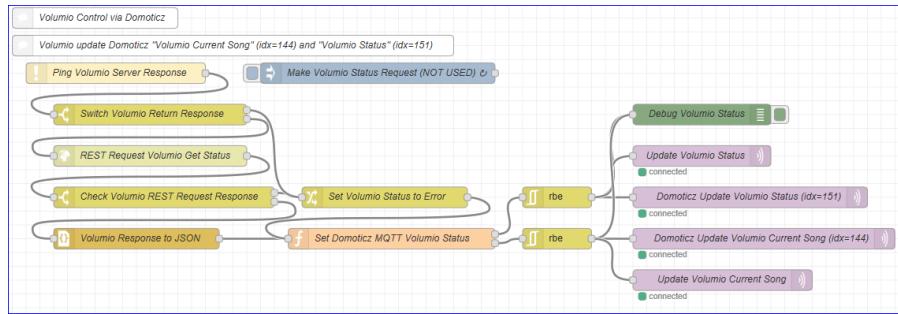
Switch Volumio Refresh

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 Web UI Setup > More Options > Plans > Roomplan).

The roomplan is used as the Domoticz Dashboard (see earlier screenshot) to control Volumio.

Screenshot Roomplan number 5 named Volumio with 7 devices

Idx	Name
149	Volumio Play
148	Volumio Volume
144	Volumio Current Song
145	Volumio Bookmark
146	Volumio Favorites
151	Volumio Status
150	Volumio Shutdown

Volumio API Commands

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

Volumio

([Reference](#))

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

The getstate command is triggered every 5 seconds by an *Inject Node*.

The response is either a string in case the Volumio server is not reachable (which is converted to a JSON string with status error) or a JSON string containing current song information.

Domoticz

([Reference](#))

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

The listed message payloads are created in *Function Nodes* with variables, like the idx names and current song (i.e. var idxvolumiostatus = 151;).

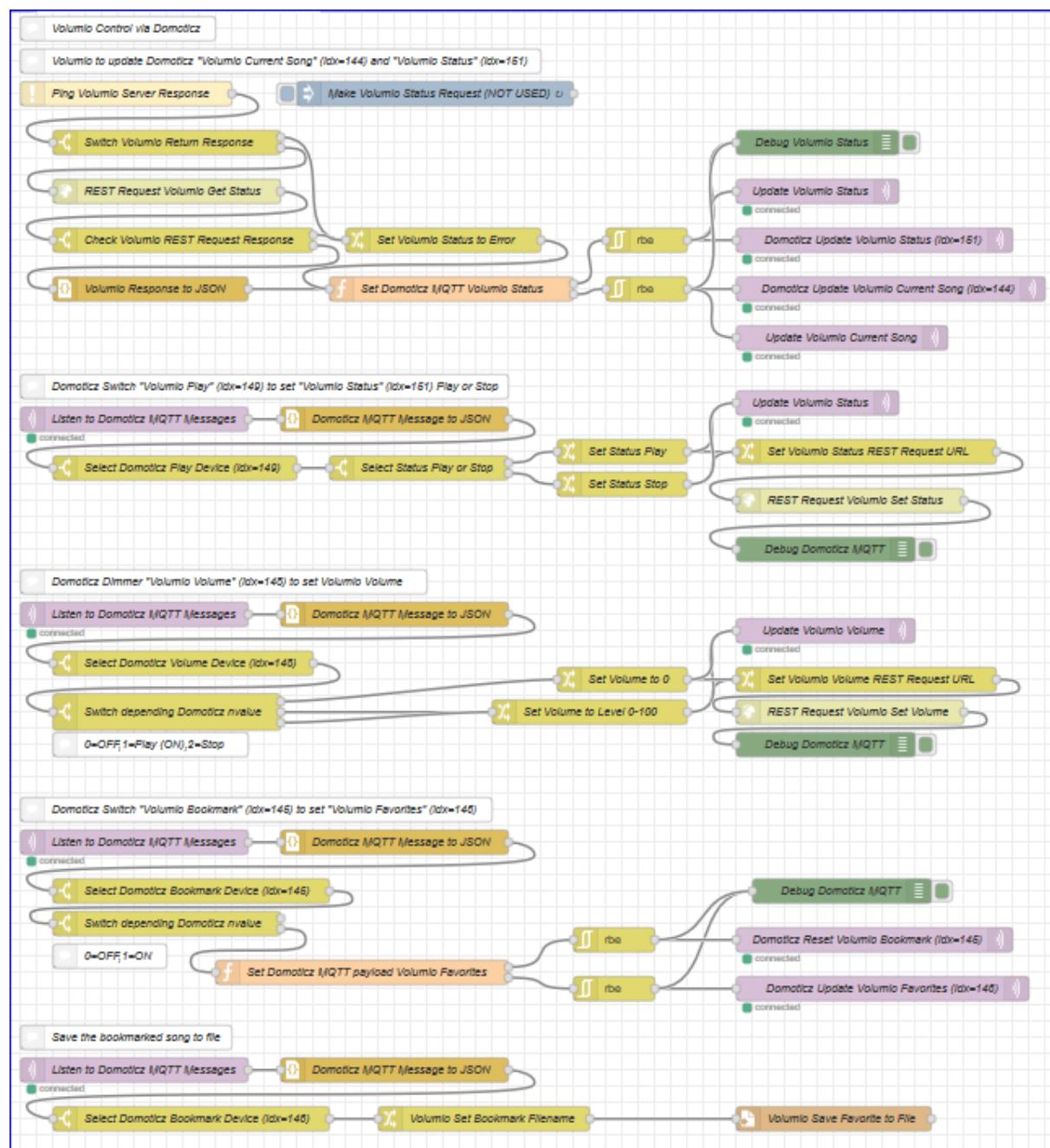
The output of the *Function Node* goes to a *MQTT Out Node*, which connects to the Domoticz MQTT broker (localhost:1883) and publishes the messages.

Node-RED Flows

Volumio Control

The comprehensive Volumio Control flow with subflows:

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

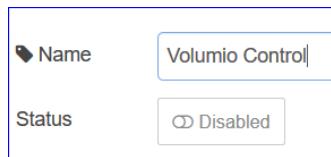


Notes

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

Disable Flow

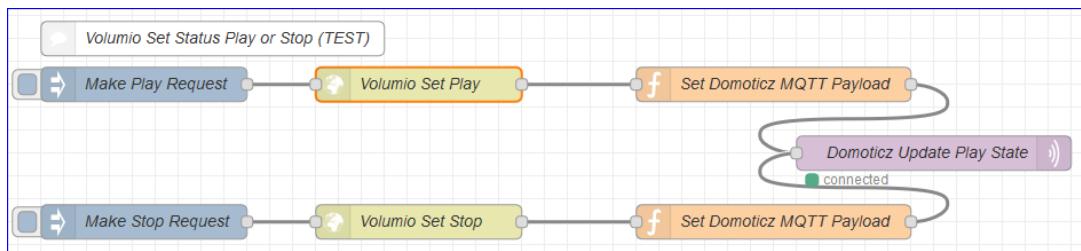
The Node-RED flow *Volumio Control* can be enabled and disabled by clicking on the Tab *Volumio Control*:



If the flow is enabled again, then the Volumio Status is updated depending reachability of the Volumio server (OFF, PLAY or STOP).

Test Example

Use the Inject Node to set the Volumio status to Play or Stop and update the Domoticz device *Volumio Status*.



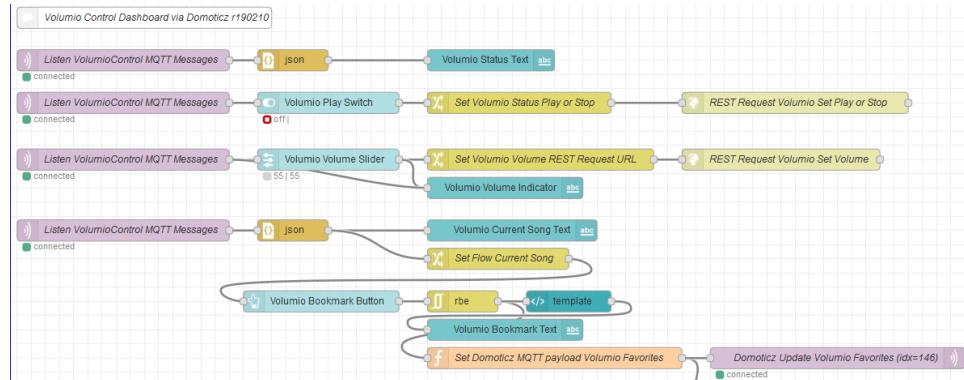
Note

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

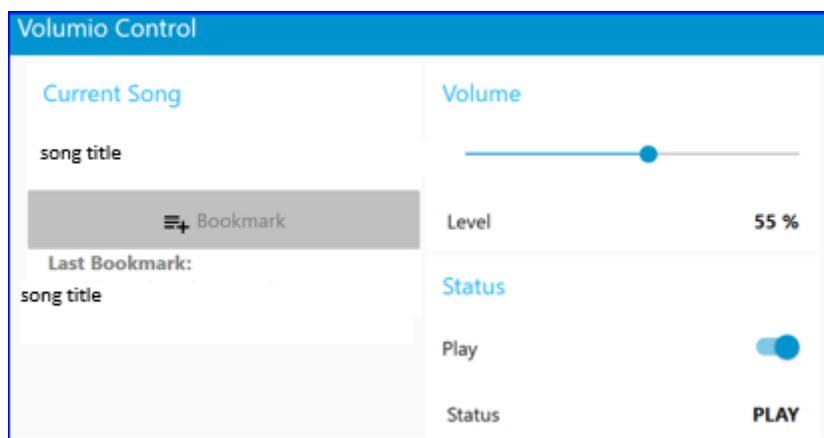
Volumio Control Dashboard

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

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



Displayed in web browser with <http://domoticz-ip-address:1880/ui>



Description (Dasboard UI Node Types)

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

Dashboard Properties

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

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

MQTT Message	Properties
Topic	volumiocontrol/status
Payload Sample	{"command":"udevice","idx":151,"nvalue":1,"svalue":"PLAY"}
Usage	To set the status of Volumio in a Dashboard ui_text node:
Topic	volumiocontrol/currentsong
Payload Sample	{"command":"udevice","idx":144,"svalue":"Current Song Title"}
Usage	To set the text of the current song playing in a Dashboard ui_text node: Value format: {{msg.payload.svalue}}
Topic	volumiocontrol/volume
Payload Sample	NN containing the level
Usage	To set the volume level of the slider and text of the indicator. Value format: {{msg.payload}} %

Favorites List

The bookmarked songs are added to the favorites device *Volumio Favorites* (idx=146) and are listed (log).

The list is used to follow up on favorite song via other media.

Show 25 entries	
Date	
2019-02-10 11:45:05	song title
2019-02-10 11:33:13	song title
2019-02-10 11:18:31	song title
2019-02-10 11:15:55	song title

The favorite songs are stored in the Domoticz database domoticz.db, located in folder /home/pi/domoticz/ on the Raspberry Pi.

The table LightingLog contains the records.

Access via SQLite Tool:

To select the favorite songs, use SQL Query command like

```
SELECT * FROM LightingLog WHERE DeviceRowID=146;
```

This enables to create applications using these records from the database.

<TODO>Brainstorm a Favorite Songs application. Thoughts: 1) Node-RED with the SQLite3 Library to select and list in a customised widget. 2) Lazarus or B4J Desktop application.

Stock Monitor

Purpose

To monitor stock quotes (values):

- Request every 30 minutes stock quotes from [Alpha Vantage](#)
- Display quotes in dedicated Domoticz dashboard (Room Stocks)
- View trends in charts day, month, year
- Notify via alert message and (optional) email on thresholds

Screenshots Stock Monitor with Dashboard Room Stocks

The screenshot displays several components of the Domoticz interface related to the Stock Monitor:

- Utility Sensors:** A list showing two stock sensors:
 - Stock SYMBOL 27.985 EUR
 - Stock SYMBOL 14.275 EUR
- Stock Details:** Two detailed views for the stocks:
 - Stock SYMBOL 27.985 EUR:** Last Seen: 2019-02-14 09:21:04, Type: General, Custom Sensor. Buttons: Log, Edit, Notifications.
 - Stock SYMBOL 14.275 EUR:** Last Seen: 2019-02-14 09:21:09, Type: General, Custom Sensor. Buttons: Log, Edit, Notifications.
- Alert Configuration:** A table for setting alerts:

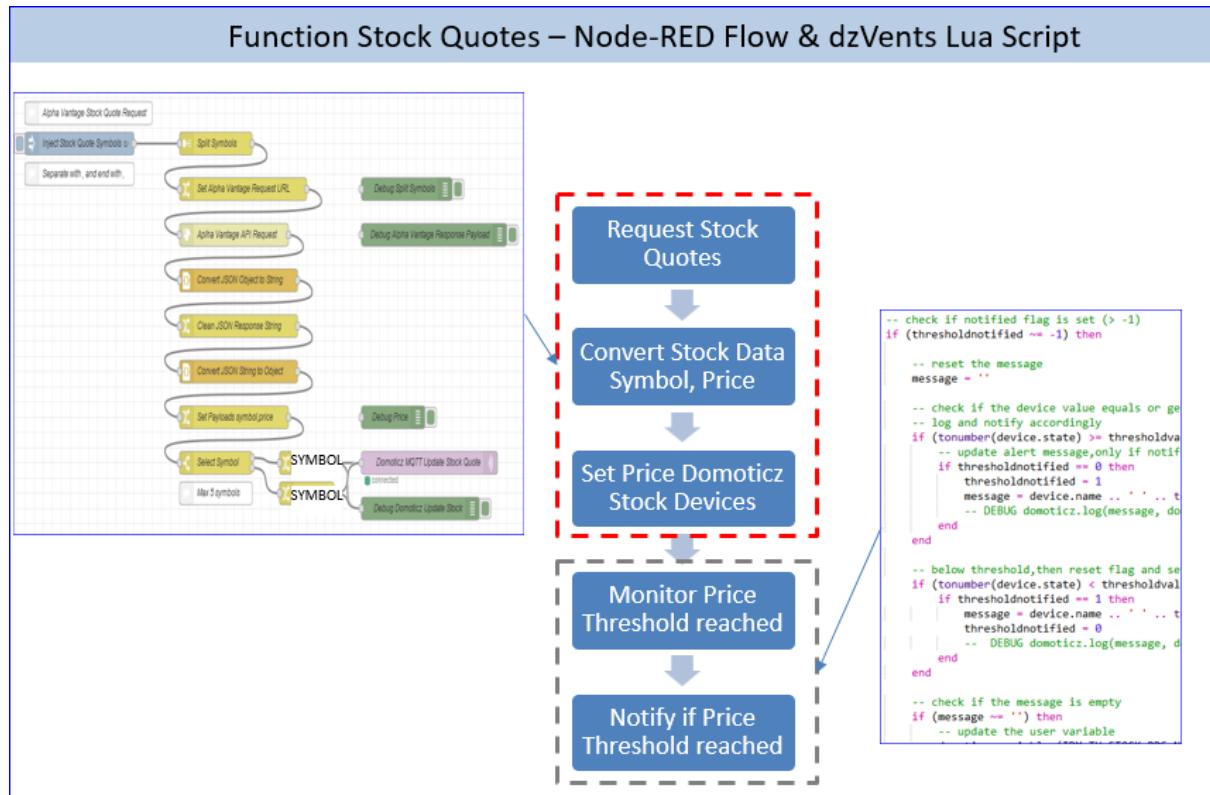
Type	When	Active Systems	Custom Message
Usage	Greater 28	email	Stock Name 28 EUR!
- Day Chart:** A line chart titled "Day" showing the price of the first stock over a 24-hour period. The Y-axis is labeled "EUR" and ranges from 27.7 to 28.1. The X-axis shows times from 12:00 to 18:00. The chart shows a sharp drop from approximately 27.95 to 27.80 around 12:00, followed by a steady recovery and some minor fluctuations towards the end of the day.

Solution

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.

An Alpha Vantage API key is required to place requests (get [here](#)).



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%"
    }
}
```

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 and Percentage_Calendar (Percentage_Min,_Max,_Value for per day).

Roomplan

Created a roomplan *Stocks* with the previous defined devices
(Domoticz Web UI Setup > More Options > Plans > Roomplan).

The roomplan is used as the Domoticz Dashboard (see earlier screenshot) to monitor the stocks.

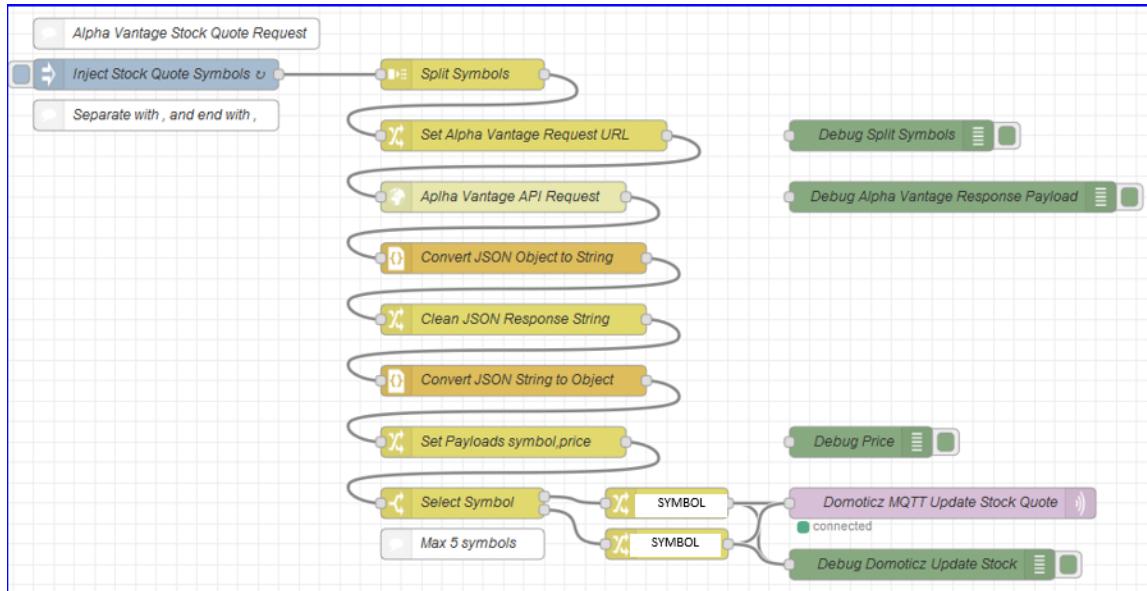
Screenshot Roomplan number 6 named Stocks with 2 devices

Idx	Name
152	Stock SYMBOL
153	Stock SYMBOL

Add more devices as required.

Node-RED Flow

The flow is used to define the Stock Quote Symbols, obtain the data from Alpha Vantage, extract the price and update the Domoticz devices.



Inject Node

The stocks to monitor are defined in the Inject Node, with their symbol as a comma separated string (last symbol MUST end with a comma).

Example for two stocks:

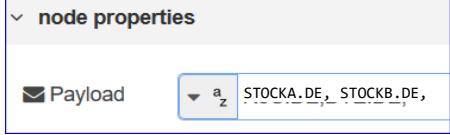
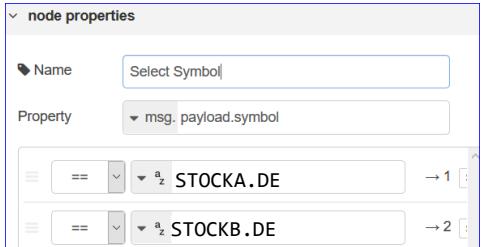
```
STOCKA.DE, STOCKB.DE,
```

Select Symbol Node

The Select Symbol switch node splits the symbols, which require each a change node to define the Domoticz MQTT update command.

Add, change or remove symbols

Change following nodes in the flow:

Inject Node	Property Payload String
	 <pre> node properties Payload: STOCKA.DE, STOCKB.DE, </pre>
Select Symbol Node	Properties
	 <pre> node properties Name: Select Symbol Property: msg.payload.symbol == STOCKA.DE == STOCKB.DE </pre>
Change Node	Properties for each of the Change Nodes for a Stock
	 <pre> node properties Name: STOCKA.DE Rules Set msg.payload to J: {"command": "udevice", "idx": 152, "nvalue": 0, "svalue": payload.price} </pre>

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"}
  
```

Notifications

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

Option Widget

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

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

To change the threshold, modify the rule.

