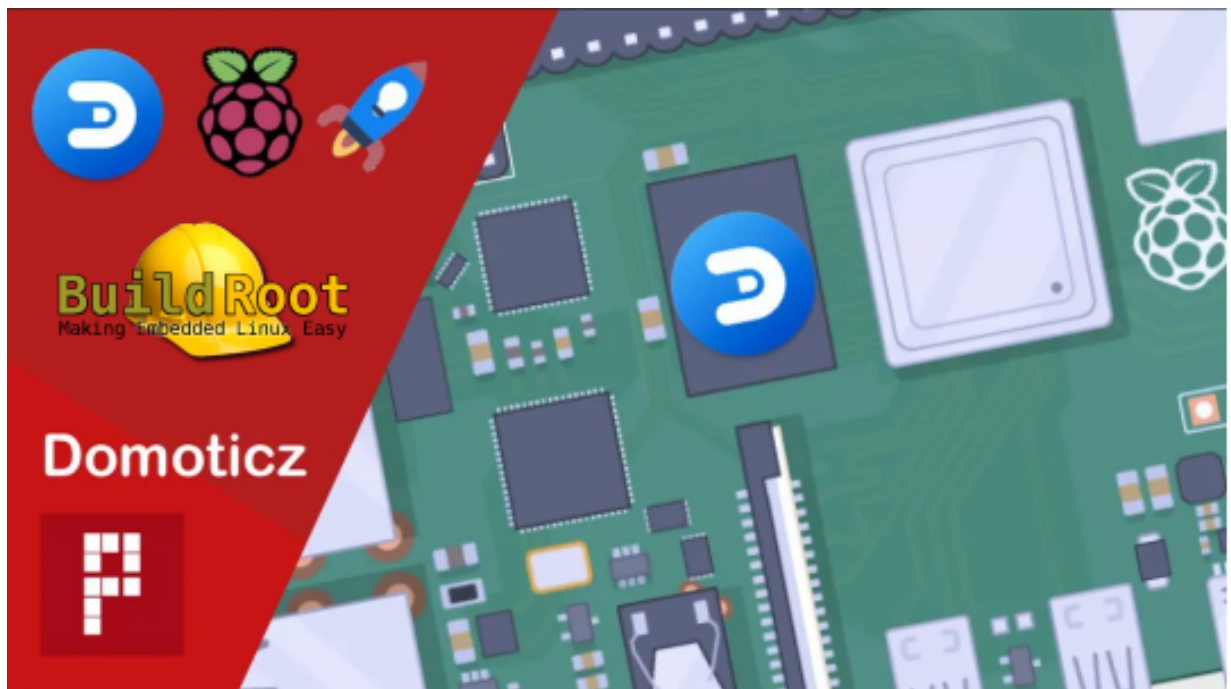


Embedded Linux for domotic

Marvyn Pannetier
Hadrien Jaubert

10th of June 2021



Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 3 |
| 2 | requirements | 3 |
| 3 | Generate and emulate our embedded Linux | 3 |
| 3.1 | optional: install your crossed compilation environment | 3 |
| 3.2 | Install the emulation software Qemu | 3 |
| 3.3 | Download a kernel for raspberrypi3 | 3 |
| 3.4 | Install Buildroot | 4 |
| 3.5 | Configure and generate your first Embedded Linux | 4 |
| 3.5.1 | Change the crossed compilation chain | 5 |
| 3.5.2 | Change the image size | 5 |
| 3.5.3 | Generate your image | 5 |
| 3.5.4 | Recompile your kernel for QEMU | 5 |
| 3.6 | Emulate a raspberrypi with your OS | 6 |
| 4 | Put and test your OS on a raspberrypi 3B+ | 8 |
| 4.1 | Configure your Embedded Linux | 8 |
| 4.2 | Put our image on the sd card | 9 |
| 4.3 | Modify the network configuration | 10 |
| 4.4 | Optional: put your keyboard in AZERTY | 11 |
| 4.5 | Configuration script | 12 |
| 4.6 | Test your image ! | 12 |
| 4.7 | Add user | 12 |
| 4.8 | Connect with ssh using Keys | 13 |

| | | |
|----------|---|-----------|
| 5 | Configure and test your Domoticz | 14 |
| 5.1 | Check that your domoticz is running | 14 |
| 5.2 | Add sensors in domoticz | 14 |
| 5.3 | Add actuator (LED) | 17 |
| 6 | Application and access to our Domoticz | 23 |

1 Introduction

The goal of this document is to explain the different state to create an embedded linux with buildroot. In the next page you will have all the keys to build your own Linux. So good reading and enjoy !

2 requirements

In order to this you will need to work on a Linux environment (Ubuntu, Debian...). If you want to do the "Part 2" you will also need a Raspberrypi card. In this document we will work with Raspberrypi 3 B+.