Option dzVents

Notify by using User Variables and a dzVents script to monitor the threshold.

If a threshold is reached or is below the threshold, send a notification and update the alert message.

User Variables

The User Variables are defined in the Domoticz GUI
(Setup >More Options > User variables).

Idx	Variable name	Variable type	Current value
9	TH_STOCK_SYMBOLNOTIFIED	Integer	1
8	TH_STOCK_SYMBOL	Float	27

Name uses syntax: PREFIX_TYPE_SYMBOL_NAME

The TH_STOCK_SYMBOL sets the value of the threshold for the stock.

The TH_STOCK_SYMBOL_NOTIFIED is used to determine if notification should be used (0), if not notified (0) or if notified (1).

The values 0 and 1 are to avoid sending out new messages when the stock value is checked against its threshold. These values are set via the dzVents script *stock_monitor*.

Changing the value of the threshold can be done via Domoticz GUI changing the User Variable or as an alternative via Node-RED Dashboard UI Input Node.

dzVents Script

```
-- 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)

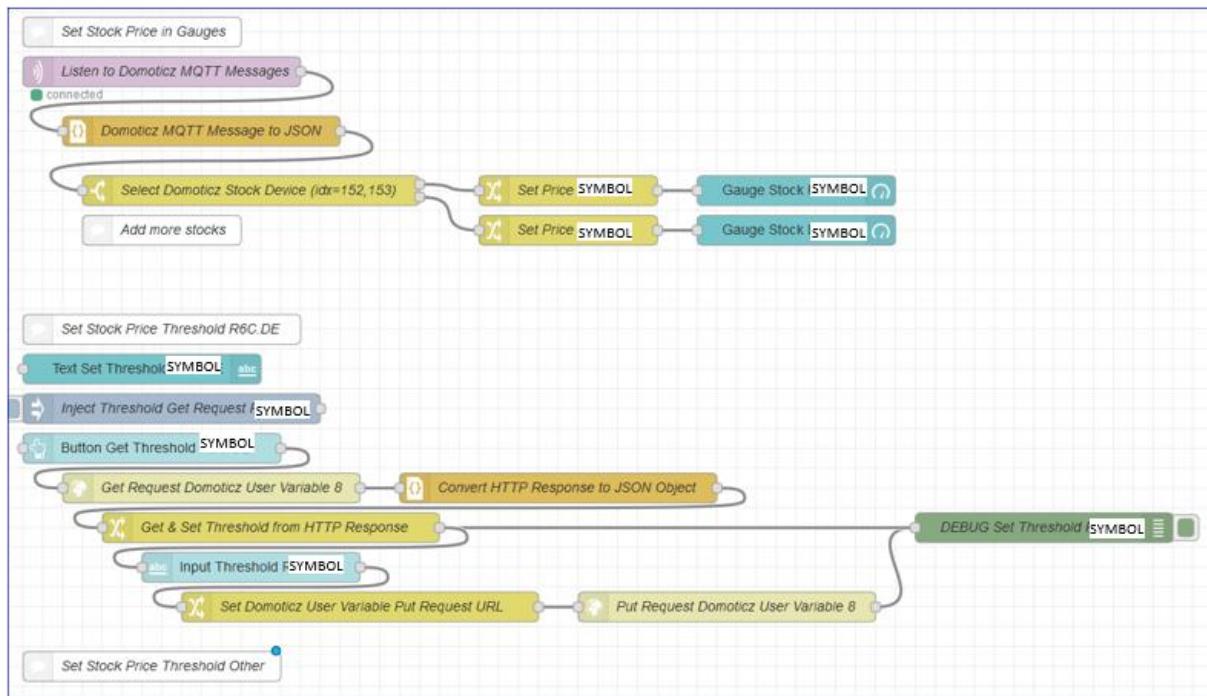
    -- select the device to obtain the thresholdvalue
    if device.idx == IDX_STOCK_NAME then
      thresholdvalue = domoticz.variables(IDX_TH_STOCK_NAME).value
      thresholdnotified = domoticz.variables(IDX_TH_STOCK_NAME_NOTIFIED).value
    domoticz.log('Device ' .. device.name .. ':' .. tostring(thresholdvalue) .. ', ' ..
    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 greater 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
}
```

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.



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
...
```

Ambient Light

Purpose

To measure the Ambient Light in Lux with a BH1759 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 device uses I2C communication.

Two addreses are supported depending the "ADDR" pin connection:

- ADDR to A3 or GND = 0x23
- ADDR to Vin = 0x5C

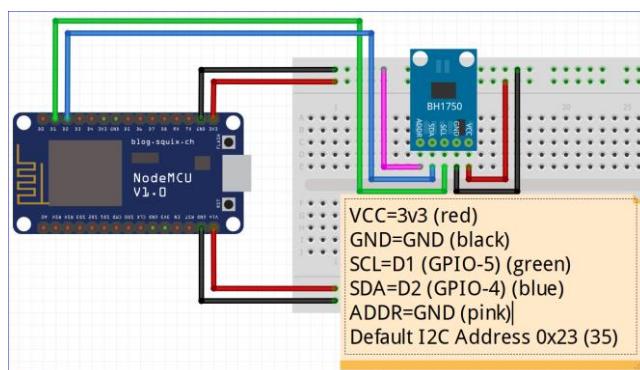
This device uses the default address; ADDR connected to GND.

Parts Needed

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

Wiring

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



Domoticz Device Setup

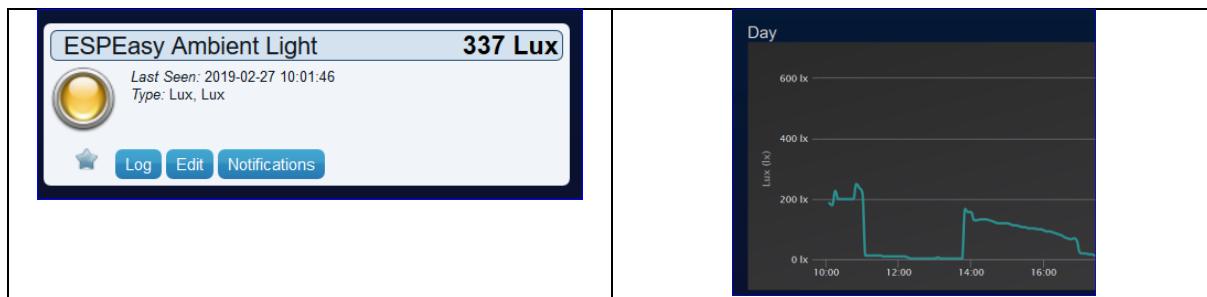
Create a virtual sensor:Sensor Type Lux with name ESPEasy Ambient Light.
After adding, the idx is required for the ESP device settings. Idx=46.

Devices List Entry

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

Widget

The widget and day chart showing actual values.



ESP Easy Device Setup

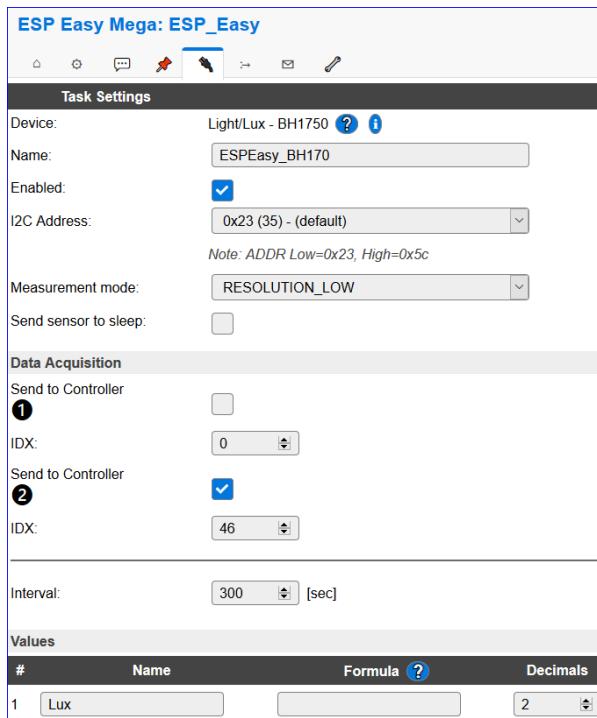
Define the device via web browser by opening <http://esp-device-ip-address> > goto Devices > select Task Edit and configure.

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

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

Domoticz Log Entry Sample

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



ESP Easy Devices List Entry

Edit	4		Light/Lux - BH1750	ESPEasy_BH170			(46)	GPIO-4 GPIO-5	Lux:	290.00
----------------------	---	--	--------------------	---------------	--	--	------	------------------	------	--------

Trigger Switch Light

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

Threshold

Defined as Domoticz user variable: TH_AMBIENT_LIGHT with value integer 100.
(Setup > More Options > User variables)

Idx	Variable name	Variable type	Current value
10	TH_AMBIENT_LIGHT	Integer	100

dzVents Script

Script name: ambient_light_control_espeasy

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

```
--[[[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 level and 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
```

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 Web UI.

Syntax

A syntax is defined as a naming convention for User Variables:

PREFIX_DEVICENAME in UPPERCASE

Prefix

Prefix	Meaning	Example
DEF	Default	DEF_ALERTMSG
IDX	Device Index	IDX_ALERTMSG
TH	Threshold	TH_RPI_HDDUSAGE

Device Name

Use the device name without special characters and blanks.

User Variable List

(Domoticz Web UI Setup > More Options > User variables)

Idx	Variable name	Variable type	Current value
10	TH_AMBIENT_LIGHT	Integer	111
9	TH_STOCK_SYMBOL OTIFIED	Integer	0
8	TH_STOCK_SYMBOL	Float	28
7	DEF_ALERTMSG	String	Reset
6	IDX_ALERTMSG	Integer	55
5	TH_RPI_MEMORYUSAGE	Integer	80
4	TH_RPI_HDDUSAGE	Integer	70
3	TH_RPI_TEMPERATURE	Integer	50

Usage in Scripts

Examples on how to use the User Variables in scripts.

dzVents, Lua

Read more [here](#) how to use with dzVents.

Read Value

```
-- Update the alert message with level.  
function msgbox.alertmsg(domoticz, level, msg)  
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)  
end
```

```
local message = domoticz.variables('DEF_ALERTMSG').value
```

Set Value

```
domoticz.variables('DEF_ALERTMSG').set('Hello World')
```

Usage in Browser Interactive

Examples on how to use the User Variables in scripts.

Read more [here](#) how to use.

Read Value

A value is obtained by using the idx of the user variable.

```
http://rpi-ip-address/json.htm?type=command&param=getuservariable&idx=8
```

Response

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

Set Value

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

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

Response

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

Usage in Node-RED

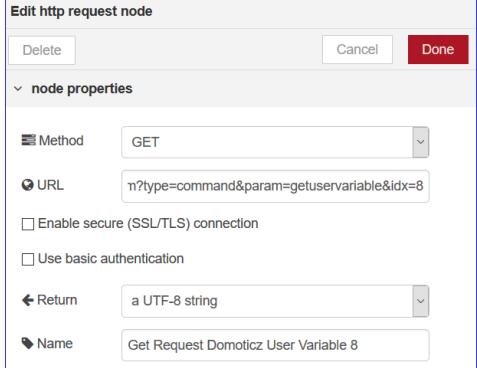
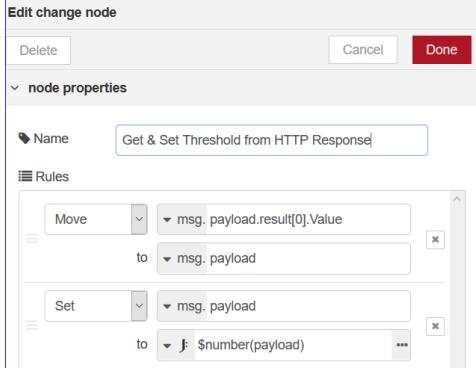
Examples on how to use the User Variables in Node-RED.

For Node-RED, there are several possibilities.

This example uses HTP Response Nodes.

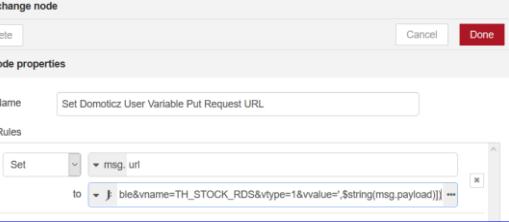
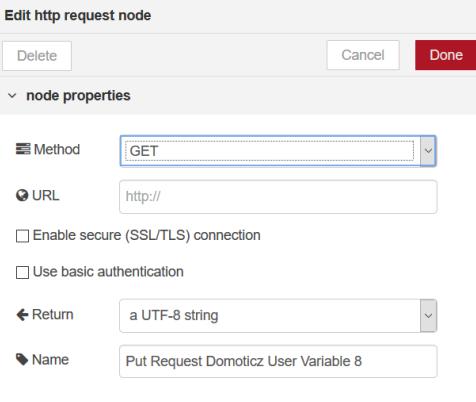
Read Value

Same request as previous used in Browser request.

 <p>Use a JSON node to convert the output to JSON Object used by the Change Node.</p>	 <p>Two rules, first move the property Value from the JSON response to the msg.payload, then use that string to get converted to a number using JSONata function.</p>
--	---

Set Value

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

 <p>Set the property msg.url using JSONata function join (JSON/API + Value)</p>	 <p>The request uses the previous defined msg.url to send to Domoticz</p>
--	---

Usage in Node-RED Dashboard

The Node-RED Dashboard has input widgets.

This makes it possible to build a web page (with several tabs and groups) to configure User Variables.

Next an example setting the value of a User Variable.

Ambient Light Threshold

Set the threshold of the Ambient Light Sensor. If below threshold, then a light is switched on. The threshold is used by a dzVents script to control the light (see [Ambient Light](#)).

Enhancement Idea

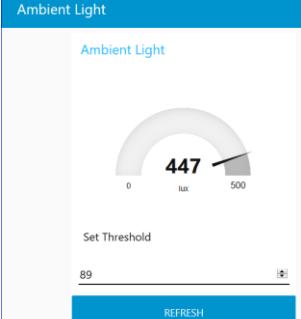
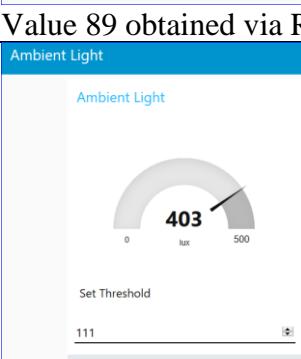
Define a User Variable (like TH_AMBIENT_LIGHT_SWITCH) holding the idx of the light (better switch) to be turned ON when the value is below threshold. A ui_dropdown list could be defined to select the light (switch) and update the User Variable.

Node-RED Dashboard

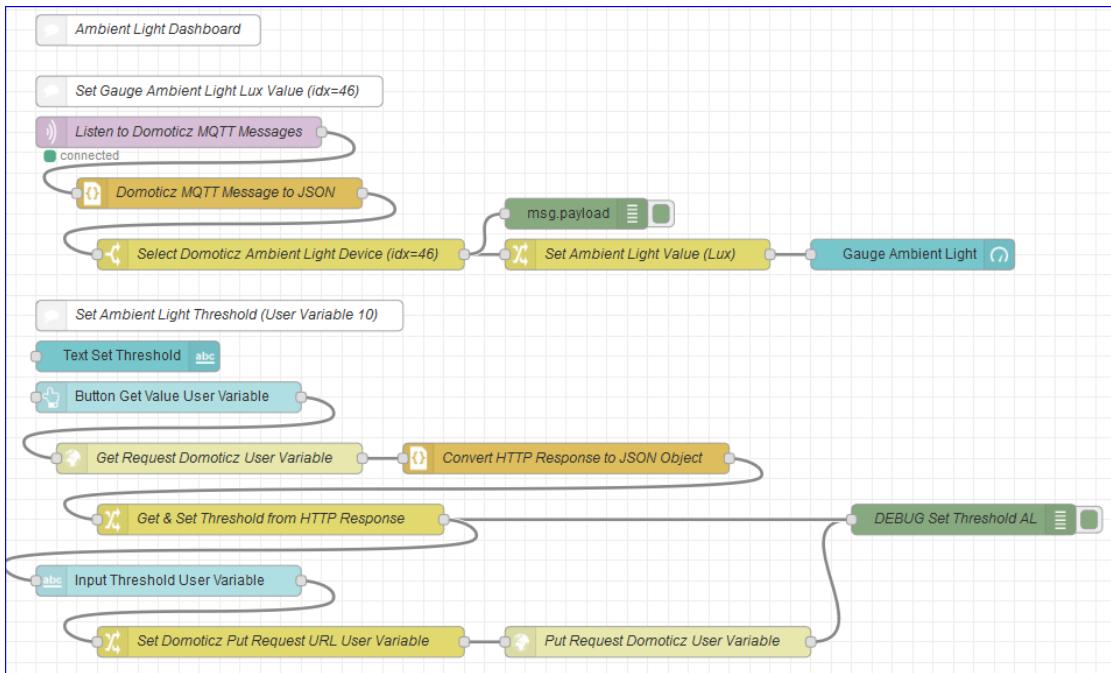
A ui_gauge node displays the actual Ambient Light Lux value, which is obtained from MQTT payload of the Ambient Light device. Example from which property "svalue1" is used by the gauge:

```
{"Battery":255, "RSSI":5, "description":"The ambient light intensity in Lux using a BH1750 sensor.", "dtype":"Lux", "id":82045, "idx":46, "name":"ESPEasy Ambient Light", "nvalue":0, "stype":"Lux", "svalue1":3050.00, "unit":1}
```

The threshold is set through the ui_input node, which is configured for number input. Entering the new number and pressing Enter, the value is updated in Domoticz.

 <p>Value 89 obtained via Refresh</p>	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Idx</th> <th>Variable name</th> <th>Variable type</th> <th>Current value</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>TH_AMBIENT_LIGHT</td> <td>Integer</td> <td>89</td> </tr> </tbody> </table> <p>Domoticz User Variable List (extract for idx=10)</p>	Idx	Variable name	Variable type	Current value	10	TH_AMBIENT_LIGHT	Integer	89
Idx	Variable name	Variable type	Current value						
10	TH_AMBIENT_LIGHT	Integer	89						
 <p>Changed to 111 and pressed enter</p>	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Idx</th> <th>Variable name</th> <th>Variable type</th> <th>Current value</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>TH_AMBIENT_LIGHT</td> <td>Integer</td> <td>111</td> </tr> </tbody> </table> <p>Refresh the page in the browser, shows the updated value for the User Variable</p>	Idx	Variable name	Variable type	Current value	10	TH_AMBIENT_LIGHT	Integer	111
Idx	Variable name	Variable type	Current value						
10	TH_AMBIENT_LIGHT	Integer	111						

Node-RED Flow



Overview API Requests

Defined User Variable

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

HTTP Request Get User Variable

<http://localhost:8080/json.htm?type=command¶m=getuservariable&idx=10>

HTTP Response (JSON Format)

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

HTTP Request Update User Variable

http://localhost:8080/json.htm?type=command¶m=updateuservariable&vname=TH_AMBIENT_LIGHT&vtype=0&vvalue=NNNN

Note

vtype is 0 which is an Integer.

HTTP Response (JSON Format)

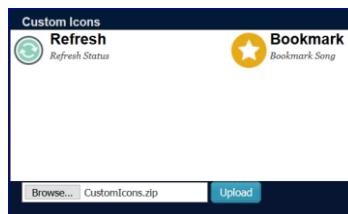
```
{
  "status": "OK",
  "title": "UpdateUserVariable"
}
```

Custom Icons

Custom icons are used for several devices.

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

Browse for the file *CustomIcons.zip* and upload. After upload, the custom icons are shown and are listed when editing the icon on a device (via Widget > Edit).



Each icon has 3 files:

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

The archive *CustomIcons.zip* contains the files.

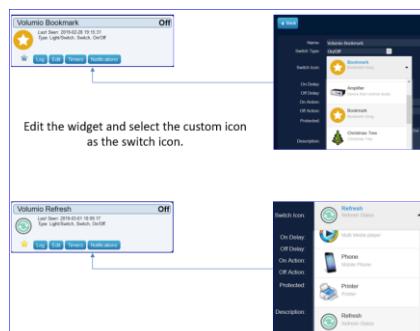
- icons.txt
- Refresh.png
- Refresh48_On.png
- Refresh48_Off
- Bookmark.png
- Bookmark48_On.png
- Bookmark48_Off.png

The content of the file icons.txt for the two icons:

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

Ensure the first two entries match the icon file name!

The icons are added to the widgets by selecting from the icon list:



Explore

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

Events System

Purpose

To explore the Domoticz Events System and the Event Editor (see 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.

Important to read [this](#) first.

Note

The Lua editor is context sensitive, prompt with auto-complete options and show common errors to help with debugging. The editor theme can be changed by pressing the control and comma keys at the same time. Domoticz will save the scheme when events are saved.

Python

In favour to program using Python instead Lua, **BUT** the new **dzVents** is so great, that for event scripting will use dzVents.

For building Domoticz PlugIns will use Python.

Note

The Python Event System uses Python3 (same as Python Plugin System). Events are enabled along with plugins, it's no longer possible to enable one without the other.

Event Execution Order

The order of events being executed, is:

1. file scripts - stored in the folder /home/pi/domoticz/scripts/...
2. database scripts - stored in tables [EventMaster] and [EventRules]
3. on/off action script - explore usage & storage

Important (taken from 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

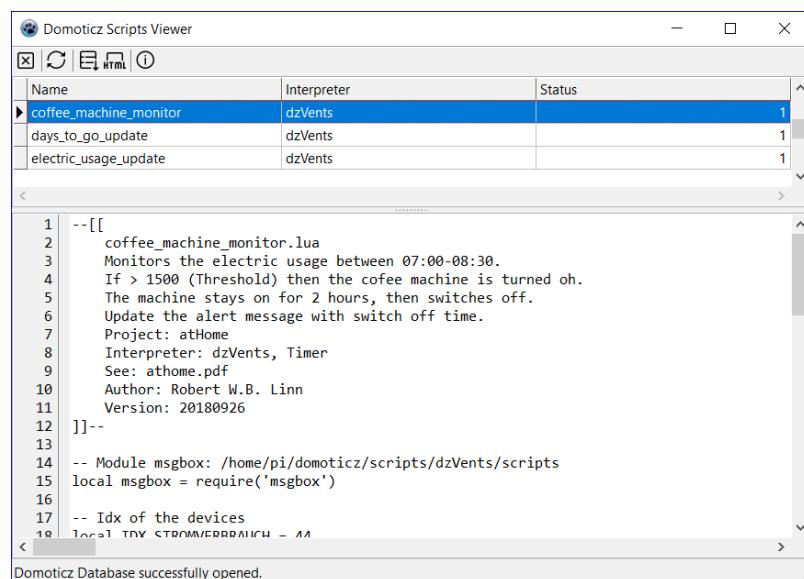
The events which are created using the Domoticz Event Editor, are stored in tables [EventMaster] and [EventRules]

```
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.



Name	Interpreter	Status
coffee_machine_monitor	dzVents	1
days_to_go_update	dzVents	1
electric_usage_update	dzVents	1

```

1 --[[ coffee_machine_monitor.lua
2 Monitors the electric usage between 07:00-08:30.
3 If > 1500 (Threshold) then the coffee machine is turned on.
4 The machine stays on for 2 hours, then switches off.
5 Update the alert message with switch off time.
6 Project: aHome
7 Interpreter: dzVents, Timer
8 See: ahome.pdf
9 Author: Robert W.B. Linn
10 Version: 20180926
11 ]]
12 --
13 --
14 -- Module messagebox: /home/pi/domoticz/scripts/dzVents/scripts
15 local messagebox = require('messagebox')
16 
17 -- Idx of the devices
18 local dev_idx = 1

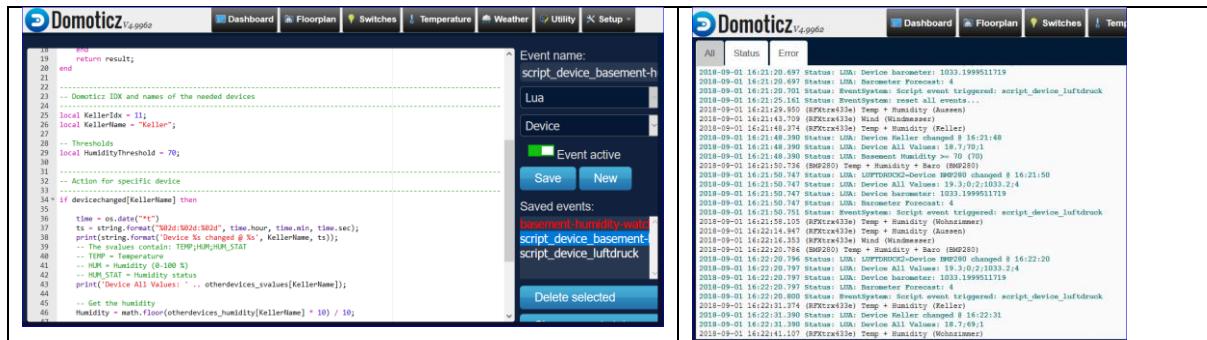
```

Domoticz Database successfully opened.

Event Development

Whilst creating scripts using the Event Editor, two Browser Tabs are open:

- Domoticz Event Editor – to create/modify scripts using Ace Editor, to activate/deactivate a script,
 - Domoticz Log (Setup > Log > Tab All) – to check script OK after save.



Event Sample Basement Humidity Monitor

Purpose

To explore the various interpreter using the same event to monitor a device:

Device: Keller (idx=11)

Property: humidity

Rule:

If the Basement Humidity is ≥ 70 (threshold), write a message to the Domoticz Log.

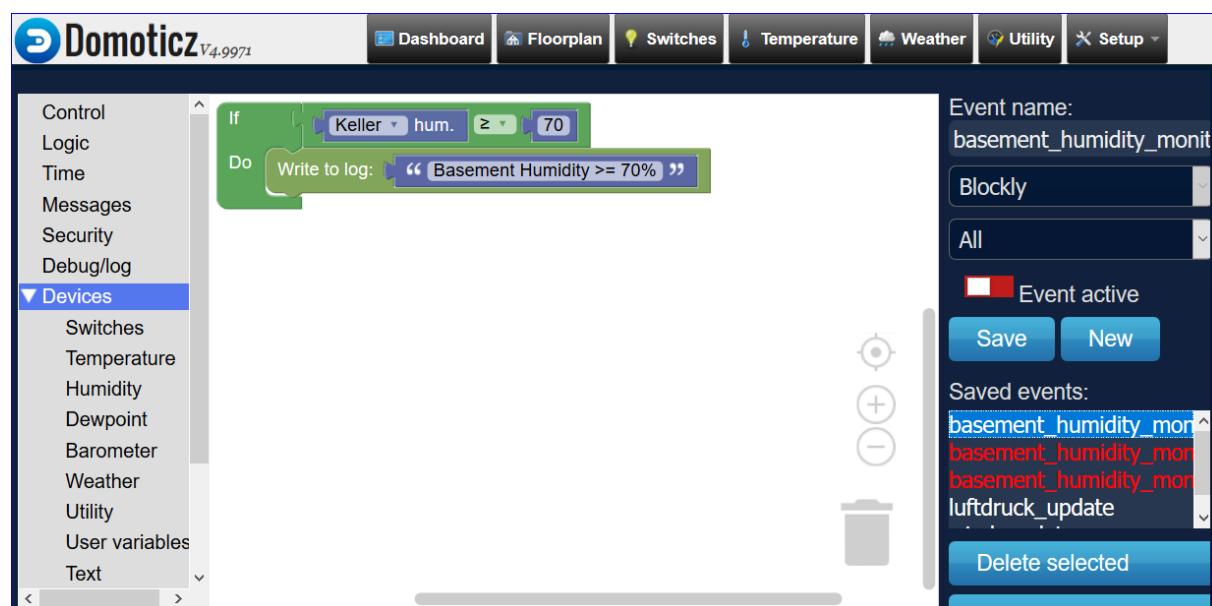
Note

For the sample scripts, added to the Event Names, a suffix as an interpreter indicator:

_d (Blockly), _l (Lua), _d (dzVents), _p (Python).

Blockly

Event Properties: basement_humidity_monitor_b, Blockly, All.



Domoticz Log Entry

```

2018-09-01 10:32:04.478 EventSystem: Event triggered: basement_humidity_monitor_b
2018-09-01 10:32:04.478 Status: Basement Humidity >= 70%
```

Lua

Event Properties: basement_humidity_monitor_1, Lua, Device.

```

1 -- basement_humidity_monitor_1
2 -----
3 -- Monitor the basement humidity if exceeds threshold and write to Domoticz log.
4 -- @project: mysmarthome
5 -- @interpreter: Lua, Device
6 -- @author: Robert W.B. Linn
7 -- @since: 20180901
8 -- @version: 1.0
9 -- @see: mysmarthome.doc
10 -----
11 commandArray = {}
12
13
14 -- Split a string by delimiter
15 function split(s, delimiter)
16     result = {}
17     for match in (s..delimiter):gmatch("(.-)"..delimiter) do
18         table.insert(result, match);
19     end
20     return result;
21 end
22
23
24 -- Domoticz IDX and names of the needed devices
25 -----
26 local KellerIdx = 11;
27 local KellerName = "Keller";
28
29 -- Thresholds

```

Script

```

commandArray = {}

-- Domoticz IDX and names of the needed devices
local KellerIdx = 11;
local KellerName = "Keller";

local HumidityThreshold = 70;

if devicechanged[KellerName] then
    time = os.date("*t")
    ts = string.format("%02d:%02d:%02d", time.hour, time.min, time.sec);
    print(string.format('Device %s changed @ %s', KellerName, ts));
    print('Device All Values: ' .. otherdevices_svalues[KellerName]);
    -- The svalues contain: TEMP;HUM;HUM_STAT
    -- TEMP = Temperature
    -- HUM = Humidity (0-100 %)
    -- HUM_STAT = Humidity status
    -- Get the humidity
    Humidity = math.floor(otherdevices_humidity[KellerName] * 10) / 10;

    -- Check if the humidity is equal or above threshold - notify
    if Humidity >= HumidityThreshold then
        print('Basement Humidity >= ' .. HumidityThreshold .. ' (' .. Humidity .. ')');
    end
end

return commandArray

```

Domoticz Log Entry

Sample Script – Error

```
2018-09-01 13:20:29.442 Status: LUA: Device Keller changed @ 13:20:29
2018-09-01 13:20:29.442 Status: LUA: Device All Values: 18.7;74;3
2018-09-01 13:20:29.442 Error: EventSystem: in basement-humiditymonitor_1:
[string "-- script_device_basement-humidity..."]:41: bad argument #2 to 'format'
(number expected, got nil)
```

Caused by using wrong variable name, i.e. humidity instead of Humidity.

Sample Script – OK

```
2018-09-01 13:43:25.433 Status: LUA: Device Keller changed @ 13:43:25
2018-09-01 13:43:25.433 Status: LUA: Device All Values: 18.6;75;3
2018-09-01 13:43:25.433 Status: LUA: Basement Humidity >= 70 (75)
```

Lua dzVents

Event Properties: basement_humidity_monitor_d, dzVents, Device.

```

1 --~/domoticz/scripts/lua/basement_humidity_monitor_d.lua
2 -----
3 -- Monitor the basement humidity if exceeds threshold and write to Domoticz log.
4 -- project: mysmarthome
5 -- @interpreter: dzVents, Device
6 -- @see: readme.txt
7 -- @author: Robert W.B. Linn
8 -- @version: 20180902
9 -----
10 -- Thresholds
11 -- Set the threshold to monitor the humidity (%RH)
12 local HumidityThreshold = 70;
13
14 -- dzVent
15 return {
16     on = {
17         devices = {
18             'Keller'
19         }
20     },
21     execute = function(domoticz, device)
22         if (device.name == 'Keller' and device.humidity >= HumidityThreshold) then
23             domoticz.log('Basement Humidity >= '..HumidityThreshold.. '(' .. device.humidity .. ')', domoticz.LOG_INFO)
24             -- domoticz.notify('Fire', 'The room is on fire', domoticz.PRIORITY_EMERGENCY)
25         end
26     end
27 }
28 }
29 <

```

Script

```

local HumidityThreshold = 70;
return {
    on = {
        devices = {
            'Keller'
        }
    },
    execute = function(domoticz, device)
        if (device.name == 'Keller' and device.humidity >= HumidityThreshold) then
            domoticz.log('Basement Humidity >= '..HumidityThreshold.. '(' .. device.humidity .. ')', domoticz.LOG_INFO)
        end
    end
}

```

The script is stored in:

```
/home/pi/domoticz/scripts/dzVents/generated_scripts/basement_humidity_monitor_d.lua
```

With owner root.

Domoticz Log Entry

Sample Script – Error

```
2018-09-01 19:05:55.584 Status: dzVents: Error (2.4.7): error loading module 'basement_humidity_monitor_d' from file '/home/pi/domoticz/scripts/dzVents/generated_scripts/basement_humidity_monitor_d.lua':
```

```
2018-09-01 19:05:55.584 .../generated_scripts/ basement_humidity_monitor_d.lua:12:  
'then' expected near ')'
```

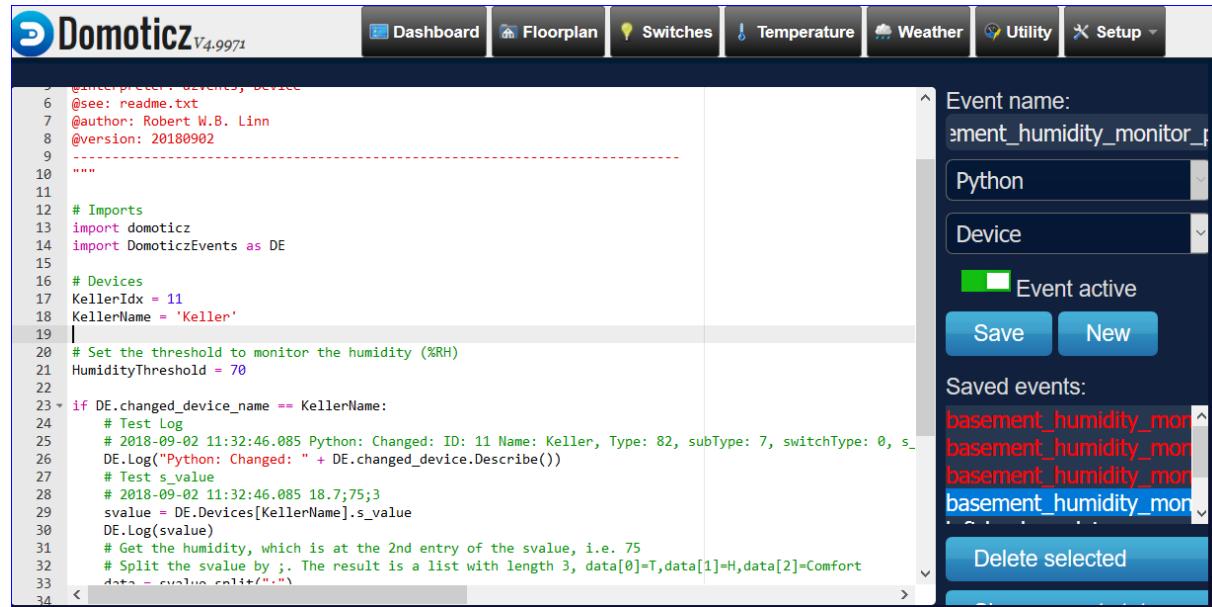
Caused by using double)) prior then.

Sample Script – OK

```
2018-09-01 19:09:42.474 Status: dzVents: Write file:  
/home/pi/domoticz/scripts/dzVents/generated_scripts/  
basement_humidity_monitor_d.lua  
2018-09-01 19:11:39.449 Status: dzVents: Info: Handling events for: "Keller",  
value: "18.6;70;1"  
2018-09-01 19:11:39.449 Status: dzVents: Info: ----- Start internal script:  
script_device_basement_humidity_2: Device: "Keller (RFXtrx433e)", Index: 11  
2018-09-01 19:11:39.449 Status: dzVents: Info: Basement Humidity >= 70 (70)  
2018-09-01 19:11:39.449 Status: dzVents: Info: ----- Finished  
basement_humidity_monitor_d
```

Python

Event Properties: basement_humidity_monitor_p, Python, Device.



Script

```

"""
Script: basement_humidity_monitor_p
project: AtHome
Monitor the basement humidity if exceeds threshold and write to Domoticz log.
@interpreter: Python, Device
@see: readme.txt
@author: Robert W.B. Linn
@version: 20180902
-----
"""

# Imports
import domoticz
import DomoticzEvents as DE

# Devices
KellerIdx = 11
KellerName = 'Keller'

# Set the threshold to monitor the humidity (%RH)
HumidityThreshold = 70

if DE.changed_device_name == KellerName:
    # Test Log
    # 2018-09-02 11:32:46.085 Python: Changed: ID: 11 Name: Keller, Type: 82, subType: 7,
    switchType: 0, s_value: 18.7;75;3, n_value: 0, n_value_string: 18.7;75;3,
    last_update_string: 2018-09-02 11:32:46
    DE.Log("Python: Changed: " + DE.changed_device.Describe())
    # Test s_value
    # 2018-09-02 11:32:46.085 18.7;75;3
    svalue = DE.Devices[KellerName].s_value
    DE.Log(svalue)
    # Get the humidity, which is at the 2nd entry of the svalue, i.e. 75
    # Split the svalue by ;. The result is a list with length 3,
    data[0]=T,data[1]=H,data[2]=Comfort
    data = svalue.split(";");

```

```

# Log all 3 entries
# for temp in data:
#     DE.Log(temp)
# Get the :
if len(data) == 3:
    Humidity = int(data[1])
    if Humidity >= HumidityThreshold:
        # Log that the humidity is above threshold. The log string uses new string
formatting
        DE.Log('Basement Humidity >= {}%RH ({}%RH)'.format(HumidityThreshold,
Humidity))

```

Domoticz Log Entry

Sample Script – Error

```

...
2018-09-02 11:58:34.884 Status: EventSystem: Script event triggered:
luftdruck_update
2018-09-02 11:58:34.107 Error: EventSystem: Failed to execute python event script
"basement_humidity_monitor_p"
2018-09-02 11:58:34.108 Error: EventSystem: Traceback (most recent call last):
2018-09-02 11:58:34.108 Error: EventSystem: File "<string>", line 41, in <module>
2018-09-02 11:58:34.108 Error: EventSystem: IndexError: tuple index out of range

```

Caused by using wrong string format tuples, i.e. {1} ({2}) instead of 0,1 or leave blank {} ({}).

Sample Script – OK

```

2018-09-02 12:01:26.253 Python: Changed: ID: 11 Name: Keller, Type: 82, subType:
7, switchType: 0, s_value: 18.7;74;3, n_value: 0, n_value_string: 18.7;74;3,
last_update_string: 2018-09-02 12:01:26
2018-09-02 12:01:26.253 18.7;74;3
2018-09-02 12:01:26.253 Basement Humidity >= 70%RH (74%RH)

```

Python with Device Update

Extending the previous Python script

Update the Virtual Sensor “Info Message” (idx=52) (Type: General, Text) with the message logged, if the humidity exceeds the threshold.

52	VirtualSensors	00082051	1	Info Message	General	Text	Basement Humidity >= 70%RH (78%RH)
----	----------------	----------	---	--------------	---------	------	------------------------------------

Result



Event Properties: basement_humidity_monitor2_p, Python, Device.

Script

```
"""
script: basement_humidity_monitor2_p
project: AtHome
Monitor the basement humidity (device=Keller) if exceeds threshold, write to Domoticz log
and update the virtual sensor text (device=Info Message)
@interpreter: Python, Device
@see: readme.txt
@author: Robert W.B. Linn
@version: 20180902
-----
"""

# Imports
import domoticz
import DomoticzEvents as DE
from urllib.request import Request, urlopen
from urllib.error import URLError, HTTPError

# Define the domoticz server ip with port!
domoticzserver="localhost:8080"

# Devices
KellerIdx = 11
KellerName = 'Keller'
TextInfoIdx = 52
TextInfoName = 'Info Message'

# Set the threshold to monitor the humidity (%RH)
HumidityThreshold = 70

def domoticzrequest (url):
    """Send update request to domoticz
    :url: Domoticz JSON/API url
    :result: Result is a json object
    """
    req = Request(url)
    try:
        response = urlopen(req)
    except HTTPError as e:
        print("HTTP Error:", e)
        return None
    else:
        return response.read().decode('utf-8')
```

```

        # 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+H
umidity+%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)
--

dzVents

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

As a general point, be very secure in writing the scripts, also when making changes, i.e. ensure all variables have the same name, do not mix local vars with the same name etc.

Device Properties

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

Script

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

```
-- Idx of the devices
local IDXMEMORYUSAGE = 1;
local IDXHDDUSAGE = 3;
local IDXTEMPERATURE = 4;
local IDXCONTROLMSG = 52;

return {
  on = {
    devices = {
      IDXMEMORYUSAGE,
      IDXHDDUSAGE,
      IDXTEMPERATURE
    }
  },
  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:

<pre> 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 </pre>	<pre> 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 </pre>	<pre> 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 </pre>
---	--	---

External Modules

Purpose

To use external modules in dzVents scripts, which are created with the Events editor.

Important

In the meantime, these modules are replaced by a dzVents script *global_data* (containing shared helper functions), maintained using the internal event editor.

Example

Update the virtual device named “Alert Message” (Type General, SubType Alert, Idx=55) by using a function defined in external module **msgbox.lua**.

```

local message = device.name .. ' switched OFF ' -- .. msg.isnowdatetime(domoticz)
msgbox.alertmsg(domoticz, domoticz.ALERTLEVEL_GREEN, message)

```

Module Access

The module “msgbox.lua” MUST be in folder:

```
/home/pi/domoticz/scripts/dzVents/scripts
```

To access functions from the module, use the syntax **msgbox.functionname**, when the module is defined like:

```
local messagebox = require('msgbox')
```

Example (screenshot from WinSCP)

/home/pi/domoticz/scripts/dzVents/scripts/*.*			
Name	Size	Changed	
..		08.10.2017 17:17:48	
msgbox.lua	2 KB	06.09.2018 13:15:11	
README.md	1 KB	29.08.2017 16:35:10	

Module Script

The script defines date & time functions and functions to update a text device & alert device. The idx of the text device is set in the external module, for the alert device as a user variable. This is to show both ways – best would be to have both device idx as user variables to be flexible in assigning devices.

A domoticz object is required to access properties & methods.

```
msgbox.lua: /home/pi/domoticz/scripts/dzVents/scripts
```

```
-- msgbox.lua
-- Update the virtual devices to display a control (Text Sensor) or alert message
-- (Alert Sensor).
-- Requires a user variable IDX_ALERTMSG for the Alert Sensor
-- Robert W.B. Linn
-- 20180906

local msgbox = {};

local IDX_CONTROLMSG = 52

-- Get the current date & time from the domoticz instance with time object
-- Return datetime now, i.e. 2018-09-06 09:09:00
function msgbox.isnowdatetime(domoticz)
    return domoticz.time.rawDate .. ' ' .. domoticz.time.rawTime
end

-- Get the current date from the domoticz instance with time object
-- Return date now, i.e. 2018-09-06
function msgbox.isnowdate(domoticz)
    return domoticz.time.rawDate
end

-- Get the current time from the domoticz instance with time object
-- Return time now, i.e. 09:09:00
function msgbox.isnowtime(domoticz)
    return domoticz.time.rawTime
end

-- Update the alert message with level.
function msgbox.alertmsg(domoticz, level, msg)
    domoticz.devices(domoticz.variables('IDX_ALERTMSG').value).updateAlertSensor(level, msg)
end

-- Update the control message.
function msgbox.controlmsg(domoticz, msg)
    domoticz.devices(IDX_CONTROLMSG).updateText(msg)
end

return msgbox
```

dzVents Script

The script hue_control2.lua, created with the 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 function used by the dzVents scripts.

The functions are defined, using the internal events editor, in the file global_data.lua.

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)
```

API Interaction

Purpose

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

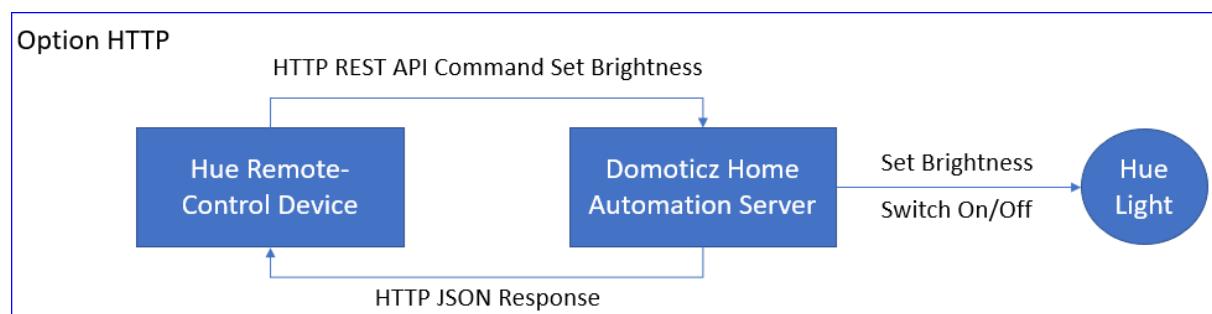
This section does not go into the depth – just a small example on setting the brightness of a Hue Light via a remote-control device, which could be a micro-controller like an ESP8266 or ESP32 or a Raspberry Pi Zero W.

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

Option HTTP

Domoticz allows you to interact with all devices using HTTP REST API Commands.
This means, it is possible to sent commands to the Domoticz server to act upon and receive a response back.

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



Note

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

Example HTTP REST API command issued via web browser

Goal

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

HTTP URL

<http://192.168.1.60:8080/json.htm?type=command¶m=switchlight&idx=118&switchcmd=Set%20Level&level=50>

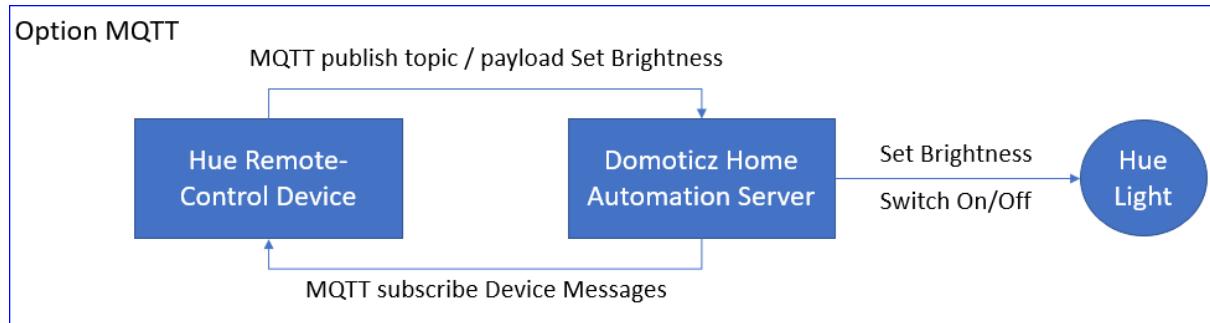
HTTP JSON Response

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

Option MQTT

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

Read more in chapter [MQTT](#).



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

Communication from and to Domoticz works via JSON.

Default MQTT topics of the Domoticz for incoming (domoticz/in) and outgoing (domoticz/out) messages.

To set the brightness of the Hue Light, a message must be published from the Hue Remote-Control to Domoticz.

Example MQTT Messages publish & subscribe

Goal

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

Publish

The MQTT message to publish requires a topic and payload.

Topic:

domoticz/in

Payload:

```
{"command": "switchlight", "idx": 118, "switchcmd": "Set Level", "level": 100}
```

Note

Lookup for the payload syntax in the API reference. The API reference uses HTTP syntax which is used for MQTT as well.

Again, the example:

HTTP: type=command¶m=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 responds. 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 REST 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.

MQTT

Purpose

To enable subscribing and publishing MQTT messages ([see Wiki](#)).

Ensure also to lookup the reference [Domoticz API/JSON Url's](#).

Install

Raspberry Pi

Install [mosquitto](#) on the Raspberry Pi.

Open Terminal and run

```
sudo apt-get install mosquitto mosquitto-clients
```

Check if mosquitto is running:

```
mosquitto -h
```

Output

```
mosquitto version 1.4.10 (build date Fri, 22 Dec 2017 08:19:25 +0000)
mosquitto is an MQTT v3.1 broker.
```

MQTT Hints

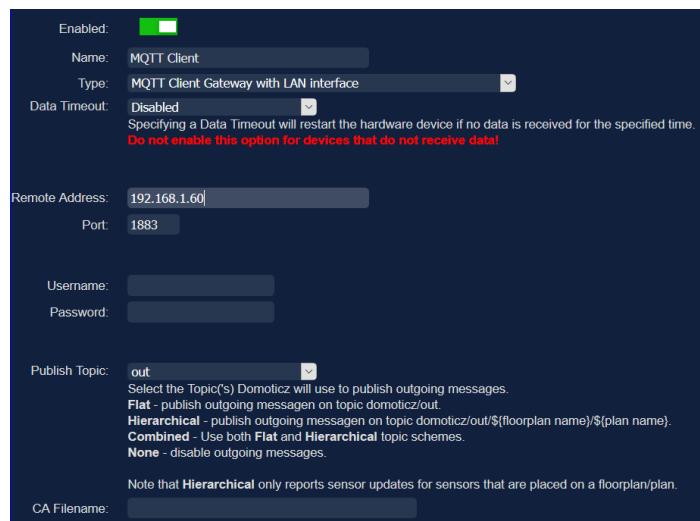
Add the host to the mosquitto command:

```
-h localhost
```

Domoticz

Domoticz Add Hardware: Domoticz Web UI > Setup > Hardware and add the “MQTT Client Gateway with LAN Interface”.

Property	Value
Type	MQTT Client Gateway with LAN Interface
Name	MQTT
Remote Address	rpi-domoticz-ip
Port	1883
Publish Topic	Out
Username	Empty
Password	Empty



Check the Domoticz Log: Domoticz Web UI > Setup > Log

```
2018-08-27 12:21:12.126 Status: MQTT: Connecting to rpi-domoticz-ip:1883
2018-08-27 12:21:12.228 Status: MQTT: connected to: rpi-domoticz-ip:1883
2018-08-27 12:21:12.328 Status: MQTT: Subscribed
```

Domoticz Topics

Subscribe to Topics from Domoticz	domoticz/out
Publish a Topic to Domoticz	domoticz/in

Mosquitto

Test MQTT messaging using mosquitto commands from a terminal.

Subscribe

Command

Subscribe to all MQTT messages published by Domoticz (topic: domoticz/out):

```
mosquitto_sub -t 'domoticz/out'
```

Sample Messages

Some messages published by Domoticz in JSON format.

The source is from the devices provided by the Hardware Device Motherboard (idx=1 and 2):

	2	Motherboard	0001	1	Internal Temperature	Temp	LaCrosse TX3	42.4 C
	1	Motherboard	0000044D	1	CPU_Usage	General	Percentage	0.19%

For each of the messages the topic is *domoticz/out* and the payload is a JSON string.

To get a value from a key, the JSON string needs to be parsed.

CPU Usage

```
{  
    "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "General", "id" : "0000044D",  
    "idx" : 1, "name" : "CPU_Usage", "nvalue" : 0, "stype" : "Percentage", "svalue1" : "0.19",  
    "unit" : 1  
}
```

CPU Temperature

```
{  
    "Battery" : 255, "RSSI" : 12, "description" : "", "dtype" : "Temp", "id" : "1",  
    "idx" : 2, "name" : "Internal Temperature", "nvalue" : 0, "stype" : "LaCrosse TX3",  
    "svalue1" : "42.4", "unit" : 1  
}
```

Publish

Let's change the text for a Virtual Device Type General, Text with Idx 9, Name "Biotonne Pi" from "Hello World" to "Moin Moin" (a typical greeting word in northern part of Germany).

Idx	Hardware	ID	Unit	Name	Type	SubType	Data
9	Virtual Devices	00082009	1	Biotonne Pi	General	Text	Hello World

Command