3 Generate and emulate our embedded Linux

3.1 optional: install your crossed compilation environment

Firstly, you can create a folder called "Development-tools" using:

```
user@pcname:"yourpath"$ mkdir Development-tools
```

Then you have to clone the official development tools for Raspberrypi where we can find the crossed compilation chains compatible with the different versions of the Raspberry PI card:

```
user@pcname:"yourpath"$ git clone https://github.com/raspberrypi/tools
```

Now you have all the tools you want, if you want to verify that the crossed compilation is installed, use:

```
user@pcname:"yourpath"$ ./tools/arm-bcm2708/arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc -v
```

3.2 Install the emulation software Qemu

The installation is simple you just need to use the next command:

```
user@pcname:"yourpath"$ sudo apt-get install qemu-system-arm
```

3.3 Download a kernel for raspberrypi3

In order to emulate our linux os we will have to use a modify kernel for raspberrypi to be sure we will don't have any problem. So clone the next git repository in your folder Development-tools:

```
user@pcname:"yourpath"$ git clone https://github.com/dhruvvyas90/qemu-rpi-kernel
```

3.4 Install Buildroot

In this part, we will talk about something more serious and interesting ! Indeed, Buildroot is a software which will allow to generate our own embedded Linux. It will allow to choose the applications we want to include in our OS and it will take care of generating sources, create dependencies, generate an image...etc. You certainly understood it, you have a lot of choice for your Linux with Buildroot.

Firstly, you must install a package needed for Buildroot:

```
user@pcname:"yourpath"$ sudo apt-get install libncurses5-dev
```

Then you can clone the next repository:

```
user@pcname:"yourpath"$ git clone https://github.com/buildroot/buildroot.git
```

Now you have a folder called buildroot. However, there is an issue with domoticz in the last version of buildroot so at the moment we will use the version 2019.02.x. In order to use this version you need to go to the right branch. So, when you are in the buildroot folder use the following command:

```
user@pcname:"yourpath"$ git checkout -b 2019.02.x origin/2019.02.x
```

You can now verify if you are on the right branch with the command:

```
user@pcname:"yourpath"$ git branch -a
```

3.5 Configure and generate your first Embedded Linux

Now we can talk seriously! At the end of this section you will have your own OS.

Firstly, always in your folder Development-tools, use:

```
user@pcname:"yourpath"$ cp -R buildroot buildroot-qemu-rpi-domoticz
```

This command will duplicate our buildroot folder, in that way we will always have the original folder if we make a mistake with the new one. Then go to the new folder:

```
user@pcname:"yourpath"$ cd buildroot-qemu-rpi-domoticz
```

We can start the configuration of our future OS, first configure your buildroot for raspberrypi using:

```
user@pcname:"yourpath"$ make raspberrypi_defconfig
```

then use the following command to enter in the configuration menu:

```
user@pcname:"yourpath"$ make menuconfig
```

So now, let's make some change!

3.5.1 Change the crossed compilation chain

In a first time, we can change the crossed compilation chain and choose the one we install before. Go to the Toolchain menu and put External toolchain in Toolchain Type. Then select Toolchain (Custom toolchain) below. Now you must give the path to our crossed compilation chain. If you choose create the folder Development-tools in /home/user and if you install correctly the crossed compilation in this Development-tools, then you have to write /home/user/Development-tools/tools/arm-bcm2708/arm-linux-gnueabi/ in the toolchain path line. Finally, you can write arm-linux-gnueabi in the line Toolchain prefix.

After that choose: External toolchain gcc version (4.9.x), External toolchain kernel headers series (4.1.x) and External toolchain C library (glibc/eglibc). Finally, select Toolchain has C++ support.

3.5.2 Change the image size

Indeed, if you choose to use the crossed compilation chain we installed before you need to use glibc instead of uClibc that takes a smaller space. So go to the "Filesystem images" menu and change (60M) exact size into (200M) exact size

You can save and go out of this window

3.5.3 Generate your image

This section is very simple but may take quite a long time to be completed. To generate you just have to use the command `user@pcname:"yourpath"$ make` in your folder buildroot-qemu-rpi-domoticz and wait !