```
mosquitto_pub -t 'domoticz/in' -m '{"idx":9,"nvalue":0,"svalue":"Moin Moin"}'
```

There is no terminal output from this command.

Possible checks if the command was successful:

Check Domoticz Widget has changed (Tab Utility)	
Check Domoticz Log entry (Tab Setup > Log)	2018-08-27 15:57:26.792 MQTT: Topic: domoticz/in, Message: {"idx":9,"nvalue":0,"svalue":"Moin Moin"}

Python

To be able to use MQTT from python, the paho client library is required.

Paho Client Library

To install, download the client and run the Python setup script to install the client.

There are two ways:

- cloning from github, which will install the client with source code and examples
- using pip and pip3

GitHub

Run from a terminal

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

Output

```
pi@DomoticzDev:~ $ mkdir python
pi@DomoticzDev:~ $ git clone https://github.com/eclipse/paho.mqtt.python.git
Cloning into 'paho.mqtt.python'...
remote: Counting objects: 2932, done.
remote: Compressing objects: 100% (66/66), done.
remote: Total 2932 (delta 27), reused 85 (delta 24), pack-reused 2834
Receiving objects: 100% (2932/2932), 835.76 KiB | 447.00 KiB/s, done.
Resolving deltas: 100% (1487/1487), done.
pi@DomoticzDev:~ $ cd paho.mqtt.python
pi@DomoticzDev:~/paho.mqtt.python $ sudo python setup.py install
running install
...
Installed /usr/local/lib/python2.7/dist-packages/paho_mqtt-1.3.1-py2.7.egg
Processing dependencies for paho-mqtt==1.3.1
Finished processing dependencies for paho-mqtt==1.3.1
```

Pip (Python2) & Pip3 (Python3)

Run from a terminal with sample output

Command

```
pip install paho-mqtt
```

Output

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

```
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.3.1
```

Command

```
pip3 install paho-mqtt
```

Output

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

Subscribe Sample Script

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

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

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

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

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

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(topic)

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)
    json_string = str(message.payload.decode("utf-8"))
    getidxvalue(json_string, idx)

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

# JSON Parsing
def getidxvalue(json_string, idx):
    parsed_json = json.loads(json_string)
    parsed_idx = int(parsed_json['idx'])
    if parsed_idx == idx:
        parsed_name = str(parsed_json['name'])
        parsed_value = str(parsed_json['svalue1'])
        print(parsed_idx, parsed_name, parsed_value)

print("Creating new instance")
client = mqtt.Client("P1")
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log
print("Connecting to the broker", broker)
client.connect(broker, 1883, 60)
time.sleep(4)
client.loop_forever()
```

Output Sample with lots of prints to show the steps & actions

```
$ python3 mqtttest.py
creating new instance
connecting to broker
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
log: Received CONNACK (0, 0)
Connected with result code 0
log: Sending SUBSCRIBE (d0) [(b'domoticz/out', 0)]
log: Received SUBACK
log: Received PUBLISH (d0, q0, r0, m0), 'domoticz/out', ... (229 bytes)
message received {
    "Battery" : 255,
    "RSSI" : 12,
    "description" : "",
    "dtype" : "General",
    "id" : "0000044D",
    "idx" : 1,
    "name" : "CPU_Usage",
    "nvalue" : 0,
    "stype" : "Percentage",
    "svalue1" : "0.77",
    "unit" : 1
}

message topic= domoticz/out
message qos= 0
message retain flag= 0
1 CPU_Usage 0.77

message topic= domoticz/out
message qos= 0
message retain flag= 0
1 CPU_Usage 0.33
^CTraceback (most recent call last):
  File "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
```

Publish Sample Script

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

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

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

Python3 Script

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

import time
import paho.mqtt.client as mqtt

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

# MQTT Callback Functions
def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))

def on_message(client, userdata, message):
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)
    print("message qos=",message.qos)
    print("message retain flag=",message.retain)

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

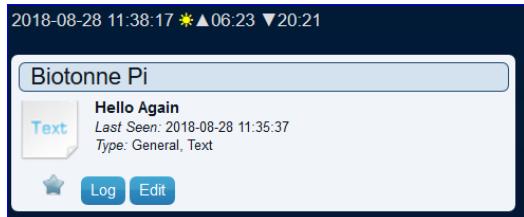
print(version)
print("Creating new instance")
client = mqtt.Client("P1")
client.on_connect = on_connect
client.on_message = on_message
client.on_log = on_log
print("Connecting to the broker", broker)
client.connect(broker, 1883, 60)
time.sleep(4)
print("Publishing message to topic",topic, "with payload",payload)
domoticz/in, Message: {"command": "udevice", "idx": 9, "svalue": "Hello Again"}
client.publish(topic,payload)
time.sleep(4)
client.loop_stop()
```

Output

```
pi@DomoticzDev:~/python $ python3 mqtt-pub-test.py
MQTT Test v20180827
Creating new instance
Connecting to the broker localhost
log: Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'P1'
```

```
Publishing message to topic domoticz/in with payload {"command": "udevice", "idx": 9, "svalue": "Hello Again"}  
log: Sending PUBLISH (d0, q0, r0, m1), 'b'domoticz/in'', ... (57 bytes)
```

Widget with updated text



SQL

Purpose

To execute CRUD operations on the Domoticz SQLite Database located /home/pi/Domoticz/domoticz.db.

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

Thoughts:

- using Python3 scripts triggered by a Switch or Node-RED with Dashboard UI (and input field)
- using Python3 with guizero – simple UI

HINTS

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

SQL Commands

For testing of SQL statements on the Domoticz database, used

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

Tables

The tables holding measurement data are

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

<TODO> need to check if this will change depending Domoticz version.

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

```
CREATE TABLE [DeviceStatus] ([ID] INTEGER PRIMARY KEY, [HardwareID] INTEGER NOT NULL, [DeviceID] VARCHAR(25) NOT NULL, [Unit] INTEGER DEFAULT 0, [Name] VARCHAR(100) DEFAULT Unknown, [Used] INTEGER DEFAULT 0, [Type] INTEGER NOT NULL, [SubType] INTEGER NOT NULL, [SwitchType] INTEGER DEFAULT 0, [Favorite] INTEGER DEFAULT 0, [SignalLevel] INTEGER DEFAULT 0, [BatteryLevel] INTEGER DEFAULT 0,
```

```
[nValue] INTEGER DEFAULT 0, [sValue] VARCHAR(200) DEFAULT null, [LastUpdate] DATETIME DEFAULT (datetime('now','localtime')), [Order] INTEGER BIGINT(10) default 0, [AddjValue] FLOAT DEFAULT 0, [AddjMulti] FLOAT DEFAULT 1, [AddjValue2] FLOAT DEFAULT 0, [AddjMulti2] FLOAT DEFAULT 1, [StrParam1] VARCHAR(200) DEFAULT '', [StrParam2] VARCHAR(200) DEFAULT '', [LastLevel] INTEGER DEFAULT 0, [Protected] INTEGER DEFAULT 0, [CustomImage] INTEGER DEFAULT 0, [Description] VARCHAR(200) DEFAULT '', [Options] TEXT DEFAULT null, [Color] TEXT DEFAULT NULL);  
  
CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill] FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT 0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));
```

Select

Purpose

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

Python

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

version="SQL Test v20180828"
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: Database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_device_status(conn):
    """
    Query all rows in the tasks DeviceStatus
    :param conn: the Connection object
    :return:
    """
    print("Select Device Status")
    cur = conn.cursor()
    cur.execute("SELECT * FROM DeviceStatus")
    rows = cur.fetchall()
    for row in rows:
        print(row)

def select_all_temperatures(conn):
    """
    Query all rows in the tasks Temperature
    :param conn: the Connection object
    :return:
    """
    print("Select all Temperatures")
    cur = conn.cursor()
    cur.execute("SELECT * FROM Temperature")
    rows = cur.fetchall()
    for row in rows:
```

```
    print(row)

def select_temperature_by_threshold(conn, threshold):
    """
    Query temperatures by threshold
    :param conn: the Connection object
    :param threshold:
    :return:
    """
    print("Select Temperature by Threshold %s" % (threshold))
    cur = conn.cursor()
    cur.execute("SELECT * FROM Temperature WHERE Temperature>?", (threshold,))
    rows = cur.fetchall()
    for row in rows:
        print(row)

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

        conn.close()

if __name__ == '__main__':
    main()
```

Output data from a Domoticz Test Database

Command

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

Output

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

Delete

Purpose

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

Sample SQL create statement for table [Temperature]

```
CREATE TABLE [Temperature] ([DeviceRowID] BIGINT(10) NOT NULL, [Temperature] FLOAT NOT NULL, [Chill] FLOAT DEFAULT 0, [Humidity] INTEGER DEFAULT 0, [Barometer] INTEGER DEFAULT 0, [DewPoint] FLOAT DEFAULT 0, [SetPoint] FLOAT DEFAULT 0, [Date] DATETIME DEFAULT (datetime('now','localtime')));
```

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

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

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

SELECT statement with WHERE clause

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

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

DELETE statement with WHERE clause

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

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

Checked result:

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

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

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

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

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

Python

Task: Delete temperature data.

Table: [Temperature]

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

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

import sqlite3
from sqlite3 import Error
import time
import os

# Define globals
version="SQL Test v20180828"
# Domoticz database path
database="/home/pi/domoticz/domoticz.db"

def DatabaseExists(database):
    if not os.path.exists(database):
        print("Database %s not found." % (database))
        return False
    else:
        print("Database %s found." % (database))
        return True

def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by the db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None

def select_all_temperatures(conn):
    """
    Query all rows in the tasks Temperature
    :param conn: the Connection object
```

```
:return:  
"""  
print("Select all Temperatures")  
cur = conn.cursor()  
cur.execute("SELECT * FROM Temperature")  
  
rows = cur.fetchall()  
  
for row in rows:  
    print(row)  
  
  
def select_temperature_by_threshold(conn, threshold):  
    """  
    Query temperatures by threshold  
    :param conn: the Connection object  
    :param threshold:  
    :return:  
    """  
    print("Select Temperature by Threshold %s" % (threshold))  
    cur = conn.cursor()  
    cur.execute("SELECT * FROM Temperature WHERE Temperature>?", (threshold,))  
  
    rows = cur.fetchall()  
  
    for row in rows:  
        print(row)  
  
def delete_temperature_by_threshold(conn, threshold):  
    """  
    Delete all temperatures by threshold  
    :param conn: the Connection object  
    :param threshold:  
    :return:  
    """  
    print("Delete Temperature by Threshold %s" % (threshold))  
    cur = conn.cursor()  
    cur.execute("DELETE FROM Temperature WHERE Temperature>?", (threshold,))  
    conn.commit()  
  
def main():  
    print(version)  
  
    if DatabaseExists(database):  
        print("Connecting to the database...",database)  
        # create a database connection  
        conn = create_connection(database)  
        with conn:  
            threshold = 40.0  
  
                select_temperature_by_threshold(conn, threshold)  
  
                delete_temperature_by_threshold(conn, threshold)  
  
                # Check if zero  
                select_temperature_by_threshold(conn, threshold)  
  
        conn.close()  
  
if __name__ == '__main__':  
    main()
```

Command

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

Output

```
SQL Test v20180828
Database /home/pi/domoticz/domoticz.db found.
Connecting to the database... /home/pi/domoticz/domoticz.db
Select Temperature by Threshold 40.0
(2, 40.8, 0.0, 0, 0.0, 0.0, '2018-08-28 19:20:00')
Delete Temperature by Threshold 40.0
Select Temperature by Threshold 40.0
```

Node-RED

Purpose

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

- Node-RED as alternative script engine, i.e. perform tasks, i.e. MQTT subscribe or publish
- Node-RED UI Dashboard Add-On as a simple monitor using Gauges & Charts or to control using buttons / slider / input-fields
- Trigger notifications or as watchdog for Domoticz process

[Node-RED](#) enables Flow-based programming for the Internet of Things.

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](#).

Installation

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

The package will be installed.

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

To enable autostart of Node-RED at every boot, run terminal command:

```
sudo systemctl enable nodered.service
```

Output

```
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service →  
/lib/systemd/system/nodered.service.
```

To go for sure if Node-RED is started during boot, reboot the Raspberry Pi

```
sudo shutdown -r now
```

and access Node-RED (<http://rpi-domoticz-ip:1880>).

Use disable, to disable autostart:

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

Update

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

Important

Prior updating Node-RED, backup the flows file, located in /home/pi/.node-red/
The flows have, depending on the Raspberry Pi device, following name:

flows_DomoticzDev.json

or

flows_DoPro.json

After updating, reboot the Raspberry Pi.

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

or if not working

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

Note

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

If this message occurs, then use the bash command:

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

Update Log Example

```
This script will remove versions of Node.js prior to version 6.x, and Node-RED and
if necessary replace them with Node.js 8.x LTS (carbon) and the latest Node-RED from Npm.
It also moves any Node-RED nodes that are globally installed into your user
~/.node-red/node_modules directory, and adds them to your package.json, so that
you can manage them with the palette manager.
It also tries to run 'npm rebuild' to refresh any extra nodes you have installed
that may have a native binary component. While this normally works ok, you need
to check that it succeeds for your combination of installed nodes.
To do all this it runs commands as root - please satisfy yourself that this will
not damage your Pi, or otherwise compromise your configuration.
If in doubt please backup your SD card first.
```

```
Are you really sure you want to do this ? [y/N] ? y
Running Node-RED update for user pi at /home/pi
This can take 20-30 minutes on the slower Pi versions - please wait.
```

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

```
Any errors will be logged to /var/log/nodered-install.log
All done.
```

You can now start Node-RED with the command node-red-start
or using the icon under Menu / Programming / Node-RED
Then point your browser to localhost:1880 or http://{your_pi_ip-address}:1880

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

Access Flows & Dashboard UI

To access Node-RED, point a browser at

<http://localhost:1880> (when using a Raspberry Pi with a connected monitor)

or

[http:// rpi-domoticz-ip:1880](http://rpi-domoticz-ip:1880) (remote)

This enables to manage Node-RED, define & run flows etc.

If the Dashboard add-on is installed, access the UI with its tabs:

<http://rpi-domoticz-ip:1880/ui>

Start, Stop, Log

Task	Terminal Command
Node-RED Start	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
```

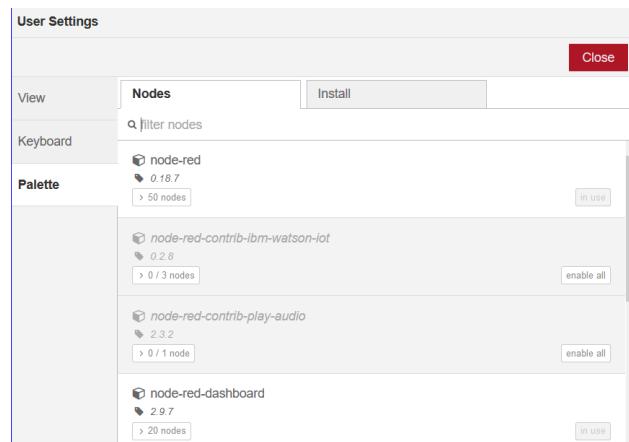
Install/Updates

To install or update any node package, NR recommends via *Node-RED Menu Settings > Manage Palette > Install*.

Note

If direct install via npm, Node-RED might be missing dependencies.

After install/update, restart Node-RED (terminal commands node-red-stop, node-red-start). It is not recommended to install using terminal commands.



Folder Locations

Data	Folder
Modules	/home/pi/.node-red
Flows	/home/pi/.node-red/flows_DomoticzDev.json
Additional Node Packages	/home/pi/.node-red/node_modules/<package name>

Additional Node Packages Sample Dashboard	/home/pi/.node-red/node_modules/node-red-dashboard
--	--

Flow Domoticz MQTT Messages

A simple example flow, to test Node-RED.

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

Node-RED Flow



The MQTT Broker is localhost with default port 1883.

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

Node-RED Flow Source

```
[{"id": "45e7c703.b9b9f", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 340, "wires": [[[ "cc4fd39c.684108"]]]}, {"id": "cc4fd39c.684108", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 340, "wires": [[[ "552cdad.37816a4"]]]}, {"id": "552cdad.37816a4", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG idx=ALL", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 810, "y": 340, "wires": []}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "useTLS": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}]
```

Output debug

<pre> info 36.38 [8/30/2018, 1:03:39 PM] node: DEBUG idx=ALL domoticz/out : msg.payload : Object ↪ { Battery: 255, RSSI: 12, description: "", dtype: "General", id: "0000044C" ... } [8/30/2018, 1:03:43 PM] node: DEBUG idx=ALL domoticz/out : msg.payload : Object ↪ { Battery: 100, RSSI: 5, description: "", dtype: "Temp + Humidity", id: "10766" ... } [8/30/2018, 1:03:49 PM] node: DEBUG idx=ALL domoticz/out : msg.payload : Object </pre>	<p>With a payload example (Hue Light with idx=118)</p> <pre>{ "Battery":255, "Level":90, "RSSI":12, "description":"", "dtype":"Light/Switch", "id":"00000005", "idx":118, "name":"Hue MakeLab", "nvalue":2, "stype":"Switch", "svalue1":"90", "switchType": "Dimmer","unit":1}</pre>
--	--

To access a property, convert the Domoticz message to a JSON object, then get the property, like
msg.payload.idx or msg.payload.Level

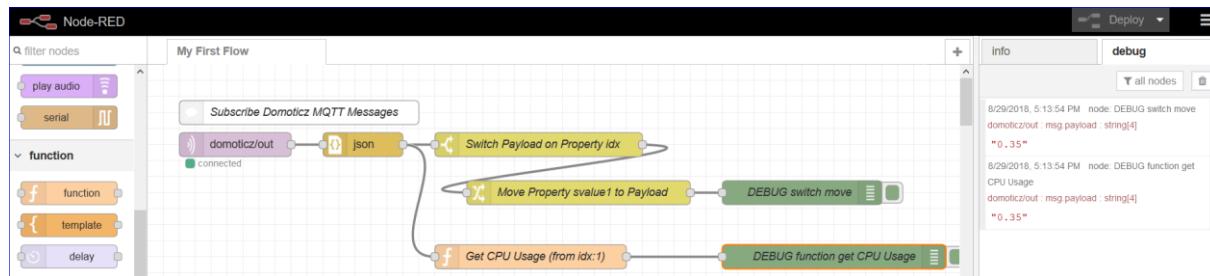
Flow Raspberry Pi CPU Usage

Subscribe to MQTT topic “Domoticz/in” to listen to all Domoticz MQTT messages. From the message payload (after conversion to JSON), select the property idx with value 1. Then select the property svalue1 to get the CPU Usage.