3.5.4 Recompile your kernel for QEMU

In order to use the last kernel you can recompile him with the following protocol:

Firstly, you have to edit the script called build-kernel-qemu. Go to tools with:

```
user@pcname:"yourpath"$ cd Development-tools/qemu-rpi-kernel/tools
```

and edit the script, for example:

```
user@pcname:"yourpath"$ emacs build-kernel-qemu &
```

in this file change the line `#COMMIT=""` by `COMMIT="raspberrypi-kernel_1.20180417-1"`
Then comment the line `USE_GIT=1` and finally replace

```
wget -c https://github.com/raspberrypi/linux/archive/$COMMIT.tar.gz -O linux-$COMMIT.tar.gz
```

by

```
if [ ! -f linux-$COMMIT.tar.gz ]
then
wget -chttps : //github.com/raspberrypi/linux/archive/$COMMIT.tar.gz -O linux-$COMMIT.tar.gz
fi
```

After that you have to make this command which will install gcc for arm:

```
user@pcname:"yourpath"$ apt-get install gcc-arm-linux-gnueabi
```

Now you can save and exit this file and execute it:

```
user@pcname:"yourpath"$ bash build-kernel-qemu
```

After a certain time a window will open, you have nothing to do with it so just exit and save. Then you must wait that the compilation phase finished

When it is finished, you can copy your kernel for example, in a folder called raspbian that you can create in the folder Development-tools, use: `user@pcname:"yourpath"$ mkdir raspbian`

when you are in Development-tools, go in `/qemu-rpi-kernel/tools/linux-raspberrypi-kernel_1.20180417-1/arch/arm/boot/` and copy with:

```
user@pcname:"yourpath" $ cp ../zImage "your_path"/Development-tools/raspbian/kernel-20180417-1
```

So now you have recompiled your kernel and we can use it later!

3.6 Emulate a raspberrypi with your OS

We installed qemu-system-arm earlier, it is time to use it now. Go in Development-tools folder and create a file called start.sh in which you will write the following lines:

```
qemu-system-arm \
  -kernel qemu-rpi-kernel/kernel-qemu-4.14.79-stretch \
  -cpu arm1176 \
  -m 256 \
  -M versatilepb \
  -dtb qemu-rpi-kernel/versatile-pb.dtb \
  -no-reboot -append "root=/dev/sda2 panic=1 rootfstype=ext4 rw" \
  -net nic -net user,hostfwd=tcp::5022-:22,hostfwd=tcp::8080-:8080 \
  -drive file=buildroot-qemu-rpi-domoticz/output/images/sdcard.img,format=raw \
  -nographic
```

if you want to understand this line you can go read the man page of this command using:

```
user@pcname:"yourpath"$ man qemu-system-arm
```

but it is not the goal of this report to explain that. But just to have the minimum, this line will allow us to connect to domoticz on qemu by listening to the port 8080.

Now qemu is launched, you can type root as login. If you type <http://localhost:8080> on a web browser, you will arrive on the domoticz page.

4 Put and test your OS on a raspberrypi 3B+

4.1 Configure your Embedded Linux

Ok, here we will configure and generate an OS for your raspberrypi 3/3B+ if you have one.

Always in the Development-tools folder, use the next copy command:

```
user@pcname:"yourpath"$ cp -R buildroot buildroot-rpi3-armhf-domoticz
```

Then go in the new folder buildroot-rpi3-armhf-domoticz and use:

```
user@pcname:"yourpath"$ make raspberrypi3_64_defconfig
```

to have the base configuration for your raspberrypi 3/3B+

The rest of this part will be quite similar to the previous part, but here we will also see how to configure the WIFI !

For now we will keep the crossed compilation chain generated by Buildroot. First change to do is within the Toolchain menu, choose glibc instead of uClibc. Next, select Lua in "Target packages -> Interpreter languages and scripting" and Domoticz in "Target Package -> Miscellaneous" in order to add domoticz to our Linux. As before don't forget to put "(200M) exact size" instead of 120M in Filesystem images.

Then we will add the necessary packages for the internet connection!

Go in "Target packages" then "Networking applications" and select openssh, iw, wpa_supplicant, dhcpcd and dhcpcdutils. Always in "Target packages" go in Hardware handling and Firmware, select Install DTB overlays and rpi-wifi-firmware.

We are soon at the end of the configuration, go on "System configuration" menu, then "/dev management", enable the "Dynamic using devtmpfs + mdev" option. Finally, you have to modify the file board/raspberrypi3/post-build.sh in your folder buildroot-rpi3-armhf-domoticz, adding the three following lines:

```
#!/bin/sh

set -u
set -e

# Add a console on tty1
if [ -e ${TARGET_DIR}/etc/inittab ]; then
    grep -qE '^tty1::' ${TARGET_DIR}/etc/inittab || \
        sed -i '/GENERIC_SERIAL/a\
tty1::respawn:/sbin/getty -L  tty1 0 vt100 # HDMI console' ${TARGET_DIR}/etc/inittab
fi

cp package/busybox/S10mdev ${TARGET_DIR}/etc/init.d/S10mdev
chmod 755 ${TARGET_DIR}/etc/init.d/S10mdev
cp package/busybox/mdev.conf ${TARGET_DIR}/etc/mdev.conf
```

That will allow to start mdev every time your system turn on by copying the mdev startup script in /etc/init.d, mdev is necessary for the peripheral devices.

One more thing: go in Target packages -> libraries -> Hardware handling end activate wiringpi at the end.

Note: you can add a password for the root user in the "System configuration" menu

Finally!!! Your configuration is finished, it is time to use the command:

`user@pcname:"yourpath"$ make` in the folder buildroot-rpi3-armhf-domoticz and wait until it is over

4.2 Put our image on the sd card

go in the folder "your_path"//Development-tools/buildroot-rpi3-armhf-domoticz/output/images/, find the name of your SD card and use the following command:

```
user@pcname:"yourpath"$ sudo dd if=./sdcard.img of=/dev/"sd_card_name"
```

when the copy is finished use mount the two partitions on two folders you can create:

```
user@pcname:"yourpath"$ cd /mnt
```

```
user@pcname:"yourpath"$ mkdir boot system
```

```
user@pcname:"yourpath"$ sudo mount /dev/sdb1 /mnt/boot
```

```
user@pcname:"yourpath"$ sudo mount /dev/sdb2 /mnt/system
```

```
user@pcname:"yourpath"$ sudo du -sh /mnt/boot/ /mnt/system/
```

you will see the size of each partition with the last command

4.3 Modify the network configuration

Add the Wlan0 interface for the WIFI modifying the `/mnt/system/etc/network/interfaces` file:

```
user@pcname:"yourpath"$ sudo emacs /mnt/system/etc/network/interfaces
```

and add these lines at the end of the document, remember to replace `<ssid>` by your WIFI login:

```
# interface file auto-generated by buildroot

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
    pre-up /etc/network/nfs_check
    wait-delay 15
    hostname $(hostname)

auto wlan0
iface wlan0 inet dhcp
    wireless-essid <ssid>
    pre-up wpa_supplicant -B w -D wext -i wlan0 -c /etc/wpa_supplicant.conf -dd
    post-down killall -q wpa_supplicant
```

then edit the file `/mnt/system/etc/wpa_supplicant.conf` like that:

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1

network={
    ssid="<ssid>"
    scan_ssid=1
    proto=WPA RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    group=CCMP TKIP
    psk="<wpa_passphrase>"
}
```

Replace `<ssid>` by your login WIFI and `<wpa_passphrase>` by your WIFI password.

The last thing to do is to edit the file `/mnt/system/etc/ssh/sshd_config` and add the line: `PermitRootLogin yes`

```
# Authentication:
#LoginGraceTime 2m
#PermitRootLogin prohibit-password
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

4.4 Optional: put your keyboard in AZERTY

Go on `/mnt/system/etc` and use the command:

```
user@pcname:"yourpath"$ sudo busybox dumpkmap > azerty.kmap
```

This one will create a configuration file from your own computer keyboard and save this file in the `etc` folder on your SD card.

Now go to `/mnt/system/etc/init.d` and create a file named "S15keyboard" for example, with this script:

```
#!/bin/sh
#
# Keyboard AZERTY
#
case "$1" in
  start)
    echo -n "Keyboard disposition : AZERTY: "
    loadkmap < /etc/azerty.kmap
    echo "OK"
    ;;
  *)
    echo "Usage: $0 {start}"
    exit 1
esac
```

Don't forget to give the execution right:

```
user@pcname:"yourpath"$ chmod 755 S15keyboard
```

In that way your keyboard will be put in Azerty at each startup

Now you can properly unmount your SD card:

```
user@pcname:"yourpath"$ sudo umount /mnt/boot /mnt/system
```

4.5 Configuration script

As you can see, if you have to make several images to make some tests it could be quickly boring to add the lines to each file, to mount and dismount the SD card..etc. It is why we decided to create a script which will do all the manipulation of the part 4.1 and 4.2! You can find this script here: [Configuration script](#)

We called it ImConf.sh and there are four parameters to give it, but don't worry if you don't give the good parameters the scripts will give an example of how to use it. One last thing, this script has to be used in root, so use sudo if you aren't root.

4.6 Test your image !

In order to test your image it will be easier to have a screen and a HDMI cable

Firstly, put your SD card in the raspberrypi, supply it and connect it to a screen with the HDMI cable

Normally, if there is not any problem, your system will start writing root as login and then your password

When you are connected use: ifconfig to have the IP adress of your raspberrypi

Futhermore, you can go to a web browser and write root@"Ipadress" to connect to your domoticz.

Of course, if you are not in the same network than your raspberry, you must connect to your wifi box with the good port which is initially 8080 for domoticz.

4.7 Add user

When you are in buildroot go to /etc/passwd and add the line "user":2000:100:root:/tmp:/bin/sh , save and exit, "user" is the name you want as a user. Now use the command: passwd "user" , replace "user" by the name you chose before, you will have to choose a password and now you can use this "user" as login with the password you chose !

4.8 Connect with ssh using Keys

On your own computer, open a terminal and use:

```
user@pcname:"yourpath"$ ssh-keygen
```

Go in the following folder: `~/.ssh` and use:

```
user@pcname:"yourpath"$ ssh-copy-id root@"AddrIp"
```

Now you can just use `ssh root@"AddrIp"` and you are connected to your raspberrypi without using a password.