Domoticz Topic “Domoticz/in” Message Payload for idx = 1

```
{"Battery":255,"RSSI":12,"description":"","dtype":"General","id":"0000044D","idx":1,"name":"CPU_Usage","nvalue":0,"stype":"Percentage","svalue1":"0.28","unit":1}
```

Node-RED Flow



Node-RED Flow Source

```
[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 100, "wires": [{"id": "db5fff92.295628"}]}, {"id": "35214a2f.f89656", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG switch move", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 820, "y": 160, "wires": []}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 100, "wires": [{"id": "5f7a4a09.abb57c", "z": "13e0d06f.2b15d"}]}, {"id": "5f7a4a09.abb57c", "type": "switch", "z": "76cb1798.6a823", "name": "Switch Payload on Property idx", "property": "payload.idx", "propertyType": "msg", "rules": [{"t": "eq", "v": "1", "vt": "num"}], "checkall": true, "repair": false, "outputs": 1, "x": 490, "y": 100, "wires": [{"id": "912289d2.e8c1"}]}, {"id": "13e0d06f.2b15d", "type": "function", "z": "76cb1798.6a823", "name": "Get CPU Usage (from idx:5)", "func": "// Get the property idx from the payload\nnidx = msg.payload.idx;\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}\n// Assign the cpu usage to the message payload\nmsg.payload = cpu_usage;\n// Return the message\nreturn msg;\n", "outputs": 1, "noerr": 0, "x": 480, "y": 240, "wires": [{"id": "c030082a.286338", "z": "61fef7cc.4fc02", "x": 550, "y": 280}], {"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"}]}, {"id": "c030082a.286338", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG function get CPU Usage", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 240, "wires": []}, {"id": "acdab92.b0151c8", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG idx=1", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 380, "wires": []}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}]}
```

Output debug



Node-RED as MQTT Publisher

Instead of defining MQTT topics, a Node-RED flow is an alternative to publish MQTT topics with payload.

For an example, see Dashboard UI > Flow Virtual Sensor.

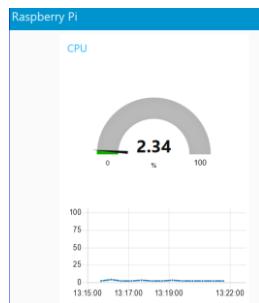
Dashboard UI

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

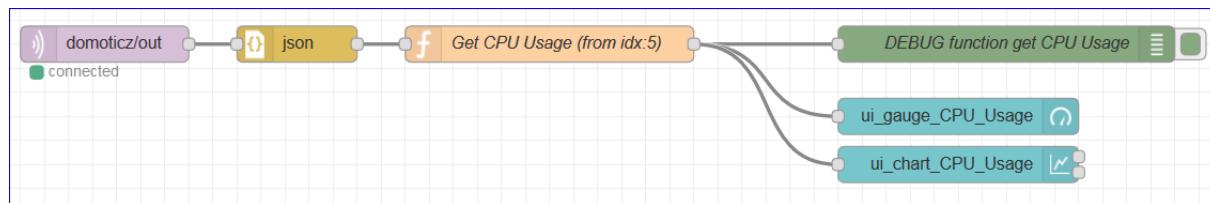
Flow Monitor CPU Usage

Build a simple example of a dashboard using the Raspberry Pi CPU Usage information, which is displayed in a Gauge and Line Chart.

Dashboard



Node-RED Flow



The Function Node selects idx 5 and builds message payload holding the CPU Usage in %.

```
// Get the property idx from the payload
idx = msg.payload.idx;
cpu_usage=-1;
// Check for idx 1
if (idx == 5) {
    // Get the cpu usage from property svalue1
    cpu_usage = msg.payload.svalue1;
    // node.warn("Idx=1, CPU Usage=" + cpu_usage);
    // Assign the cpu usage to the message payload
    msg.payload = cpu_usage;
    // Return the message
    return msg;
}
```

Node-RED Flow Source

```
[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 240, "wires": [[[ "db5fff92.295628"]]]}, {"id": "db5fff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 240, "wires": [[[ "13e0d06f.2b15d"]]]}, {"id": "13e0d06f.2b15d", "type": "function", "z": "76cb1798.6a823", "name": "Get CPU Usage (from idx:5)", "func": "// Get the property idx from the payload\nidx = msg.payload.idx;\nncpu_usage=-1;\n// Check for idx 1\\nif (idx == 5) {\n    // Get the cpu usage from property svalue1\n    cpu_usage = msg.payload.svalue1;\n    // node.warn(\"Idx=1, CPU Usage=\\" + cpu_usage);\n    // Assign the cpu usage to the message payload\n    msg.payload = cpu_usage;\n    // Return the message\n    return msg;\n}\n", "outputs": 1, "noerr": 0, "x": 480, "y": 240, "wires": [[[ "c030082a.286338", "61fef7cc.4fc02", "d5bc4ce8.ae1f28"]]]}, {"id": "c030082a.286338", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG function get CPU Usage", "active": true, "tosidebar": true, "console": false, "tostatus": false, "complete": "payload", "x": 860, "y": 240, "wires": []}, {"id": "61fef7cc.4fc02", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "gtype": "gage", "title": "", "label": "%", "form": "at": "{{value}}", "min": 0, "max": 100, "colors": ["#00b500", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "x": 820, "y": 300, "wires": []}, {"id": "d5bc4ce8.ae1f28", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "label": "", "chartType": "line", "legend": false, "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": true, "ymin": "0", "ymax": "100", "removeOlder": 1, "removeOlderPoints": "", "removeOlderUnit": "3600", "cutout": 0, "useOneColor": false, "colors": ["#1f77b4", "#aec7e8", "#ff7f0e", "#2ca02c", "#98df8a", "#d62728", "#ff9896", "#9467bd", "#c5b0d5"], "useOldStyle": false, "x": 820, "y": 340, "wires": [[[], []]]}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}, {"id": "c3b6865e.2b93e", "type": "ui_group", "z": "", "name": "CPU", "tab": "8a1743ab.9d931", "disp": true, "width": "6", "collapse": false}, {"id": "8a1743ab.9d931", "type": "ui_tab", "z": "", "name": "Raspberry Pi", "icon": "dashboard"}]
```

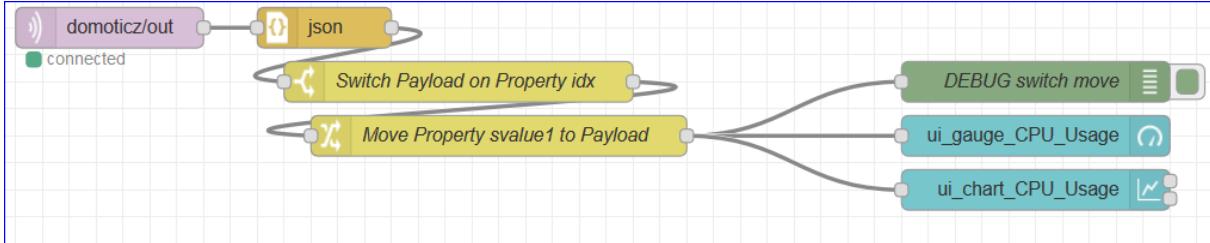
Note

The charts display each data change which could result in a rather frequent update. To avoid many updates, use a Delay Node which limits the number of messages to for example 12 messages per hour (Properties: "pauseType": "rate", "rate": "12", "nbRateUnits": "1", "rateUnits": "hour").

Switch Node and Change Node

Instead of using a Function Node, Switch Node and Change Node can be used.

Node-RED Flow



Node-RED Flow Source

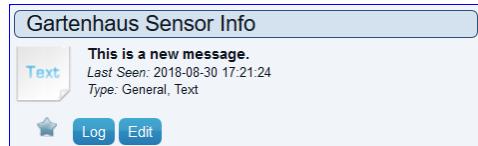
```

[{"id": "f67403bf.ebdc88", "type": "mqtt_in", "z": "76cb1798.6a823", "name": "", "topic": "domoticz/out", "qos": "2", "broker": "d58d713d.0e0eb8", "x": 110, "y": 100, "wires": [[[ "db5fff92.295628"]]}, {"id": "35214a2f.f89656", "type": "debug", "z": "76cb1798.6a823", "name": "DEBUG switch move", "x": 140, "y": 140, "wires": []}, {"id": "dbsffff92.295628", "type": "json", "z": "76cb1798.6a823", "name": "", "property": "payload", "action": "", "pretty": false, "x": 270, "y": 100, "wires": [[[ "5f7a4a09.abb57c"]]}], {"id": "5f7a4a09.abb57c", "type": "switch", "z": "76cb1798.6a823", "name": "Switch Payload on Property idx", "property": "payload.idx", "propertyType": "msg", "rules": [{"t": "eq", "v": "5", "vt": "num"}]}, {"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"}]}, {"id": "35214a2f.f89656", "type": "ui_gauge", "z": "76cb1798.6a823", "name": "ui_gauge_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "gtype": "gauge", "title": "", "label": "%", "format": "{{value}}", "min": 0, "max": 100, "colors": ["#00b500", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "x": 800, "y": 180, "wires": []}, {"id": "d5bc4ce8.ae1f28", "type": "ui_chart", "z": "76cb1798.6a823", "name": "ui_chart_CPU_Usage", "group": "c3b6865e.2b93e", "order": 0, "width": 0, "height": 0, "label": "", "chartType": "line", "legend": false, "xformat": "HH:mm:ss", "interpolate": "linear", "nodata": "", "dot": true, "ymin": 0, "ymax": 100, "removeOlder": 1, "removeOlderPoints": "", "removeOlderUnit": 3600, "cutout": 0, "useOneColor": false, "colors": ["#1f77b4", "#aec7e8", "#ff7f0e", "#2ca02c", "#98df8a", "#d62728", "#ff9986", "#9467bd", "#c5b0d5"], "useOldStyle": false, "x": 800, "y": 220, "wires": [[[], []]]}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": 1883, "clientid": "", "usetls": false, "compatmode": true, "keepalive": 60, "cleansession": true, "birthTopic": "", "birthQos": 0, "birthPayload": "", "closeTopic": "", "closeQos": 0, "closePayload": "", "willTopic": "", "willQos": 0, "willPayload": ""}, {"id": "c3b6865e.2b93e", "type": "ui_group", "z": "76cb1798.6a823", "name": "Raspberry Pi", "icon": "dashboard"}]
  
```

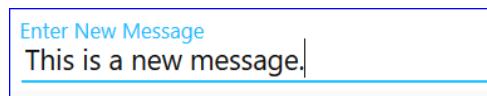
Flow Virtual Sensor Text Update

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

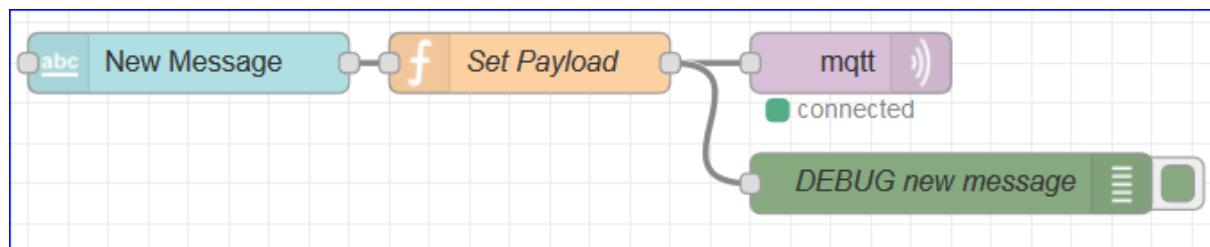
Widget showing the updated text



Node-RED Dashboard UI with ui_text_input Node



Node-RED Flow with Nodes ui_text_input > function > mqtt out



The text input node has a delay of 0, means the content is sent after pressing enter.
The function node created the new message which is published via MQTT using the localhost broker.

```

// Build a new message
var msginfo = {};
msginfo.topic = "domoticz/in";

// Get the text from the incoming payload from the text input node.
text = msg.payload;

// Build the new payload
msginfo.payload = { "idx" : 52, "nvalue": 0, "svalue": text};
node.warn(msginfo.payload);

// Return the message
return msginfo;
  
```

The information for building the MQTT payload is taken from the Domoticz API/JSON URL's documentation ([here](#)).

Node-RED Flow Source

```

[{"id": "38d6f42f.539a94", "type": "ui_text_input", "z": "76cb1798.6a823", "name": "New Message", "label": "Enter New Message", "group": "27ca36a9.4c3cba", "order": 0, "width": 0, "height": 0, "passthru": false, "mode": "text", "delay": "0", "topic": "", "x": 120, "y": 820, "wires": [[{"id": "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\nsetText = msg.payload;\n\n// Build the new payload\nmsginfo.payload = {\n    \"idx\": 52,\n    \"nvalue\": 0,\n    \"svalue\": text\n};\nnode.warn(msginfo.payload);\n\n// Return the message\nreturn\nmsginfo;\n", "outputs": 1, "noerr": 0, "x": 290, "y": 820, "wires": [[{"id": "d4be5444.33e23", "x": 450, "y": 100, "x2": 550, "y2": 100}, {"id": "f167bf33.2066e8", "x": 550, "y": 100, "x2": 650, "y2": 100}], {"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", "display": true, "width": 6, "collapse": false}, {"id": "d58d713d.0e0eb8", "type": "mqtt-broker", "z": "", "name": "Domoticz MQTT Broker", "broker": "localhost", "port": "1883", "clientid": "", "usetls": false, "compatmode": true, "keepalive": "60", "cleansession": true, "birthTopic": "", "birthQos": "0", "birthPayload": "", "closeTopic": "", "closeQos": "0", "closePayload": "", "willTopic": "", "willQos": "0", "willPayload": ""}, {"id": "8a1743ab.9d931", "type": "ui_tab", "z": "", "name": "Raspberry Pi", "icon": "dashboard"}]
```

Python Plug In

Purpose

The Python Plugin Framework allows to create an interface between Hardware (or Virtual Hardware) and Domoticz.

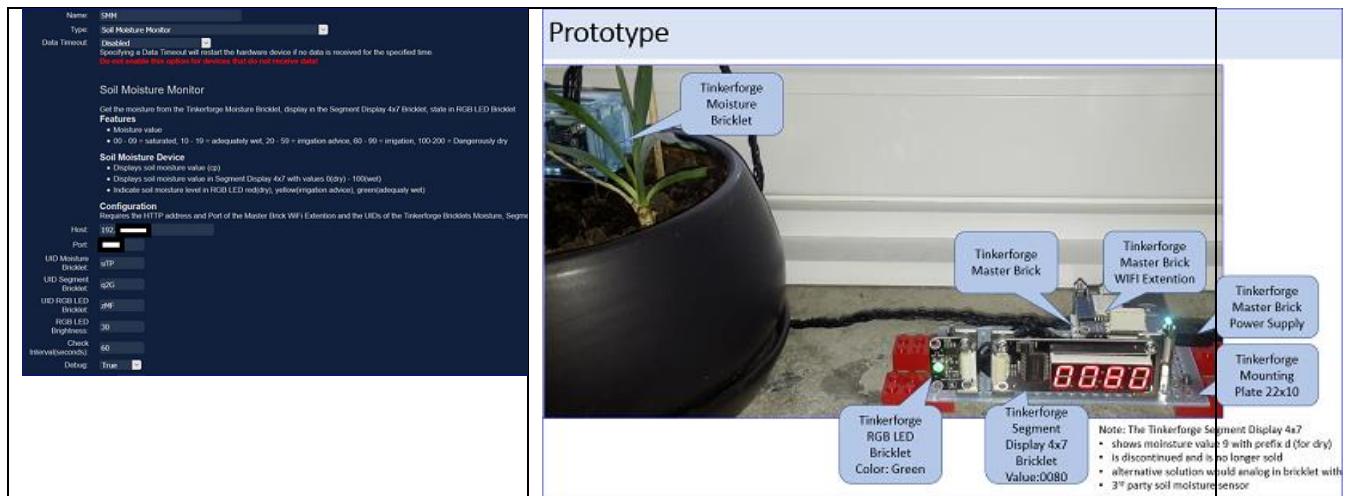
Mandatory to read this [manual](#) first and use as reference.

My first Python Plugin, **Soil Moisture Monitor**, can be found [here](#).

Purpose:

To Measure soil moisture of a plant, display the value in Domoticz Home Automation System, in a TinkerForge Segment Display 4x7 and TinkerForge RGB LED indicator.

Must state, it makes real fun to see how new hardware devices with according devices are added and used.



ESPEasy

Purpose

To explore connecting ESP type microcontrollers (like NodeMCU, WeMOS and many more), running [ESP Easy](#) with associated sensors, to Domoticz.
If not familiar with ESP Easy, start with the previous link shared.

Hardware: ESP8266 NodeMCU DevKit.

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

Experiments

Just a few experiments, to test how to connect and read values from sensors.

- Measure ambient light with sensor BH1750 (I2C) and trigger switch light if below darkness threshold.
- Potentiometer to control the brightness of a Hue light bulb.
- <TODO> More to add

Flash NodeMCU

Download

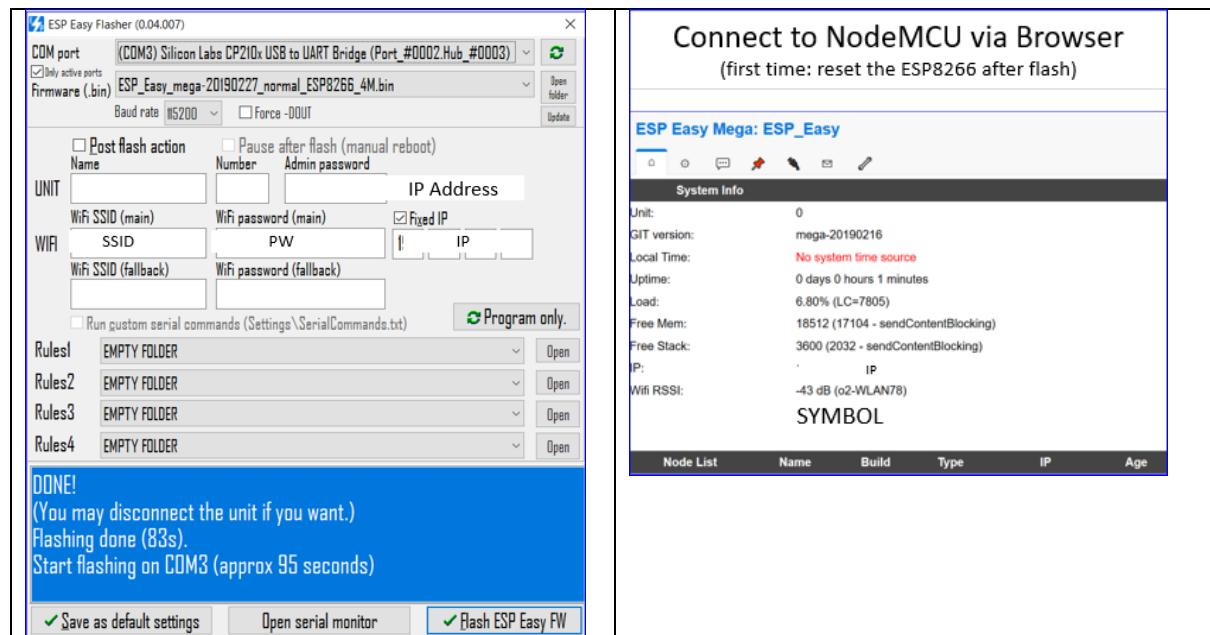
Downloaded ESPEasy_mega-20190227.zip from GitHub and extracted to a temp folder, i.e. domoticz-home-automation-workbook\functions\ESPEasy\r200\.

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 NodeMCU

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

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

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

Domoticz Controller

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

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

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 ESPEasy_BH1750 value Lux (see devices properties).
// If the lux value less then threshold set, turn on an LED else off.
// The LED is connected to pin D5 (GPIO14)

on ESPEasy_BH1750#Lux do

    if [ESPEasy_BH1750#Lux] < 100
        gpio,14,1
    else
        gpio,14,0
    endif

endon
```

Rules are logged (Menu Tools > System Log)

```
1011480: EVENT: ESPEasy_BH1750#Lux=120.00
1011499: ACT : gpio,14,0
1011502: SW : GPIO 14 Set to 0
1011524: Domoticz: Sensortype: 1 idx: 46 values: 120.00
1021477: BH1750 Address: 0x23 Mode: 0x1 : Light intensity: 63.33
1021480: EVENT: ESPEasy_BH1750#Lux=63.33
1021498: ACT : gpio,14,1
1021500: SW : GPIO 14 Set to 1
1021526: Domoticz: Sensortype: 1 idx: 46 values: 63.33
```

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

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

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

GPIO Commands

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

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

Turn an LED ON/OFF

ON

<http://espeasy-ip-address/control?cmd=gpio,14,1>

Result JSON Response

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

OFF

<http://espeasy-ip-address/control?cmd=gpio,14,0>

Result JSON Response

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

ESP8266

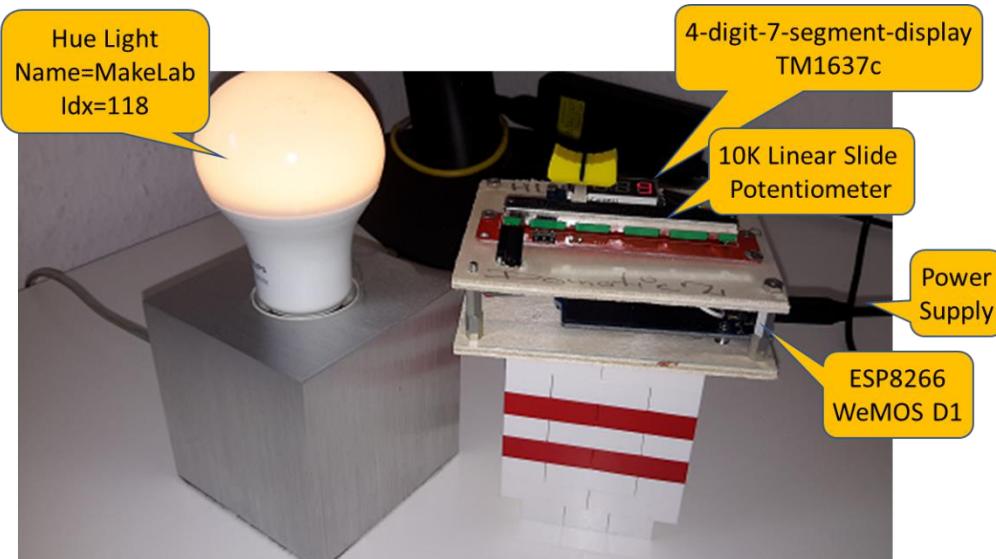
Purpose

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 (MakeLab, idx=118) connected to Domoticz, by using an ESP8266 microcontroller with a slide potentiometer 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]

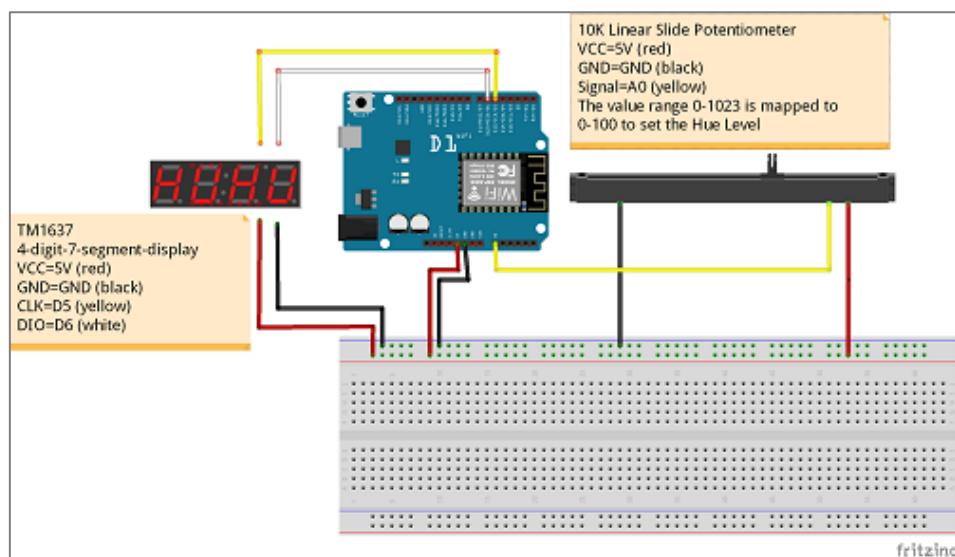
Wiring

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 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)
```

Appendix Domoticz Hints

Start, Stop, Status

Terminal commands

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

Output Status Sample

```
domoticz.service - LSB: Home Automation System
  Loaded: loaded (/etc/init.d/domoticz.sh)
  Active: active (running) since Mon 2018-08-27 09:32:39 CEST; 16min ago
    Process: 31803 ExecStop=/etc/init.d/domoticz.sh stop (code=exited,
  status=0/SUCCESS)
    Process: 32093 ExecStart=/etc/init.d/domoticz.sh start (code=exited,
  status=0/SUCCESS)
      CGroup: /system.slice/domoticz.service
              └─32100 /home/pi/domoticz/domoticz -daemon -www 8080 -sslww 443

Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.694 Status: Dom...z
Aug 27 09:32:39 139 domoticz[32098]: Domoticz is starting up....
Aug 27 09:32:39 139 domoticz[32100]: Domoticz running...
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Bui...6
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Sys...i
Aug 27 09:32:39 139 domoticz.sh[32093]: 2018-08-27 09:32:39.695 Status: Sta.../
Aug 27 09:32:39 139 domoticz.sh[32093]: domoticz: Domoticz is starting up....
Aug 27 09:32:39 139 systemd[1]: Started LSB: Home Automation System.
Hint: Some lines were ellipsized, use -l to show in full.
```

Lost Username and Password

Open terminal session and login as User Pi

Stop Domoticz and wait 15 seconds to complete

```
sudo service domoticz.sh stop
```

Check Domoticz Status

```
sudo service domoticz.sh stop
```

Change to Domoticz Folder

```
cd domoticz
```

Note

The full path is /home/pi/Domoticz

Start Domoticz with parameter no www password

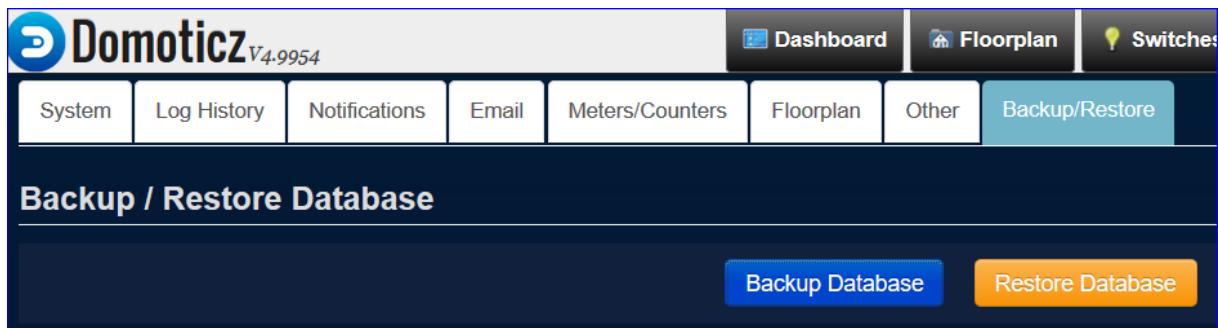
```
sudo ./domoticz -nowwwpwd
```

Create Backup

From the Domoticz UI go to:

Setup > Settings > Backup/Restore : Click Backup Database

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



Troubleshooting

Device Widget turns yellow/red

Check the device status using URL with parameter type=devices and rid=idx.

Example

Checking device named Windmesser (TFA Dostmann 30.3168 Windmeter) because the weather widget turned yellow.



From browser run:

```
http://NNN.NNN.1.60:8080/json.htm?type=devices&rid=28
```

Result JSON string (snippet)

```
{  
    "app_version" : "4.9999",  
    "result" : [  
        {  
            "BatteryLevel" : 0,  
            "Data" : "22.00;NNE;5;0;26.5;26.5",  
            "HardwareName" : "RFXtrx433e",  
            "HardwareType" : "RFXCOM - RFXtrx433 USB 433.92MHz Transceiver",  
            "SignalLevel" : 6,  
            "Used" : 1,  
            "idx" : "28"  
        }  
    ],  
    "status" : "OK",  
    "title" : "Devices"  
}
```

From the returned result, the **BatteryLevel** is 0.

It seems that the rechargeable battery is defect for this device.

The battery is recharged by a small solar panel on top of the device.

Appendix Domoticz Build Source

Sample commands for building Domoticz on a Raspberry Pi from the Domoticz Source. Followed the instructions from [here](#).

```
Swap File 100 to 500MB
sudo service dphys-swapfile status
sudo swapoff -a
free
sudo nano /etc/dphys-swapfile
  change 100 to 500
sudo swapon -a
free

Domoticz Build (took about 2 hours)
sudo apt-get install cmake make gcc g++ libssl-dev git libcurl4-gnutls-dev libusb-dev python3-dev zlib1g-dev

mkdir boost
cd boost
wget https://dl.bintray.com/boostorg/release/1.68.0/source/boost_1_68_0.tar.gz
tar xfz boost_1_68_0.tar.gz
cd boost_1_68_0/
./bootstrap.sh
./b2 stage threading=multi link=static --with-thread --with-system
sudo ./b2 install threading=multi link=static --with-thread --with-system
cd ../../
rm -Rf boost/
cd /home/pi
free

git clone https://github.com/domoticz/domoticz.git dev-domoticz
  Cloning into 'dev-domoticz'...
  remote: Counting objects: 59677, done.
  remote: Compressing objects: 100% (16/16), done.
  remote: Total 59677 (delta 9), reused 13 (delta 6), pack-reused 59655
  Receiving objects: 100% (59677/59677), 206.16 MiB | 3.82 MiB/s, done.
  Resolving deltas: 100% (43178/43178), done.
  Checking out files: 100% (2784/2784), done.

cd dev-domoticz
git pull
  Already up-to-date.
```

```
cmake -DCMAKE_BUILD_TYPE=Release CMakeLists.txt
-- The C compiler identification is GNU 6.3.0
-- The CXX compiler identification is GNU 6.3.0
...
-- Compiling Revision #9700
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/dev-domoticz

make
Scanning dependencies of target minizip
...
[100%] Linking CXX executable domoticz
[100%] Built target domoticz

ls
...
domoticz.sh
...

sudo ./domoticz
2018-09-12 16:18:10.882 Status: Domoticz V4.9999 (c)2012-2018 GizMoCuz
2018-09-12 16:18:10.882 Status: Build Hash: 9e8ea729, Date: 2018-09-09 18:27:57
2018-09-12 16:18:10.883 Status: Startup Path: /home/pi/dev-domoticz/
2018-09-12 16:18:11.292 Status: PluginSystem: Started, Python version '3.5.3'.
2018-09-12 16:18:11.299 Active notification Subsystems: gcm, http (2/13)
2018-09-12 16:18:11.304 Status: WebServer(HTTP) started on address: :: with port
8080
2018-09-12 16:18:11.318 Status: WebServer(SSL) started on address: :: with port
443
2018-09-12 16:18:11.319 Status: Proxymanager started.
2018-09-12 16:18:11.321 Starting shared server on: :::6144
2018-09-12 16:18:11.321 Status: TCPServer: shared server started...
2018-09-12 16:18:11.322 Status: RxQueue: queue worker started...
2018-09-12 16:18:13.323 Status: EventSystem: reset all events...
2018-09-12 16:18:13.323 Status: EventSystem: reset all device statuses...
2018-09-12 16:18:13.472 Status: PluginSystem: Entering work loop.
2018-09-12 16:18:13.480 Status: Python EventSystem: Initializing event module.
2018-09-12 16:18:13.481 Status: EventSystem: Started
2018-09-12 16:18:13.481 Status: EventSystem: Queue thread started...
2018-09-12 16:18:42.447 Status: Incoming connection from: NNN.NNN.1.3
```

Updating Domoticz

Binary install update:

Use the Web Interface to update Domoticz (Setup->Check for Updates)

Manual update:

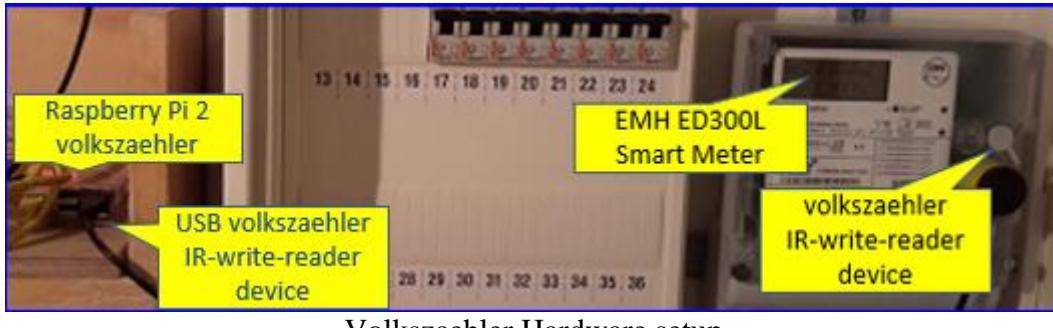
Stable: cd domoticz, ./updaterelease

Beta: cd domoticz, ./updatebeta

Source Update: cd domoticz, git pull, make -j 4 OR make

Appendix volkszaehler Setup

Description of the volkszaehler with the IR device, connected to a Raspberry Pi. volkszaehler.org is an Open Source (GPL license) Smart Metering Hard- and Software solution. Important 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 in 2015 – so things might have changed – checkout the volkszaehler website.

Option 1 WLAN static IP address

The communication uses WLAN with a static network address.

Set a static IP address by changing /etc/network/interfaces and /etc/wpa_supplicant/wpa_supplicant.conf.