5 Configure and test your Domoticz

5.1 Check that your domoticz is running

At the moment, your buildroot is running on your raspberry card. The first thing to do is to verify that domoticz is running. One way to do that is to use this command:

```
# ps aux | grep domoticz
```

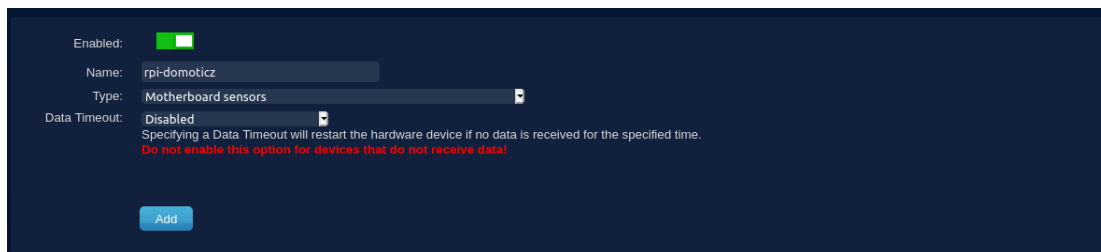
normally you will see the following line:

```
/opt/domoticz/domoticz -daemon -www 8080 -sslwww 443
```

which means that domoticz is running !

5.2 Add sensors in domoticz

An easy thing to do is to add the sensors of the raspberry card. In order to do that, go in Setup then hardware and choose Motherboard sensors for the line type. Give the name you want, for example "Rpi sensors". So you will have something like that:



Enabled: ☒

Name: rpi-domoticz

Type: Motherboard sensors

Data Timeout: Disabled

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

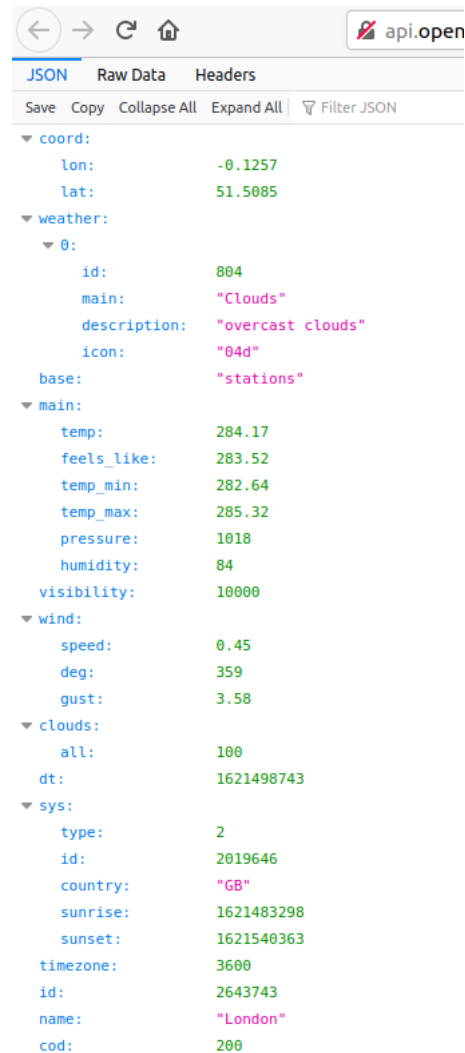
Add

Now press add and you can see your sensors and their data in the tabs temperature and utility.

Those sensors were pretty easy to add, now we will try to add weather sensors, to have data about the wind, rain, temperature of the location you want.

The first thing to do is to find a weather API to collect Json data. This site gives free API that you can call 1 time per second maximum, which is largely sufficient for us: <https://openweathermap.org/>

When you are on the site press API at the top, then, press subscribe for "Current Weather Data", next click on Get API Key in the column "Free". Now create an account and you will receive an email with your API key and an example of how to use it. You use your API key to see how the data looks, you should have something like that:



You can see that there are several data, in our case we will focus on the temperature, wind and rain/clouds... So we need to process this data and we will use a lua script to do that. Domoticz already have scripts to parse Json data. In your buildroot, go in `/opt/domoticz/scripts/lua_parsers`

There is a file called "example_json.lua" copy this file and rename it to weatherJson.lua for example. Modify it to have that:

```
-- Retrieve the request content
s = request['content'];

-- Update some devices (index are here for this example)

local temp = domoticz_applyJsonPath(s,'main.temp')

local wind = domoticz_applyJsonPath(s, 'wind.speed')
wind = wind*1000/3600

local gust = domoticz_applyJsonPath(s, 'wind.gust')
gust = gust*1000/3600

local deg = domoticz_applyJsonPath(s, 'wind.deg')

local t = domoticz_applyJsonPath(s,'weather[0].main')

domoticz_updateDevice(10,'', t)
domoticz_updateDevice(11,'', temp-273.15)
domoticz_updateDevice(13,'', deg .. ";" .. "degree" .. ";" .. wind*10 .. ";" .. gust*10 .. ";" .. 22 .. ";" .. 24)
```

This script collects the data from the Json, put it in a var and finally update a sensor we will create soon with the value of the var.