File /etc/network/interfaces:

Editing via **sudo nano /etc/network/interfaces**

```
auto lo
iface lo inet loopback
auto wlan0
iface wlan0 inet manual
wireless-power off
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
iface wlan0 inet static
address rpi-domoticz-ip
netmask 255.255.255.0
broadcast NNN.NNN.0.255
gateway rpi-domoticz-ip
```

Edit **sudo nano /etc/wpa_supplicant/wpa_supplicant.conf**

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="YOURSSID"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP
    group=CCMP
    psk="YOURPSK"
}
```

Option 2 Ethernet static IP address

The volkszahler Raspberry Pi and the Home Automation Raspberry Pi are direct connected via an Ethernet cable. A standard cable is used, no twisted wires required.

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

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

volkszahler Raspberry Pi [Raspbian GNU/Linux 8 (jessie)] Linux 39 4.1.19-v7+ #858	Home Automation Raspberry Pi [Raspbian GNU/Linux 8 (jessie)] Linux openHABianPi 4.9.35-v7+ #1014
eth0 inet addr:169.254.87.85 Bcast:169.254.255.255 Mask:255.255.0.0	eth0 inet addr:169.254.87.84 Bcast:169.254.255.255 Mask:255.255.0.0

Set the static Ethernet network Address sudo nano /etc/dhcpcd.conf	sudo nano /etc/dhcpcd.conf
Add the lines	
interface eth0 static ip_address=169.254.87.85 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

The direct communication using an Ethernet cable is used.

volkszaehler

PowerMeter Setup

Powermeter: [EMH ED300L](#)

The powerpuls are 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***
- Important when changing the Raspberry Pi or adding USB devices, 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
```

Entry:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60",
ATTRS{serial}=="00F2E620", SYMLINK+="ttyUSB-VZ", MODE="0666"
```

Check the USB device information with:

```
sudo lsusb -v
```

To check detailed USB port information, use the command:

```
udevadm info --query=all --name=/dev/ttyUSB-VZ
```

Output Example

```
P: /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
N: ttyUSB0
S: serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0
S: serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0
S: ttyUSB-VZ
E: DEVLINKS=/dev/serial/by-path/platform-3f980000.usb-usb-0:1.5:1.0-port0 /dev/serial/by-id/usb-Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620-if00-port0 /dev/ttyUSB-VZ
E: DEVNAME=/dev/ttyUSB0
E: DEVPATH=/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.5/1-1.5:1.0/ttyUSB0/tty/ttyUSB0
E: ID_BUS=usb
E: ID_MODEL=CP2104_USB_to_UART_Bridge_Controller
E: ID_MODEL_ENC=CP2104\x20USB\x20to\x20UART\x20Bridge\x20Controller
E: ID_MODEL_FROM_DATABASE=CP210x UART Bridge / myAVR mySmartUSB light
E: ID_MODEL_ID=ea60
E: ID_PATH=platform-3f980000.usb-usb-0:1.5:1.0
E: ID_PATH_TAG=platform-3f980000_usb-usb-0_1_5_1_0
E: ID_REVISION=0100
E: ID_SERIAL=Silicon_Labs_CP2104_USB_to_UART_Bridge_Controller_00F2E620
E: ID_SERIAL_SHORT=00F2E620
E: ID_TYPE=generic
E: ID_USB_DRIVER=cp210x
E: ID_USB_INTERFACES=:ff0000:
E: ID_USB_INTERFACE_NUM=00
E: ID_VENDOR=Silicon_Labs
E: ID_VENDOR_ENC=Silicon\x20Labs
E: ID_VENDOR_FROM_DATABASE=Cygnal Integrated Products, Inc.
E: ID_VENDOR_ID=10c4
E: MAJOR=188
E: MINOR=0
E: SUBSYSTEM=tty
E: TAGS=:systemd:
E: USEC_INITIALIZED=5005222
```

USB Configuration

The USB port settings are 9600 Baud 8 n.

Set via command stty:

Minicom

The setup can also be done via [minicom](#)

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

Ensure port is set during boot

Add this line into /etc/rc.local (by sudo nano /etc/rc.local)

Check Data coming from the USB device (option 1):

```
cat /dev/ttyUSB-VZ | od -tx1
```

Output Example:

```
00000000 2f 38 01 01 63 e4 c7 00 76 07 00 0a 00 07 db 75  
00000020 62 00 62 00 72 63 07 01 77 01 0b 09 01 4d 48 00  
...
```

Check Data coming from the USB device (option 2):

```
xxd </dev/ttyUSB-VZ
```

Define Channels

To define the channels, reboot the RPi and access via Webbrowser the volkszaehler Frontend using its ip address: <http://rpi-vz-ip>

Define Channels using the volkszaehler FrontEnd

```
Channel 1 =
{"version": "0.3", "entity": {"uuid": "d14e9a80-8894-11e5-9dc8-298a9a1001ac", "type": "current", "active": true, "color": "aqua", "description": "Strom Direktverbrauch", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Direktverbrauch", "yaxis": "auto"}}

Channel 2 =
{"version": "0.3", "entity": {"uuid": "e460bec0-8894-11e5-b0d8-3f141d19092c", "type": "current", "active": true, "color": "aqua", "description": "Strom Gesamtverbrauch", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Gesamtverbrauch", "yaxis": "auto"}}

Channel 3 = 1.8.0
{"version": "0.3", "entity": {"uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "type": "electric meter", "active": true, "color": "#0033ff", "cost": 0.1992, "description": "Stromverbrauch Fuchsbau", "fillstyle": 0.2, "initialconsumption": 20, "public": true, "resolution": 1000, "style": "steps", "title": "Stromverbrauch", "yaxis": "auto"}}
Channel 4 = 16.7.0
{"version": "0.3", "entity": {"uuid": "a3c76600-8895-11e5-9432-333f81e04451", "type": "current", "active": true, "color": "aqua", "description": "Wirkleistung", "fillstyle": 0, "public": true, "resolution": 10000, "style": "steps", "title": "Wirkleistung", "yaxis": "auto"}}
```

vzlogger Setup

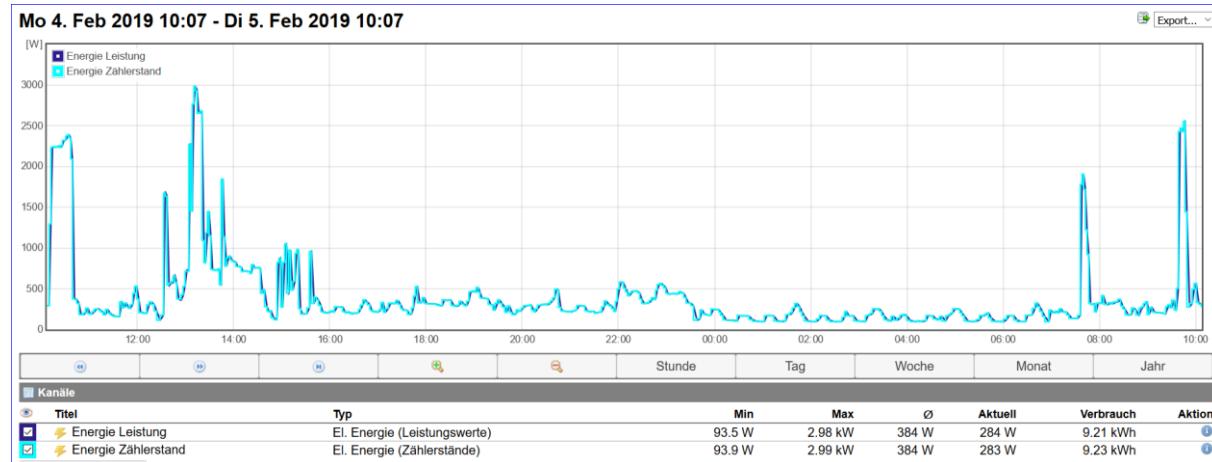
The defined channels must be defined in the file /etc/vzlogger.conf :

```
sudo nano /etc/vzlogger.conf
```

Two channels are defined "**Energie Leistung**" and "**Energie Zaehlerstand**":

```
/**  
 * vzlogger configuration  
 * Use properly encoded JSON with javascript comments  
 * http://wiki.volkszaehler.org/software/controller/vzlogger#configuration  
 * Online configuration editor: http://volkszaehler.github.io/vzlogger/  
 */  
  
{  
// General settings  
"daemon": true, // run periodically  
"verbosity": 0, // log (0=log_alert,1=log_error,3=log_warning,5=log_info,10=log_debug,15=log_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://rpi-vz-ip> showing the two channels defined.



Clicking on the Aktion (i) icon provides the details for the two channels defined:

"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4"	"uuid": "5b8ea2a0-b342-11e8-bec6-15c040e6d041"																																																		
<p>Details für Energie Leistung</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>UUID</td> <td>958bce60-b342-11e8-b54f-bbec0573e1f4</td> </tr> <tr> <td>Middleware</td> <td>middleware.php</td> </tr> <tr> <td>Typ</td> <td> El. Energie (Leistungswerte)</td> </tr> <tr> <td>Cookie</td> <td></td> </tr> <tr> <td>Titel</td> <td>Energie Leistung</td> </tr> <tr> <td>Öffentlich</td> <td></td> </tr> <tr> <td>Farbe</td> <td>#311b92</td> </tr> <tr> <td>Stil</td> <td>Linien</td> </tr> <tr> <td>Füllgrad</td> <td>0</td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Aktiv</td> <td></td> </tr> </tbody> </table> <p>Daten Löschen Bearbeiten Schließen</p>	Eigenschaft	Wert	UUID	958bce60-b342-11e8-b54f-bbec0573e1f4	Middleware	middleware.php	Typ	El. Energie (Leistungswerte)	Cookie		Titel	Energie Leistung	Öffentlich		Farbe	#311b92	Stil	Linien	Füllgrad	0	Achse	auto	Aktiv		<p>Details für Energie Zählerstand</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>UUID</td> <td>5b8ea2a0-b342-11e8-bec6-15c040e6d041</td> </tr> <tr> <td>Middleware</td> <td>middleware.php</td> </tr> <tr> <td>Typ</td> <td> El. Energie (Zählerstände)</td> </tr> <tr> <td>Cookie</td> <td></td> </tr> <tr> <td>Titel</td> <td>Energie Zählerstand</td> </tr> <tr> <td>Auflösung</td> <td>1000/kWh</td> </tr> <tr> <td>Öffentlich</td> <td></td> </tr> <tr> <td>Farbe</td> <td>aqua</td> </tr> <tr> <td>Stil</td> <td>Stufen</td> </tr> <tr> <td>Füllgrad</td> <td>0</td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Aktiv</td> <td></td> </tr> </tbody> </table> <p>Daten Löschen Bearbeiten Schließen</p>	Eigenschaft	Wert	UUID	5b8ea2a0-b342-11e8-bec6-15c040e6d041	Middleware	middleware.php	Typ	El. Energie (Zählerstände)	Cookie		Titel	Energie Zählerstand	Auflösung	1000/kWh	Öffentlich		Farbe	aqua	Stil	Stufen	Füllgrad	0	Achse	auto	Aktiv	
Eigenschaft	Wert																																																		
UUID	958bce60-b342-11e8-b54f-bbec0573e1f4																																																		
Middleware	middleware.php																																																		
Typ	El. Energie (Leistungswerte)																																																		
Cookie																																																			
Titel	Energie Leistung																																																		
Öffentlich																																																			
Farbe	#311b92																																																		
Stil	Linien																																																		
Füllgrad	0																																																		
Achse	auto																																																		
Aktiv																																																			
Eigenschaft	Wert																																																		
UUID	5b8ea2a0-b342-11e8-bec6-15c040e6d041																																																		
Middleware	middleware.php																																																		
Typ	El. Energie (Zählerstände)																																																		
Cookie																																																			
Titel	Energie Zählerstand																																																		
Auflösung	1000/kWh																																																		
Öffentlich																																																			
Farbe	aqua																																																		
Stil	Stufen																																																		
Füllgrad	0																																																		
Achse	auto																																																		
Aktiv																																																			
<p>Bearbeiten von Energie Leistung</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Titel</td> <td>Energie Leistung</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td></td> </tr> <tr> <td>Stil</td> <td>lines</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Auflösung</td> <td></td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </tbody> </table> <p>Speichern Abbrechen</p>	Eigenschaft	Wert	Titel	Energie Leistung	Öffentlich	<input checked="" type="checkbox"/>	Farbe		Stil	lines	Füllgrad		Achse	auto	Auflösung		Kosten		Initialverbrauch		<p>Bearbeiten von Energie Zählerstand</p> <table border="1"> <thead> <tr> <th>Eigenschaft</th> <th>Wert</th> </tr> </thead> <tbody> <tr> <td>Titel</td> <td>Energie Zählerstand</td> </tr> <tr> <td>Auflösung</td> <td>1000</td> </tr> <tr> <td>Öffentlich</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>Farbe</td> <td></td> </tr> <tr> <td>Stil</td> <td>steps</td> </tr> <tr> <td>Füllgrad</td> <td></td> </tr> <tr> <td>Achse</td> <td>auto</td> </tr> <tr> <td>Kosten</td> <td></td> </tr> <tr> <td>Initialverbrauch</td> <td></td> </tr> </tbody> </table> <p>Speichern Abbrechen</p>	Eigenschaft	Wert	Titel	Energie Zählerstand	Auflösung	1000	Öffentlich	<input checked="" type="checkbox"/>	Farbe		Stil	steps	Füllgrad		Achse	auto	Kosten		Initialverbrauch											
Eigenschaft	Wert																																																		
Titel	Energie Leistung																																																		
Öffentlich	<input checked="" type="checkbox"/>																																																		
Farbe																																																			
Stil	lines																																																		
Füllgrad																																																			
Achse	auto																																																		
Auflösung																																																			
Kosten																																																			
Initialverbrauch																																																			
Eigenschaft	Wert																																																		
Titel	Energie Zählerstand																																																		
Auflösung	1000																																																		
Öffentlich	<input checked="" type="checkbox"/>																																																		
Farbe																																																			
Stil	steps																																																		
Füllgrad																																																			
Achse	auto																																																		
Kosten																																																			
Initialverbrauch																																																			

API Channel Properties

Energie Leistung

```
http://rpi-vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

```
{"version":"0.3","entity":{"uuid":"958bce60-b342-11e8-b54f-bbec0573e1f4","type":"powersensor","color":"#311b92","fillstyle":0,"public":true,"style":"lines","title":"Energie Leistung","yaxis":"auto"}}
```

Energie Zaehlerstand

```
http://rpi-vz-ip/middleware.php/channel/5b8ea2a0-b342-11e8-bec6-15c040e6d041.json
```

```
{"version":"0.3","entity":{"uuid":"5b8ea2a0-b342-11e8-bec6-15c040e6d041","type":"electric meter","color":"aqua","fillstyle":0,"public":true,"resolution":1000,"style":"steps","title":"Energie Z\u00e4ehlerstand","yaxis":"auto"}}
```

vzlogger Service

Ensure the vzlogger service is correctly defined.

Note

Ensure the vzlogger command is using the parameter -c (see ExecStart).

```
cd /etc/systemd/system  
sudo nano vzlogger.service
```

Content:

```
[Unit]  
Description=vzlogger  
After=syslog.target network.target  
Requires=mysql.service  
[Service]  
ExecStart=/usr/local/bin/vzlogger -c /etc/vzlogger.conf  
ExecReload=/bin/kill -HUP $MAINPID  
StandardOutput=null  
[Install]  
WantedBy=multi-user.target
```

vzlogger Commands

vzlogger Enable as a Service	systemctl enable vzlogger
vzlogger Start	systemctl start vzlogger OR 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 <u>Debug</u>	vzlogger -f

Test

For test purposes a vzlogger.test can be used to check data flow ([vzlogger configuration](#)).

```
{
  "retry" : 30, /* how long to sleep between failed requests, in seconds */
  "daemon": false,           /* run periodically */
  "verbosity" : 15,          /* between 0 and 15 */
  "log" : "/var/log/vzlogger.log",/* path to logfile, optional */
  "local" :
  {
    "enabled" : false, /* start the local HTTPd for serving live readings? */
    "port" : 80,      /* the TCP port for the local HTTPd */
    "index" : true,   /* provide a index listing of available channels? */
    "timeout" : 30,   /* timeout for long polling comet requests, 0 disables comet,sec*/
    "buffer" : 600   /* how long to buffer readings for the local interface, secs */
  },
  "meters" :
  [
    {
      "enabled" : true,      /* disabled meters will be ignored */
      "protocol" : "sml",    /* use 'vzlogger -h' for list of available protocols */
      "device" : "/dev/usb-ir-lesekopf0",
    },
    {
      "enabled" : true,      /* disabled meters will be ignored */
      "protocol" : "sml",    /* use 'vzlogger -h' for list of available protocols */
      "device" : "/dev/usb-ir-lesekopf1",
    }
  }
}
```

vzlogger Start:

```
vzlogger -c /etc/vzlogger.conf
```

vzlogger Check status:

```
systemctl status vzlogger
```

Output Example:

```
? vzlogger.service - vzlogger
   Loaded: loaded (/etc/systemd/system/vzlogger.service; enabled)
   Active: active (running) since Fri 2015-11-13 11:55:53 CET; 13min ago
     Main PID: 670 (vzlogger)
        CGroup: /system.slice/vzlogger.service
                  +-670 /usr/local/bin/vzlogger -c /etc/vzlogger.conf
Nov 13 11:55:53 raspberrypi systemd[1]: Started vzlogger.
```

API volkszaehler Hints

API Reference: <http://wiki.volkszaehler.org/development/api/reference>

API Obtain Configuration Request

```
http://rpi-vz-ip/middleware.php/entity/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

Result

```
{"version": "0.3", "entity": {"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4", "type": "powersensor", "color": "#311b92", "fillstyle": 0, "public": true, "style": "lines", "title": "Energie Leistung", "yaxis": "auto"}}
```

API Channel Properties

```
http://rpi-vz-ip/middleware.php/channel/958bce60-b342-11e8-b54f-bbec0573e1f4.json
```

Result

```
{"version": "0.3", "entity": {"uuid": "958bce60-b342-11e8-b54f-bbec0573e1f4", "type": "powersensor", "color": "#311b92", "fillstyle": 0, "public": true, "style": "lines", "title": "Energie Leistung", "yaxis": "auto"}}
```

API Entities

```
http://rpi-vz-ip/middleware.php/capabilities/definitions/entities.json
```

API Read Data Requests

Note

Relative Dateformats = <http://de.php.net/manual/en/datetime.formats.relative.php>

Get All Data

```
http://rpi-vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json
```

Get One Record

```
http://rpi-vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?tuples=1
```

Result

```
{"version": "0.3", "data": {"tuples": [[[1447316802613, 246.491, 19046]]], "uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "from": 1447262000365, "to": 1447316802613, "min": [1447316802613, 246.49134831111], "max": [1447316802613, 246.49134831111], "average": 246.491, "consumption": 3752.3, "rows": 1}}
```

Get the last value

```
http://rpi-vz-ip/middleware.php/data/23844c00-8895-11e5-a642-6d8ce36c9a44.json?from=now&to=now
```

Result

```
{"version": "0.3", "data": {"tuples": [[[1447317119885, 201.681, 1], [1447317121700, 198.347, 1]]], "uuid": "23844c00-8895-11e5-a642-6d8ce36c9a44", "from": 1447317118100, "to": 1447317121700, "min": [1447317121700, 198.3470743513], "max": [1447317119885, 201.68067227331], "average": 200, "consumption": 0.2, "rows": 3}}
```

API Logging

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

Using the volkzaehler Frontend to display data

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

Appendix Tools

In progress developing some tools for managing Domoticz.

Domoticz Internal Script Viewer

Purpose

- To view offline the Domoticz internal scripts (focus on dzVents scripts), which are created using the web ui event editor
- To export the scripts to the clipboard, text file or HTML file

[Download](#)

Appendix TODO

In the documentation, specific topic ToDos are tagged with <TODO>. The list below are more generic TODOs.

Topic	Details
Notifications	Explore usage of notifications other than Email.
Thresholds	Set threshold via another UI, i.e. Node-RED or Custom UI.
Build house control functions	These are implemented but not shared yet.