Let's return to Domoticz. Go again into Setup then Hardware and this time choose "HTTP/HTTPS poller". Give the name you want and select Get as "Method" and Json as "Content Type". In URL, copy the URL which contains your API key, as it is explained in the mail you can choose the location you want by replacing "London" and "uk" by the city and country you want. In our case it is the city Melle in France(fr). Finally, write the name of your lua script in "Command" and frequency of refresh you want. Remember the maximum frequency of refresh is 1 second. At the end you have something like that:

Enabled: ☒

Name:

Type:

Data Timeout:
 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!

Method:

Content Type:

Headers:

URL:

Command:

Refresh:

Username:

Password:

Add and now we have to create our sensor. Always in the Hardware tab you can see that there is something called "create virtual sensor" in the line you added earlier. Click on it! Firstly we will create a temperature sensor, named it as you want again, choose Temperature as "Sensor Type" and click on "OK". Do it again to create the sensor for the rain/clouds/sun..., this time choose Text as "Sensor Type". Finally, create one for the wind, choose Wind as "Sensor Type". At this moment you can see your sensors in the tabs at the top and you will see that they will be updated by the Lua script we created before.

5.3 Add actuator (LED)

In this section we will use LED to simulate actuators as we don't have things like connected lights or connected sockets. The idea is simple, we will turn on a LED if the temperature in the city you chose is higher than 10°C, another if it rains and a third if the wind is faster than 10km/h for example.

Firstly go in `/etc/init.d` and create a file called `S20gpio` which has this content:

```
#!/bin/sh

case "$1" in
    start)
        echo -n "gpio conf"

        echo 17 > /sys/class/gpio/export
        echo out > /sys/class/gpio/gpio17/direction

        echo 18 > /sys/class/gpio/export
        echo out > /sys/class/gpio/gpio18/direction

        echo 23 > /sys/class/gpio/export
        echo out > /sys/class/gpio/gpio23/direction

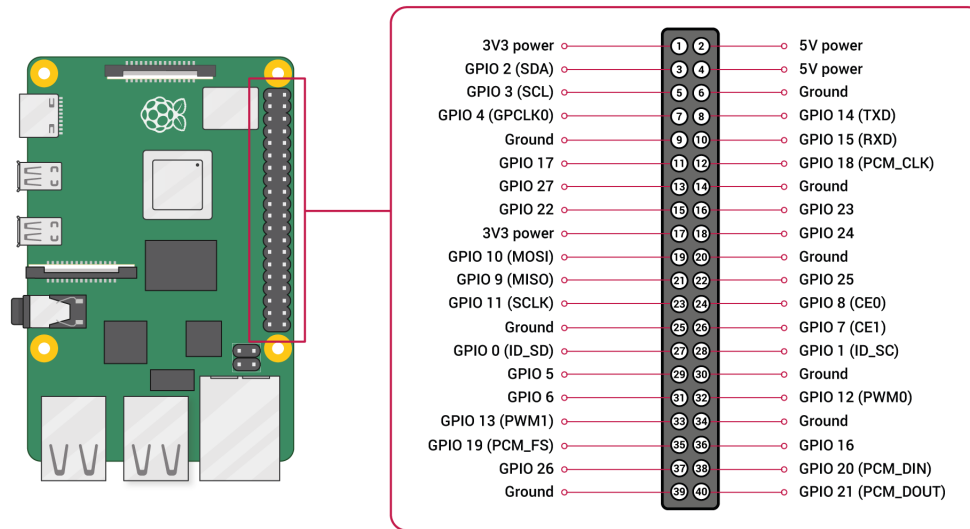
        echo 24 > /sys/class/gpio/export
        echo out > /sys/class/gpio/gpio24/direction

        echo "OK"
        ;;
    *)
        echo "Usage : $0 {start}"
        exit 1
esac
```

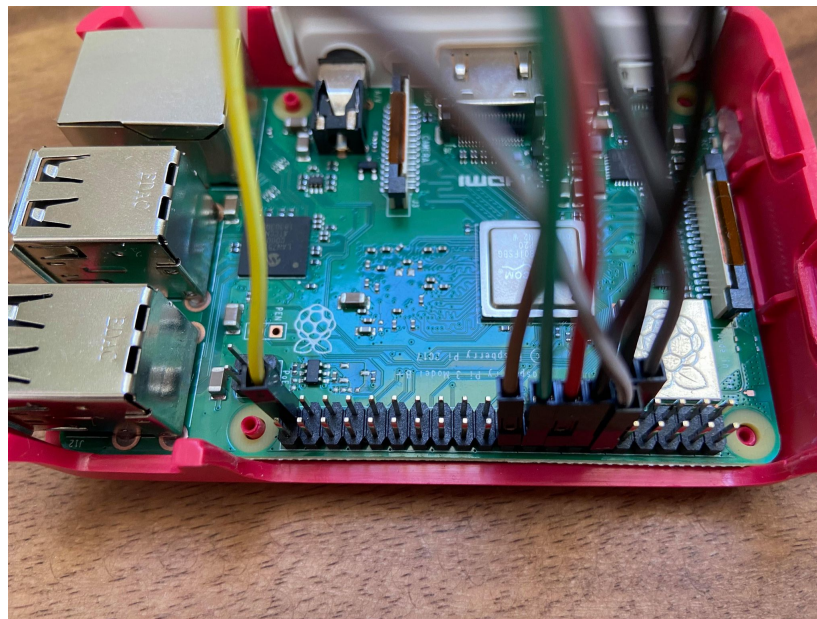
Don't forget to give the execution right ! This file will initiate four pins to output when the OS started, but if you create this file while the system is running, start the script on your own using:

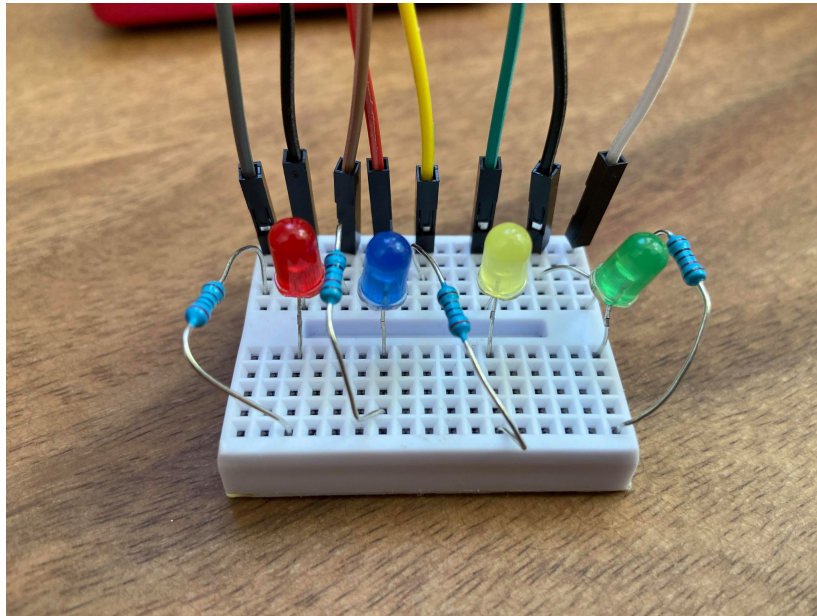
```
# ./S20gpio
```

At this point you are able to use your pins, let's connect four LEDs, three for the weather as explained before and one for the internal temperature of your raspberry. Below is the description of the raspberry 3B+ gpio:



And following is the simple circuit we did:





The rest of the job is in Domoticz. Go in Setup->Hardware and create and add a new hardware like this one:

Enabled: ☒

Name: GPIO

Type: Raspberry's GPIO port

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!

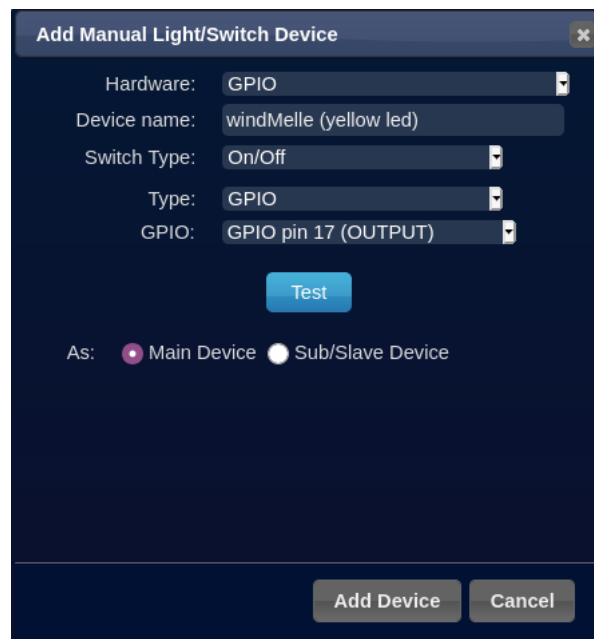
Debounce: 50 (Milliseconds)
Make sure the correct pin status is read by introducing a bounce delay.

Period: 50 (Milliseconds)
Maximum of one interrupt per pin allowed in this period (prevents interrupt storms).

Poll interval: 0 (Seconds) 0 = Disabled

Add

Go in the tab "Switches", click on "Manual Light/Switch" and configure it as follows:

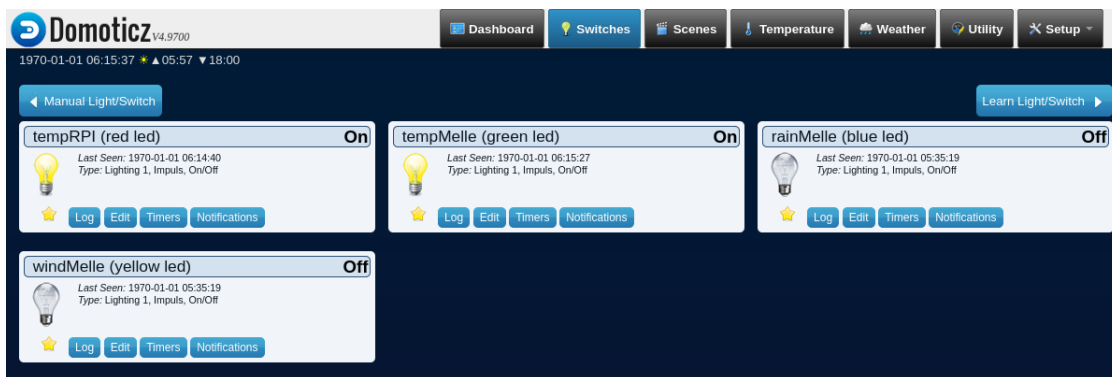


The screenshot shows a dialog box titled "Add Manual Light/Switch Device" with a close button (X) in the top right corner. The form contains the following fields:

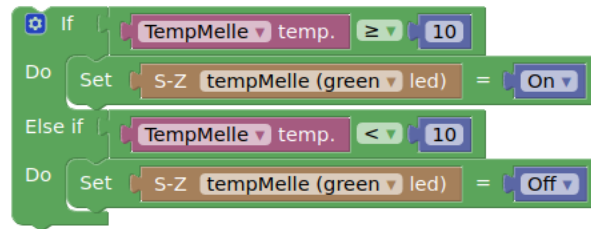
- Hardware: GPIO
- Device name: windMelle (yellow led)
- Switch Type: On/Off
- Type: GPIO
- GPIO: GPIO pin 17 (OUTPUT)

Below these fields is a blue "Test" button. At the bottom, there is a section labeled "As:" with two radio buttons: "Main Device" (selected) and "Sub/Slave Device". At the very bottom of the dialog are two buttons: "Add Device" and "Cancel".

Do it four times to have that at the end:



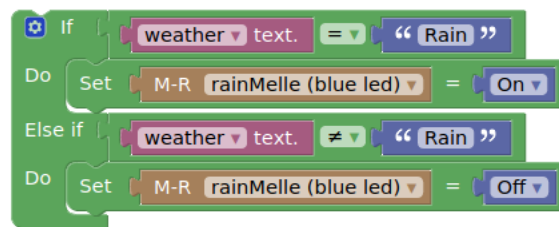
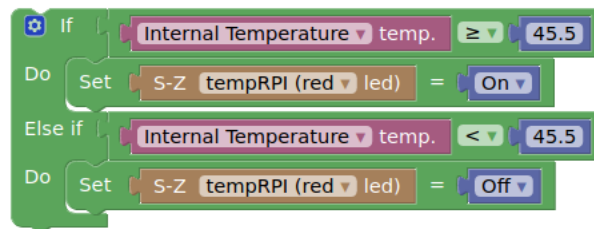
The last thing to do is to go into Setup->More Options->Events. Here you can use logic blocks to create your events. For example, for the internal temperature we did that:



This code means that the LED will be on when the temperature in the city of Melle in France is superior or equal to 10 degrees and off if inferior to 10 degrees.

Do not forget to turn on "Event active" and finally to save and your event is working!

It is the almost the same for the three others as you can see below:





The led will now turn on or off in function of the values you chose. Of course to do real domotic in your house, it need connected object which would be connected with RF,Z-wave,...etc, to domoticz and which would have replaced the leds to do things like turn off lights if there is enough sun or close windows if the temperature are too low or the wind too strong. Of course, it also needs connected sensors to do real domotic like connected interrupter or luminosity sensor for example. We can't buy all these things for this project, it is why we use sensors available for everyone.

6 Application and access to our Domoticz

You can find an the Domoticz application on IOS and android, it is free. When you are in the application you just need to enter the IP address and port to connect to your domoticz and you can check the sensor data, turn on or off leds...etc.

If you want to look at our Domoticz, you can try to access it at the address: soniksub.freebox.fr:5050 or <https://192.168.1.11:8080> when our raspberry is running, of course. You will see something like that:

