



REMOTE HEATER CONTROLER

Term paper submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Engineering at the University of Applied Sciences Technikum Wien – Degree Program Electronic and Business



Author: Jonatan Rubio Bueno

Student number: 1500255001

Advisor: Christian Kollmitzer

Wien, 26-1-2016

Declaration of Authenticity

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (for example see §§ 21, 46 and 57 UrhG (Austrian copyright law) as amended as well as § 11 of the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

In particular I declare that I have made use of third-party content correctly, regardless what form it may have, and I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see § 11 para. 1 Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.”

Place, Date

Signature



Table of contents

Declaration of Authenticity	1
1.- Abstract	3
2.- State of the art	4
3.- Project description	7
3.1.- Conclusions	8
3.2.- Choice of Equipment.....	9
4.- Hardware.....	10
4.1.- Raspberry pi Model B+.....	10
4.2.- Temperature sensor.....	11
4.3.- Relay.....	12
4.4.- WiFi USB adapter	13
4.5.- External Components.....	13
4.6.- Construction Progress.....	14
4.7.- Bill of materials	17
5.- Software	18
5.1.- Coding tools and software used.....	18
5.2.- Getting started with the Raspberry	20
5.3.- Collecting the data from the sensor	22
5.4.- Installing LAMP.....	24
5.5.- Creating the database.....	27
5.6.- Security	29
5.7 Summarize of the code	30
6.- Economic study	31
7.- Future lines	32
Bibliography	33
List of Figures	34
List of tables	35
List of abbreviations.....	36
A.- User Guide	37
B.- Useful commands for the Raspberry	39
C.- Code	40



1.- Abstract

In the society everything is connected. We are starting to develop smart appliances and even creating completely smart houses, which is a house where all the operations can be efficiently controlled by a unified ubiquitous application [1]. Taking advantage of the rise of the smartphones we are going to create a heating system remotely controlled over the internet. The device will be useful, because it allows the monitoring the temperature or connect/disconnect the heating system without being present in your home which is a gain of comfort for the user.

This project will have two different parts:

- The first one will be the hardware. This part includes making the controller, the heating sensor and all the necessary circuits to run the device.
- The second part will be the software. Here we have different applications, the first one includes the programming for the embedded system and the communications. The second part will be related to the code of the website.

One of the goals of the project is to build a device with a low budget and high modularity which will bring in the future the opportunity of improving the life with the minimum cost and effort.

Keywords: Heater, Remotely, Controlled, Device



2.- State of the art

In this section we are going to explain the performances, settings, prices of similar devices in the market.

1. The first device is the 'Thermostat' from the company NETATMO:



Figure 1: Thermostat by NETAMO

This device has two different parts, the first one is the thermostat (Figure 1) which controls the device and the second part is the relay. The communication between them is realized by radio frequency.

The thermostat dimensions are 83x82x22mm, what makes it small, but this is without the relay part. Both parts need batteries to function and requires installation. The maximum supported current is 4 Amperes.

It has a schedule programming, with different programs, and a free application on the play store available for download.

The price of the complete device is 179 Euros [2].

The possibility of programing schedules makes this device really interesting and also has a friendly interface, but will have problems with the radio frequency communication in industrial environments. It also needs batteries, has a low current working input, needs an installation and has a high price.



2. The second device is 'WeMo insight' from the company Belkin:



Figure 2: WeMo insight by belking

This device can be plug into a socket, it is controlled with a smartphone over wireless network.

It is a small device (socket size) and does not require installation. The maximum current input is 16 Amperes.

It also has a schedule programming, with a free application on the play store.

The price of the device is 59.99 Euros. [3]

Small appliance, with a reduced price and no installation. But it does not have any kind of feedback or sensor to measure the temperature. Also it cannot be used on industrial environments or environments, where humidity is high.



3. And the third is 'GSM heater controller' from the company 4UControl (Figure 3):



Figure 3: GSM controller by 4UControl

It is pretty similar to the 'WeMo insight' device, but the communication between the user and the appliance is made using GSM. It does not come with an interface, but it is controlled using SMS.

The maximum current input is 16 Amperes, and cannot work in industrial or humidity environments.

The price is 139.95 Euros.[4]

It has the same problem with the feedback as the 'WeMo insight' and a SIM card is need for functioning. The price is also high.

To summarize, there are devices in the market with the possibility of making a schedule programming of the heating cycles and are small in size, but they have several troubles with adverse environments, high prices, installation and batteries or lack of feedback.



3.- Project description

During this project I am going to explain, how I created and developed the remote heater controller.

First of all, a block diagram of the project is going to be presented. As I wrote in the previous chapters, the project will have two different parts, namely the hardware part and the software part.

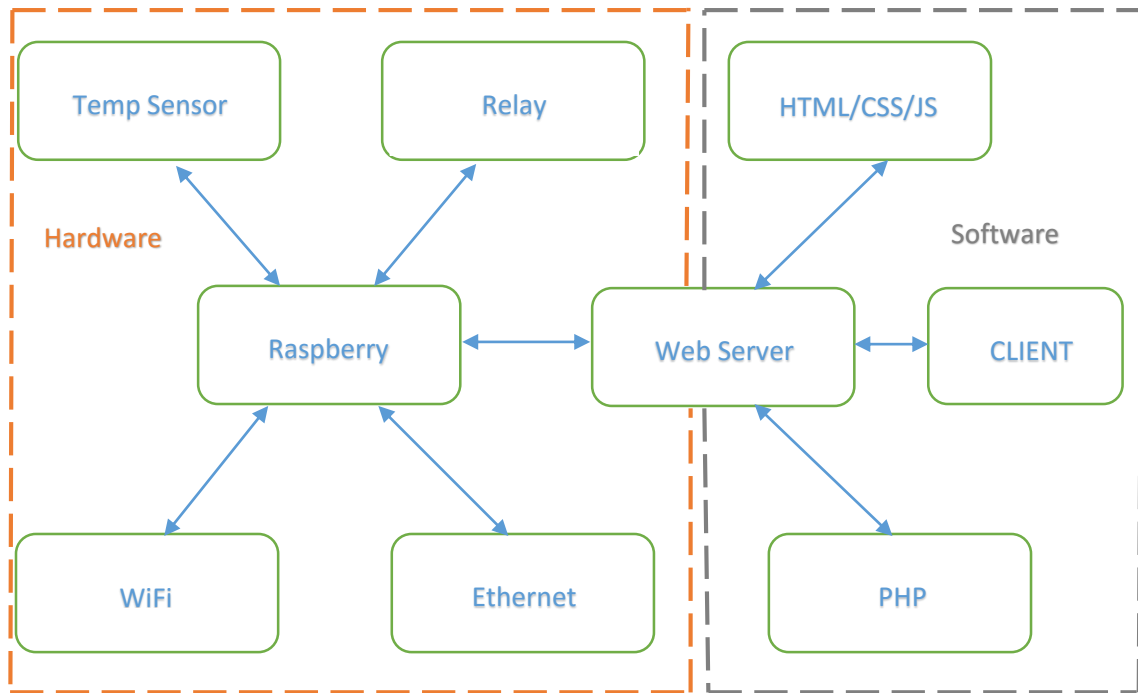


Figure 4: Block diagram of the project

As we can see in the figure 4, the raspberry is the main controller of the whole device. It controls the relay for turning the heater on and off and it also receives the incoming data from the temperature sensor and stores it.

For the communications, the user will have two different options:

- With an Ethernet cable, connecting the raspberry and the router.
- Wireless connection with a WiFi Module connected into the raspberry.

In the software part, we have two fragments, the code used for the Raspberry and the code used for the web server, web page and communication.

- The first one is coded in python and it is used for controlling the relays, read the data from the sensor and adapt it for the database and also security programs in case the device goes wild.



- The second one is related with the client, how the webpage looks like, which settings it has available and how much data is available to the user. The code is written in combination of HTML5, CSS and JavaScript.

There is also a piece of the code for the database, which hosts all the security for the web page and the temperatures from the sensor. This is done in PHP.

Now we have an overall vision of the project and we can explain the physical device in detail.

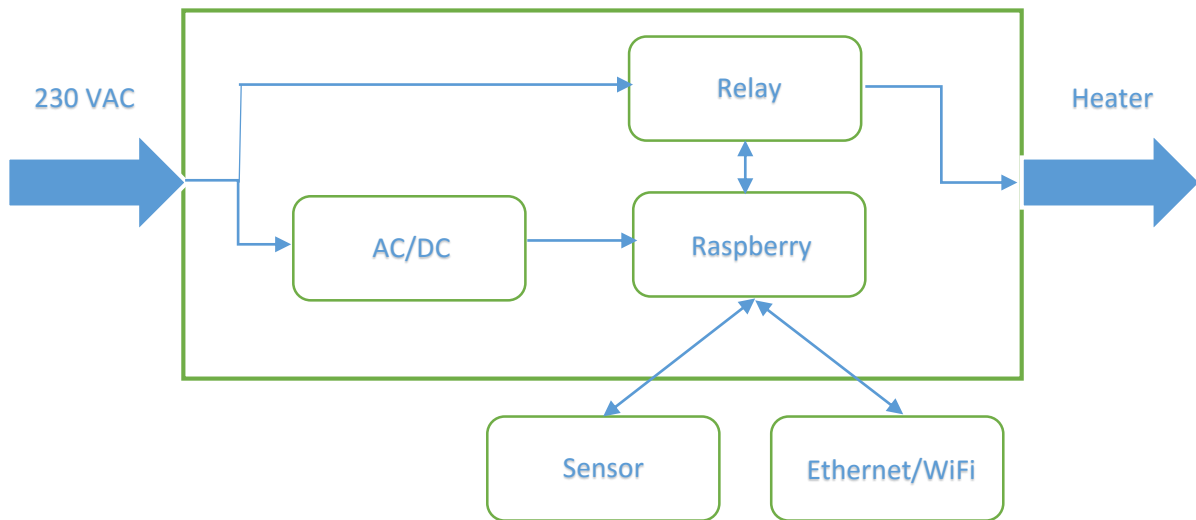


Figure 5: Block diagram of the device

We can see the wiring inside the box in the figure 5. We have a Schuko connector (Usual connector for 230 VAC single-phase) which divides into two wires.

On one hand provides energy to the AC/DC convertor, and this one converts the alternate current into direct current and provides the output for supplying the raspberry.

On the other hand it connects the 230 VAC with the relay, allowing it whether let the current pass or not, which is being controlled by the raspberry.

Moreover there are two wires leaving the raspberry, the sensor wire and the communication wire. Both are outside of the box. One is used for measuring the exterior temperature and the other is used for the internet connection. In case we use the WiFi instead of the Ethernet, the user does not have any wire going out except for the sensor.

In the end we have a socket wired with the relay, which supplies the heater with the energy in case the relay is open.

3.1.- Conclusions

Packing every part of the device in the same box has many benefits. Power for everything is supplied by socket, so the device does not need any kind of batteries or maintenance by the user.



Beside this also provides a lot of isolation against EMI (Electro Magnetic Interference) and this is perfect if we want to install the device in an industrial environment.

In addition the appliance does not require hardware installation by the user, the user only has to plug it to the current and connect the heater to it.

3.2.- Choice of Equipment

For the choice of the controller of the device, we were deciding between two different ones: Raspberry and Arduino.

Arduino [5] is a board with a micro-processor (Figure 6) with his own programming language (similar to C++), and input/output pins. It is perfect for controlling output such as motors, LEDs, servos and for reading the inputs like sensors, potentiometer, etc. Due to its analog inputs, the arduino is able to read the signals.



Figure 6: Arduino Uno

The Arduino has more problems when it tries to communicate with other devices, because it needs to use a specific shields and libraries for it, while some of them are not easy to use.

On the other hand, the Raspberry is easy to use for communications with an incorporated Ethernet connection and an easy configuration for the WiFi. The problem with the raspberry is that it does not have analog inputs, which prevents us to use analog signals in the sensor.

Moreover the raspberry has a more powerful processor than the Arduino, has more security for hosting an internet application and gives more possibilities of expansion in the future (you can connect an Arduino and a Raspberry).

Therefore for this project the optimal choice is the raspberry, because we need to communicate with the client, we want a strong security and we do not have many inputs or outputs to control.

Likewise for the temperature sensor we had to decide between two options the TMP35 and DS18B20, the first one is analog and the second is one wire digital.

For the raspberry the best choice is the DS18B20 because if we take the TMP35, will need additional hardware for reading the data (analog to digital converter).



4.- Hardware

In this chapter we are going to explain in detail every component needed to build the device, the building progress from the start to the end and a bill of materials.

4.1.- Raspberry pi Model B+

The Raspberry pi [6] is a single board computer (Figure 7) from the company Raspberry Pi Foundation.

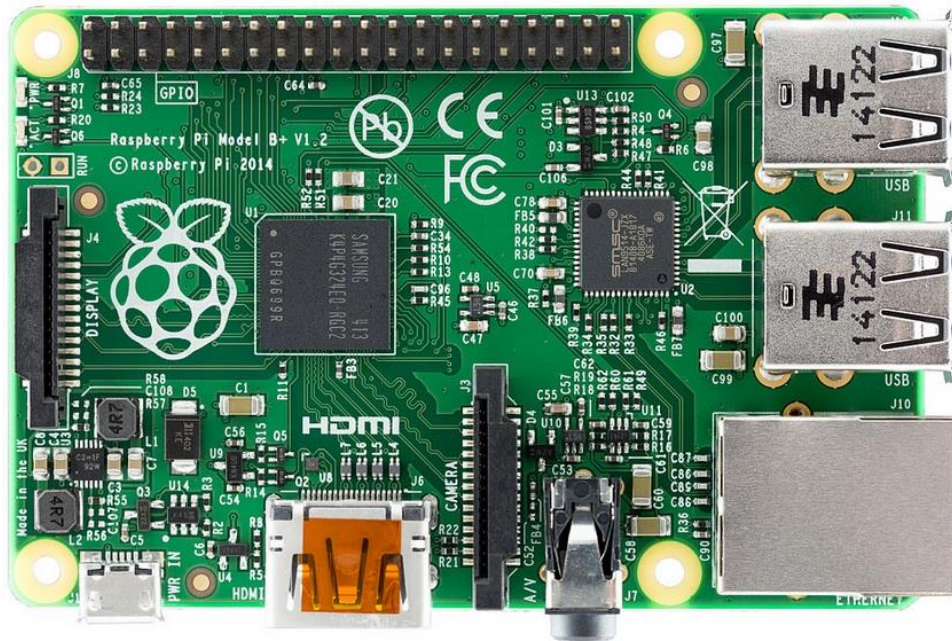


Figure 7: Raspberry Pi B+

It has an ARM architecture of 32 bits, a 700 Mhz speed processor (with possibility of overclocking to 1GHz) and 512 Mb of RAM memory. It also has one Ethernet port, four USB ports, one slot for micro-SD card and audio output. Moreover it has 40 GPIO (General purpose input output)

The power consumption of the Raspberry depends on what is connected to it, for this project the consumption will be between 3-3.5 W.

The hard drive of the Raspberry is the micro-SD card, which has at least a 4GB to save the operating system on it. The OS (operating system) is based on the linux core. We are using the Raspbian distribution from Devian designed for Raspberry.

We also need some parts for the Raspberry, such as:

- Wires
- Power supply
- Ethernet cable
- Wi-Fi module



4.2.- Temperature sensor

We need to get some kind of feedback to know if the heater is working properly. To achieve this we installed one temperature sensor, what also brings the opportunity of having a register of the temperature over time.

We used the DS18B20 sensor [7], it is a one wire digital sensor. There are diverse packages, but we chose the 3 pin sensor (Figure 8), because we do not need a PCB (printed circuit board) to hold it. There are also some waterproof packages, if it needs to be installed outside, in a water or high humidity ambient.

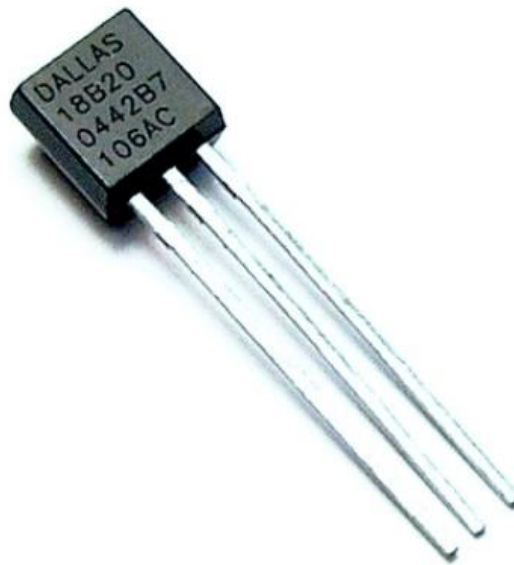


Figure 8: Temperature sensor DS18B20

There is only one pin for the communication (middle pin). The other two pins are used for the power supply. It has to be powered with 3.3 V output from the Raspberry and its average current consumption is 5 mA, which means a low power consumption of 0.02 W.

We need to install a 4.7 K Ω pull-up resistor between the 3.3 V wire and the communication wire to avoid noises in the signal and errors. Only one pull up resistor must be installed in case we use more than one sensor.

It has a 9 bit precision, so we can measure temperatures from -55 $^{\circ}\text{C}$ to 125 $^{\circ}\text{C}$ and it has 0.5 $^{\circ}\text{C}$ accuracy in the range from -10 $^{\circ}\text{C}$ to 85 $^{\circ}\text{C}$. This range of linearity is enough for measuring the temperature of the air or even the temperature of the radiator.

All the data is stored in a file inside the Raspberry and every sensor has a unique identifier, so we can use the same three wires for different sensors, because they will store the data in different files.



4.3.- Relay

A relay is an electromagnetic controlled switch, which allows to control high currents and voltages with low power.

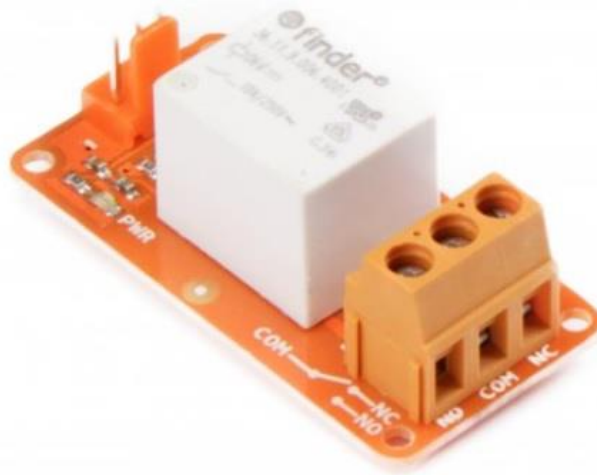


Figure 9: Tinker kit relay

The relay has three outputs (NC, NO, COM) and two inputs. If there is no voltage in the input, COM and NC are connected. If there is 5 V in the input, COM and NO are connected in this case.

The maximum rates for the output of the relay are 250 VAC and 10 A and for the input 5 VDC and 10 mA, so we need to ensure that nothing connected to the output requires values above this rates.

We have a tinker kit relay [8] in the figure 9. It is a normal relay, but it also has two LEDs and an optocoupler. One of the LEDs is to tell us, if the relay is powered (PWR LED). The other one indicates the status of the output. If the LED is off, there is no voltage in the input, and if it is on, there is voltage on the input.

The relay by itself isolates the high voltage from the microcontroller, but if something fails, the voltage on the control part could be high and potentially damage the controller, in our case the Raspberry. To avoid that we use the optocoupler.

An optocoupler is a security component. It transfers an electrical signal with light, isolating the components on the both sides of the optocoupler. It uses a LED for emitting the light and a phototransistor for receiving it (Figure 10).

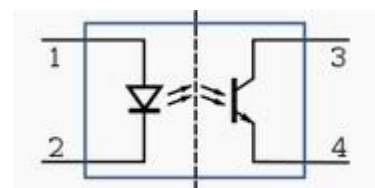


Figure 10: Optocoupler diagram

We are going to use two tinker kits for switching the power for the heater. We are going to use two of them, because we need to cut the phase and the neutral cables for safety reasons, because if you cut only one of them, the current can return over the safety ground cable and shock the user.



4.4.- WiFi USB adapter

A WiFi module is needed, because the Raspberry does not have one. The best way to have WiFi in the raspberry is an USB adapter, since we have four USB slots in the Raspberry.

We took the RTL8188CUS chipset [9] integrated onto a USB adapter, because it works perfectly with the Raspberry and does not require a long installation process.

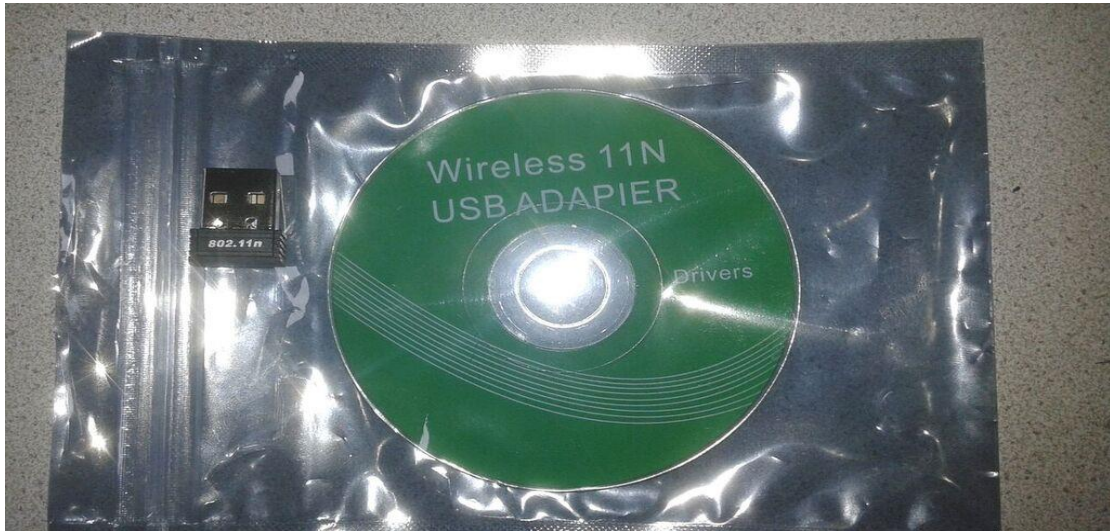


Figure 11: Wifi USB adapter

4.5.- External Components

We need something to hold all the previous parts. For that we are going to use a box. We also need to provide an electrical connection, for which we will include a plug and a socket.



Figure 12: Box, socket and plug



In the figure 12 we can see the box, the socket and plug used for the project. One of the main goals of this components is to isolate the electrical parts from the user.

The box is made of ABS plastic, which is isolating. Besides this is easy to work with for installing the socket and making the holes for the cables.

The IP protection degree is 54 that means the dust will not interfere with the electric parts inside and it can hold the water entrance, if there is not a big stream pointing at it. Due to this we can use this device outside and also in an industrial environment.

The socket has an IP54 and a ground connecter for security reasons. The maximum rates are 250 VAC and 16 A. This is higher current than the relay current so we do not have any problem with the connection.

The plug is a Schuko (Schutzkontakt), which has the same features as the socket.

4.6.- Construction Progress

First of all we need to make the holes for the socket. To do this, we are going to use a special tool on the driller to increase the size of the holes it creates.



Figure 13: Hole for the socket

After this, we need to make a hole for the power cable and another for the Ethernet/sensor:



Figure 14: Holes for the cables



When we have the external holes, we make the holes for holding the tinker kit relay and the Raspberry.

As we can see in the figure 15, we have four holes with screws inside, prepared for installing the two tinker kit relays.

Moreover we have the socket with the cables attached to it and also a cable grommet in the hole for the power cable. We used a cable grommet, because if we only have the cable going through the hole, we lose the isolation and also the user pulling the cable could break the joint.

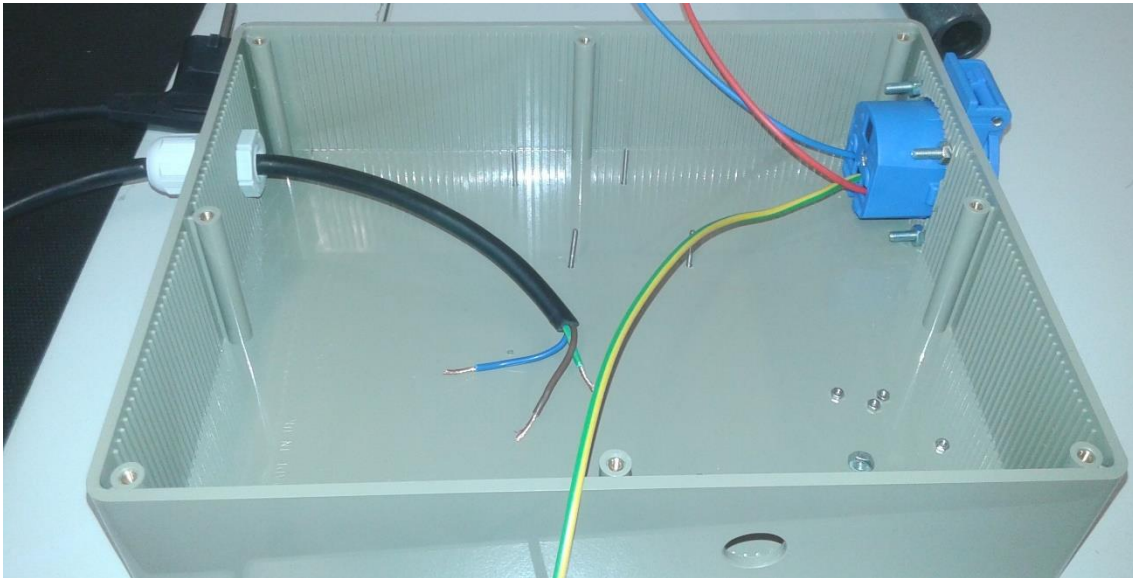


Figure 15: Installation of the socket and plug

We proceed to install the two tinker kit relay and attached the cables to them. The ground cable connects directly the plug to the socket.

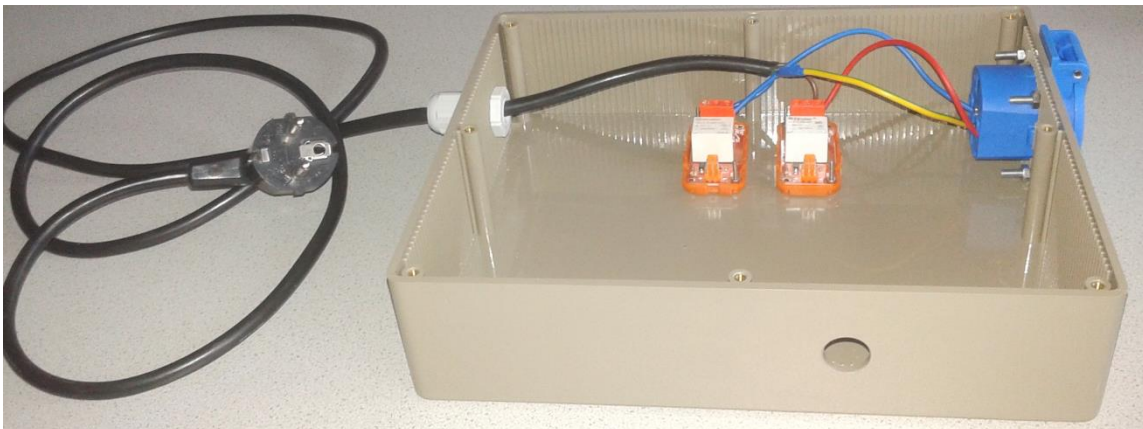


Figure 16: Installation of the relays

After that, we need to install the cable grommet for the other hole. Inside that cable grommet we have the Ethernet cable and the three wires for the sensor.

The Ethernet connector does not fit in the cable grommet, so we have to cut the cable, pass it through the hole and install a new connector.



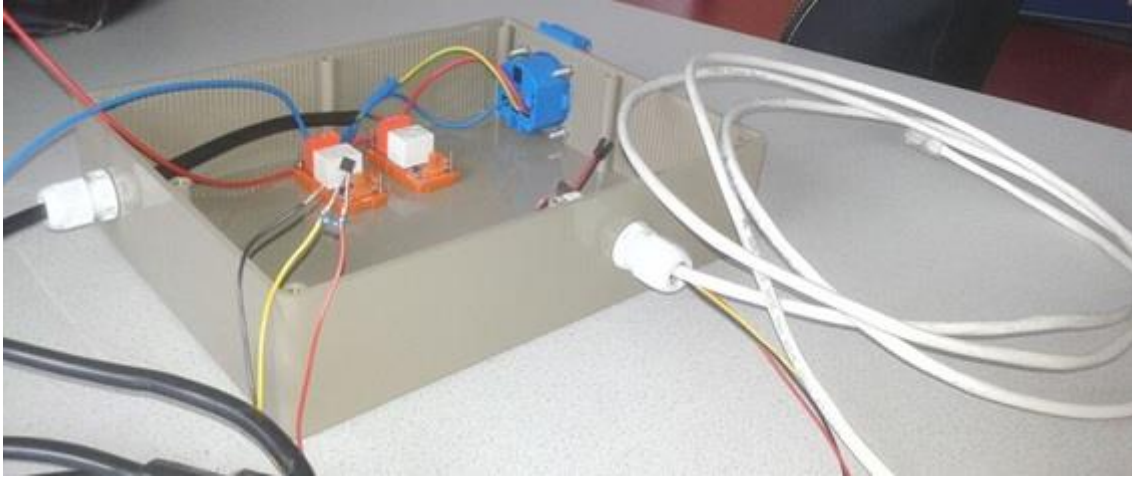


Figure 17: Installation of the Ethernet cable and sensor

When we have the sensor and the Ethernet cable ready, we proceed to install the Raspberry inside the box, attach the relay and sensor wires to it and also connect the Ethernet cable.

We can see in the figures 17 and 18 the pull up resistor soldered to the sensor. The resistor is soldered to the VCC (red) wire and the communication (yellow) wire.

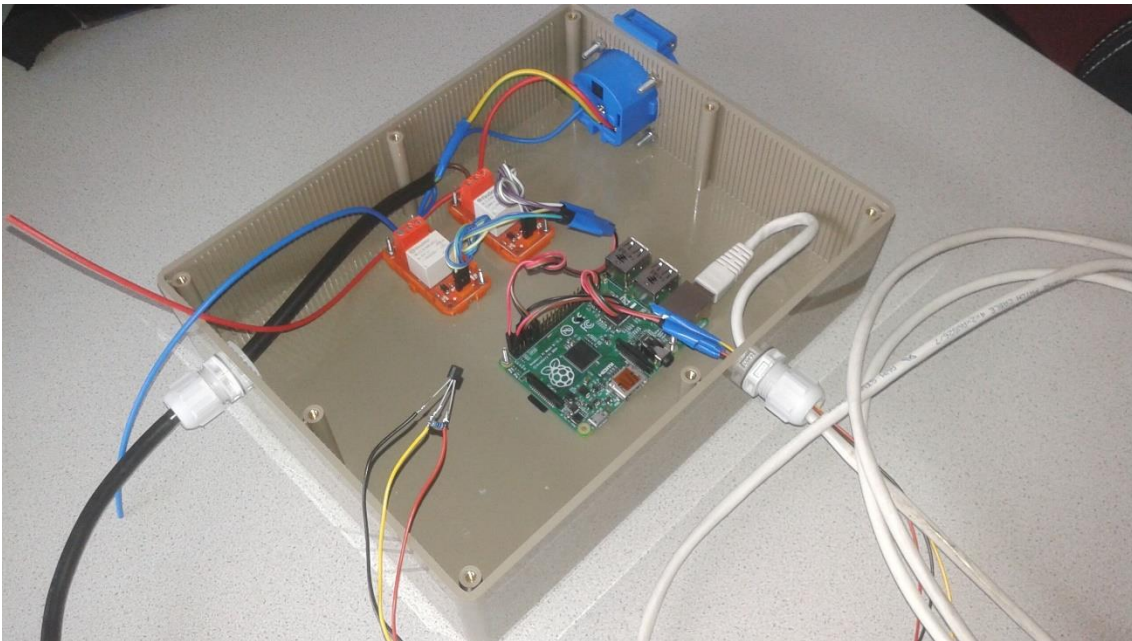


Figure 18: Installation of the Raspberry

Finally we install the AC/DC transformer into the box and connect it to the relay and then to the Raspberry to power it up (Figure 19).

The Raspberry is always powered if the plug is connected, because they are connected to the same spot in the relay.

After the hardware part of the project is complete, we only have to put the cover on the device (Figure 20).





Figure 19: Final device open

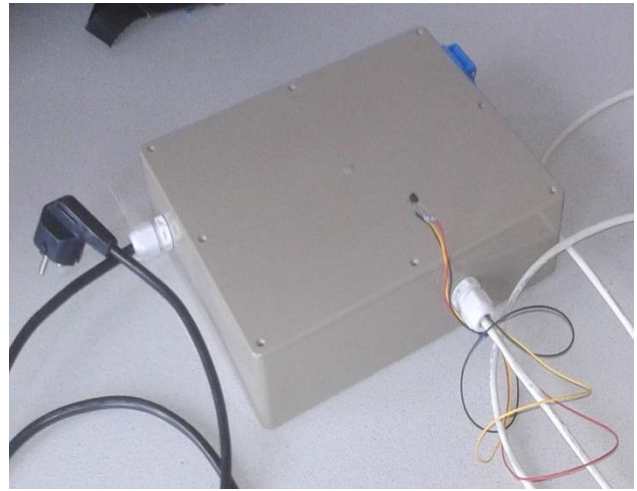


Figure 20: Final device closed

4.7.- Bill of materials

Here we have a table with all the necessary components to build this project.

Item NO	Description	Quantity
1	Raspberry pi Model B+	1
2	Tinker kit relay	2
3	DS18B20 temperature sensor	1
4	WiFi USB adapter	1
5	ABS box	1
6	Schuko plug	1
7	Socket	1
8	Screw M3	8
9	Screw M2	6
10	Screw M7	4
11	Cable grommet	2
12	10 Meter power cable	1
13	10 Jumper wire 150mm male	1
14	2,5 Meter Ethernet cable	1
15	SD Card	1

Table 1: Bill of materials



5.- Software

In this chapter we are going to explain all the software and tools that were used for coding in this project. Also we will explain the whole installation process of configuring the Raspberry, creating the webserver and the database.

5.1.- Coding tools and software used

- The Raspberry terminal: (Figure 21) this is an emulated terminal for controlling the Raspberry. The Raspbian OS is based on the UNIX/GNU core, so it is possible to use the UNIX commands to control the Raspberry.



Figure 19: Raspberry terminal

- Putty: it is a free SSH client to emulate the terminal, which is on figure 19. This program allows us to modify the files on the Raspberry without having physical access to the device.

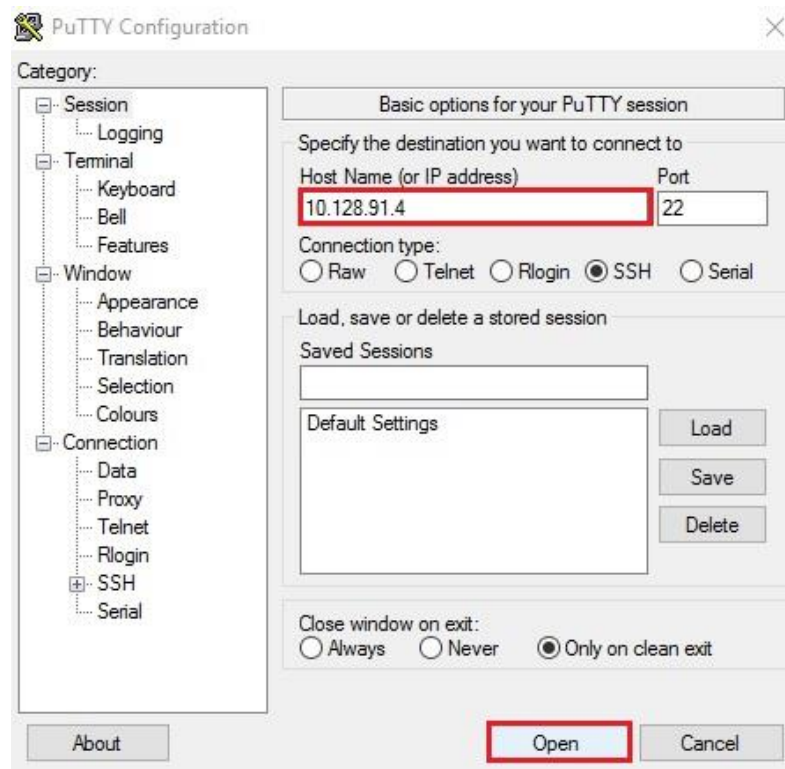


Figure 20: Putty interface



We can see in the figure 22 the interface of the putty. Inside the red box we put our Raspberry remote IP to create the connection and then we press open to start it. After this we need to log ourselves in the Raspberry and the terminal will appear.

- **Fillezilla:** it is a free FTP client to transfer files between the Raspberry and other devices such as a computer or tablets. Working with the Raspberry editor is sometimes difficult, so to overcome that we use an IDE for programming and we use Fillezilla to transfer the files to the Raspberry.

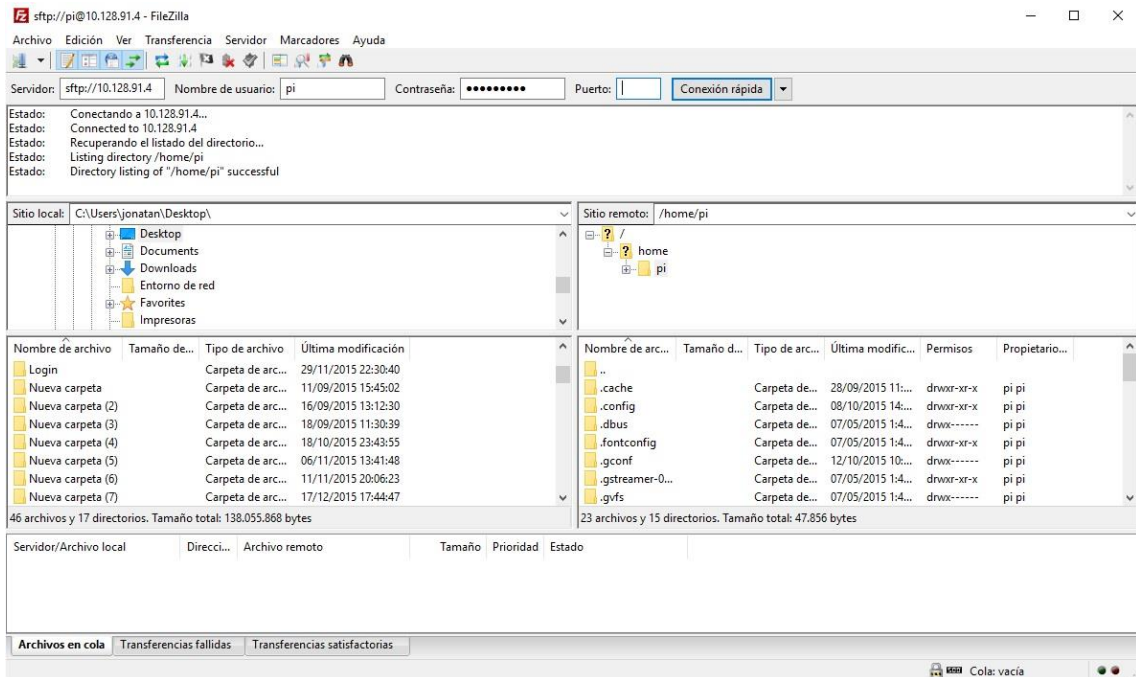


Figure 21: Fillezilla interface

- **SDFomater:** it is a small program to format SD cards, we need it to install the OS on our SD card, which will be used in Raspberry. We need to choose the SD card on the red square called "Drive" and press format to format it.

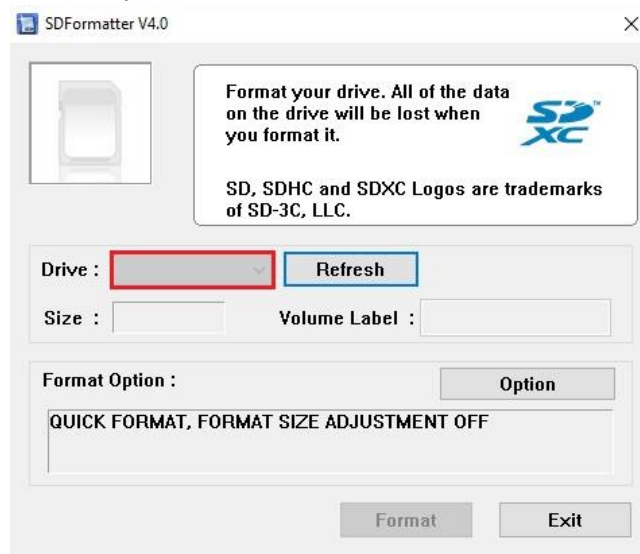


Figure 22: SDFormater Interface



- **NetBeans:** it is a free IDE (Integrated Development Environment). This IDE is mainly used for Java programming, but can also be used for PHP.

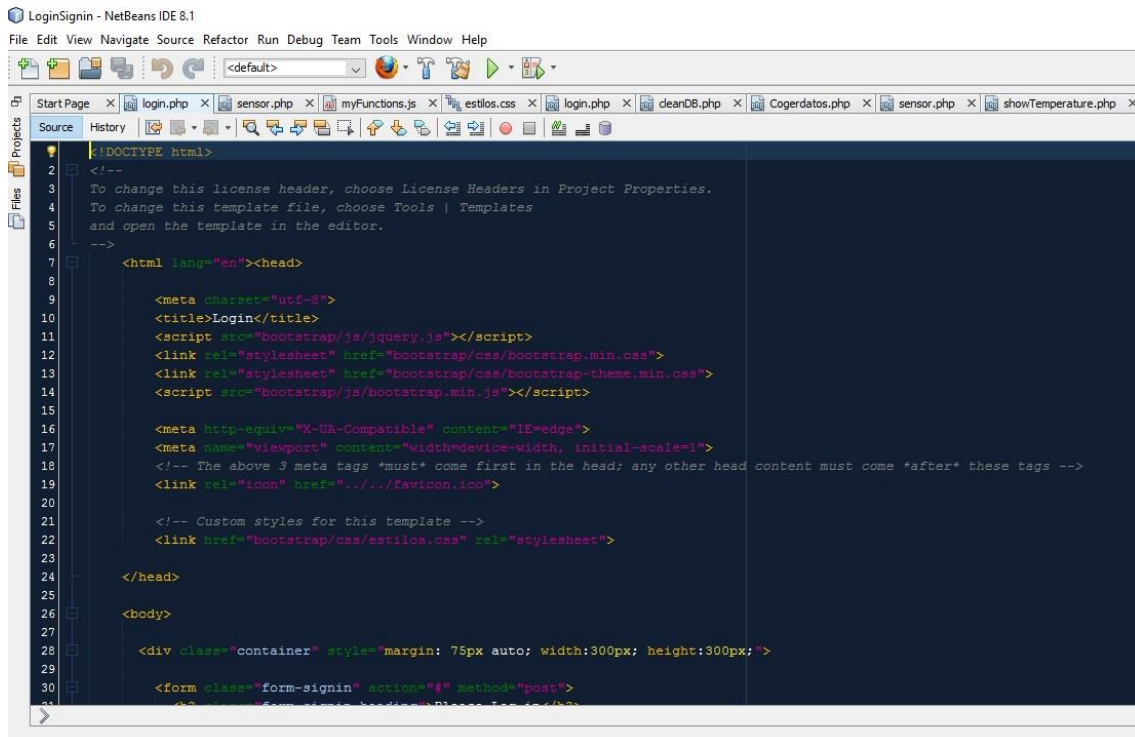


Figure 23: Netbeans interface

- **Bootstrap:** [10] it is a framework written in HTML, CSS and JavaScript, which helps to make front-end pages. It provides templates, examples and also a lot of features. It is an important tool, when you are a beginner doing the web page design, because it simplifies the code and also changes the layout of the website based on the device screen size used for accessing the webpage.

5.2.- Getting started with the Raspberry

Here we are going to explain the software installation process to configure a Raspberry from the start.

First of all we have to format the SD card to be sure it is completely empty. To achieve this, we are going to use the SDFormatter program previously mentioned.

After this step we have to download the OS from the raspberry webpage, we used NOOBS, because it will help us with the installation.[11]

When we have the NOOBS.rar file, we need to unzip it onto the SD card and introduce it into the Raspberry SD slot.

As next we have to connect the Raspberry to the electricity and launch it for the first time. The program “pi recovery” will be the first one to run. The figure 24 shows what will appear on our screen. We select Raspbian and the installation process will take around 15 minutes.



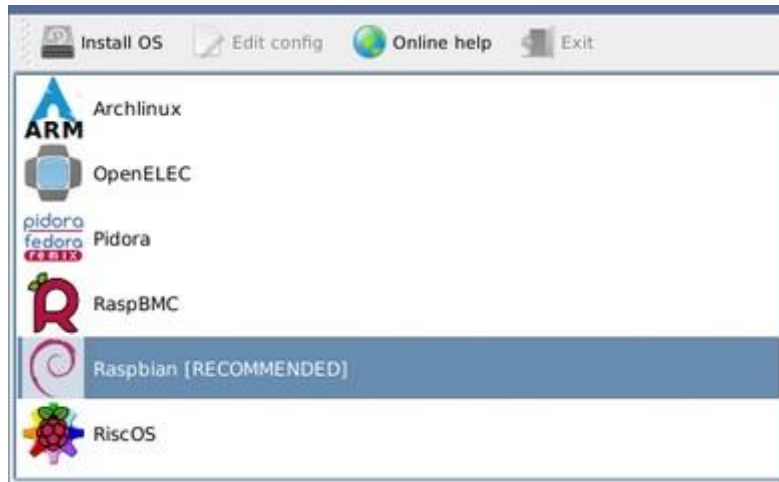


Figure 24: Installation of the OS

Later the figure 25 will appear on our screen. It is the configuration menu from the Raspberry. Here we can configure most of the options of the Raspberry and we can access it anytime we want through the terminal by typing: `sudo raspi-config`

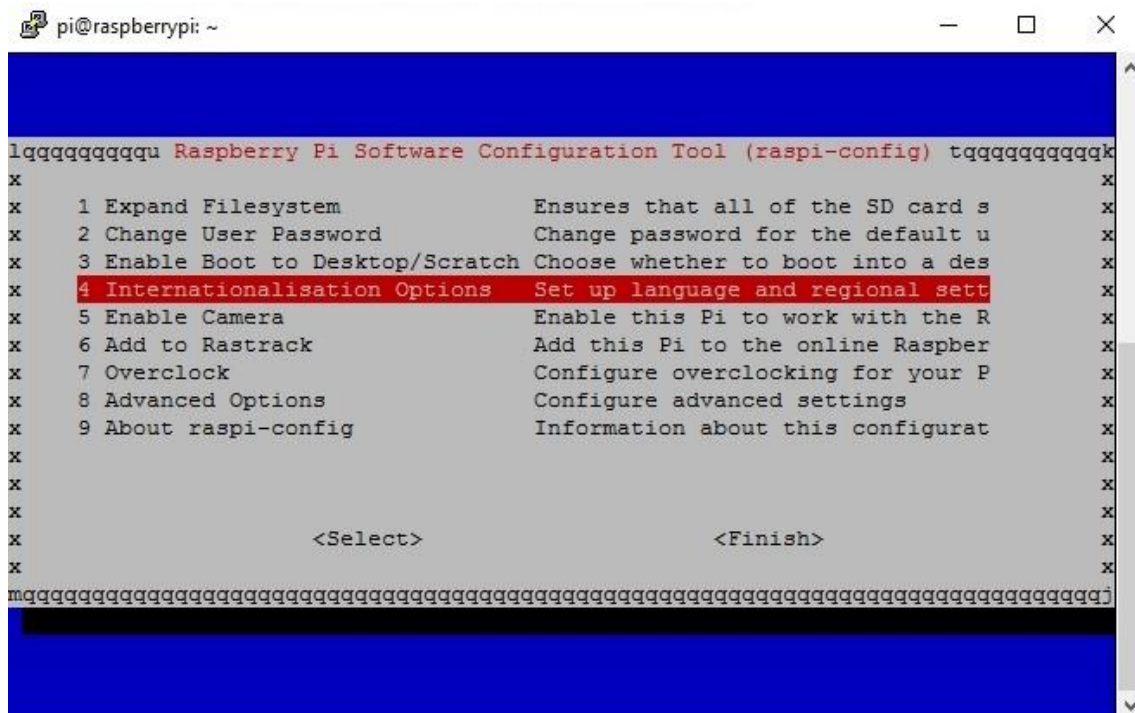


Figure 25: Configuring the Raspberry

The option number one expands the OS in the SD card to make the Raspberry function faster.

The second option allow us to change the password, we should do this because the default password is “raspberrypi” for every raspberry.

In the fourth option we can choose the timezone for the Raspberry. In our case we use the time zone Austria is in, which is GMT +1. We also choose the English language for the keyboard.

There are more options to configure, but we are not using them in this project.



Now we proceed to configure the network interface to have internet access. We modify the file “interfaces” with the text editor nano from the Raspberry. To access the file we have to type in the terminal: *sudo nano /etc/network/interfaces*

```

pi@raspberrypi: ~
GNU nano 2.2.6 File: /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.128.91.4
netmask 255.255.0.0
gateway 10.128.0.1

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-ssid "AndroidHotspot0218"
wpa-psk "0123456789"

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Figure 26: Internet settings

For running a webserver on our Raspberry, a local IP is needed to host it. If we have a static IP, it becomes easier to access it, because we always have the same local IP.

To achieve this we modify the “interface” file like it is shown on the picture 26. In the red square we have the Ethernet connection. We set the local IP to “10.128.91.4” and the gateway to “10.128.0.1”

In the blue square we have the WiFi connection. In this case we are assigned a dynamic IP, because the connection is made to a cell-phone and not to a router, but if we want to assign a static IP we need to copy the code from the Ethernet connection with an appropriate IP.

Also we must be careful assigning static IP, because if any other device is using that IP we can crash the whole LAN network.

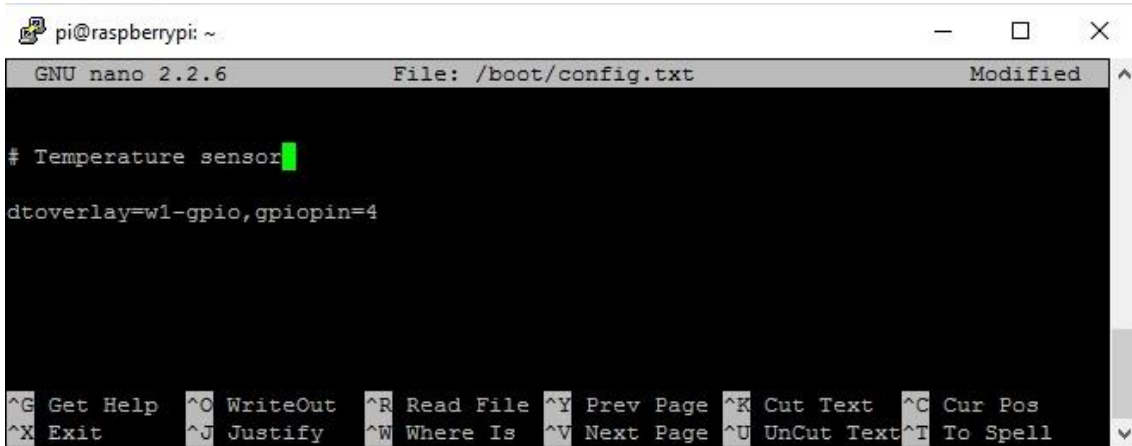
After this we need to reboot our Raspberry to save the changes with the command: *sudo reboot*

5.3.- Collecting the data from the sensor

For collecting the data from the sensor [12] we need to configure the Raspberry kernel to always read the pin of the sensor, which is connected to the Raspberry. To do this we need to open and modify the file: *sudo nano /boot/config.txt*

We write *dtoverlay=w1-gpio, gpiopin=4* and with that line we set the pin to 4 for the sensor.





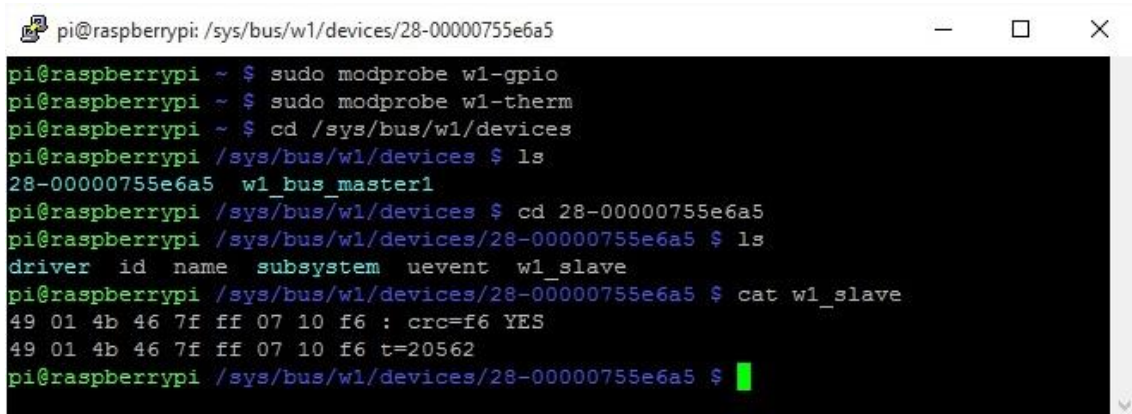
```

pi@raspberrypi: ~
GNU nano 2.2.6 File: /boot/config.txt Modified
# Temperature sensor
dtoverlay=w1-gpio,gpiopin=4
^G Get Help ^C WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Figure 27: Configuring the sensor

Then we have to write the commands for the Raspberry kernel: `sudo modprobe w1-gpio` and `sudo modprobe w1-therm`.



```

pi@raspberrypi: /sys/bus/w1/devices/28-00000755e6a5
pi@raspberrypi ~ $ sudo modprobe w1-gpio
pi@raspberrypi ~ $ sudo modprobe w1-therm
pi@raspberrypi ~ $ cd /sys/bus/w1/devices
pi@raspberrypi /sys/bus/w1/devices $ ls
28-00000755e6a5 w1_bus_master1
pi@raspberrypi /sys/bus/w1/devices $ cd 28-00000755e6a5
pi@raspberrypi /sys/bus/w1/devices/28-00000755e6a5 $ ls
driver id name subsystem uevent w1_slave
pi@raspberrypi /sys/bus/w1/devices/28-00000755e6a5 $ cat w1_slave
49 01 4b 46 7f ff 07 10 f6 : crc=f6 YES
49 01 4b 46 7f ff 07 10 f6 t=20562
pi@raspberrypi /sys/bus/w1/devices/28-00000755e6a5 $

```

Figure 28: Updating the kernel

After this we have a file with the path `/sys/bus/w1/devices/28-00000755e6a5` with the temperature data from the sensor.

The `28-00000755e6a5` is the ID from the sensor. If we connect more than one sensor to the same pin, we will have more files in the folder with different IDs for each sensor.

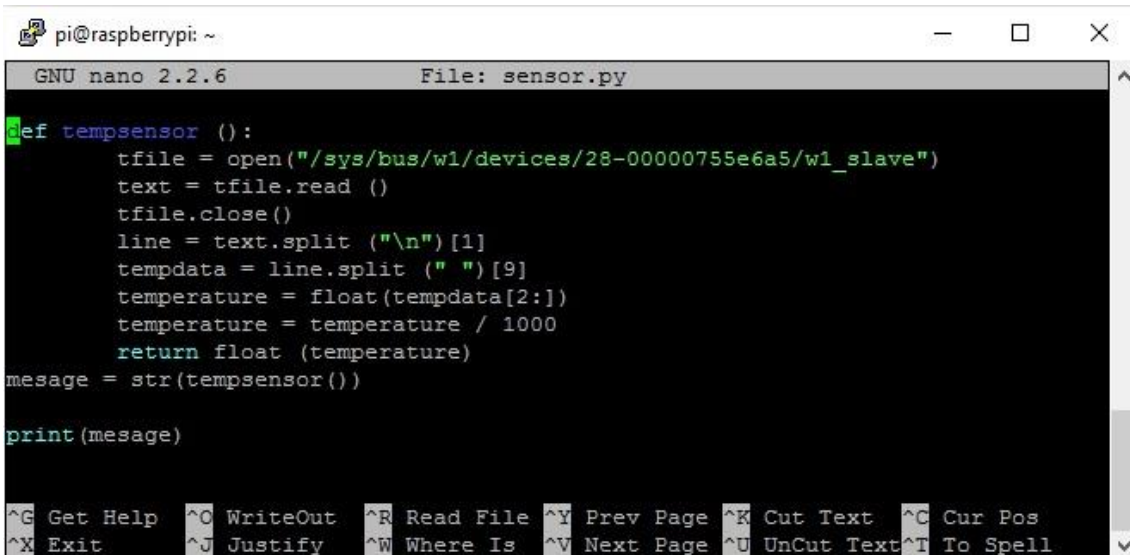
We do not need all the text in that file, we only need the temperature data for using it later, so we have to create a python program to extract it from the file.

As we can see in the figure 29, there is a file called the `sensor.py`. The program written in python is used to get the temperature data from the file. First of all we open the file with the data, we read it and close it.

We split the text into lines and we choose the second line (for splitting the text the zero is the number one). After this we take the ninth position on the second line counting the spaces.

And then we make a float variable with the number, but it is in mili-Celsius, so we have to divide it by 1000 to get it in Celsius and finally we return the temperature.





```

def tempensor ():
    tfile = open("/sys/bus/w1/devices/28-00000755e6a5/w1_slave")
    text = tfile.read ()
    tfile.close()
    line = text.split ("\n")[1]
    tempdata = line.split (" ")[9]
    temperature = float(tempdata[2:])
    temperature = temperature / 1000
    return float (temperature)
message = str(tempensor())

print (message)

```

Figure 29: Getting the temperature from the sensor

5.4.- Installing LAMP

LAMP [13] is the acronym of Linux, Apache, MySQL and PHP. With this four components we can create a webserver. Linux is already installed so we continue with the next one: Apache.

Before we start with the installation, we need to upgrade the Raspberry software to be sure that we have the last updates. To do this, we write in the terminal: `sudo apt-get update` and `sudo apt-get upgrade`

When the Raspberry is already updated and upgraded, we proceed to instal the Apache with the following comand: `sudo apt-get install apache2`

To ensure that it is installed, we open the browser and we write the IP of the Raspberry.



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Figure 30: Successful installation of LAMP

If everything is working, the figure 30 will appear. At this point we can host a webpage on the Raspberry, but we do not have a database or the web service to communicate with it.

As next we install the PHP. To get it we have to write in the terminal : `sudo apt-get install php5` and after this we install all the complements with the comand: `sudo apt-get sudo apt-get install libapache2-mod-php5 libapache2-mod-perl2 php5 php5-cli php5-common php5-curl php5-dev php5-gd php5-imagick php5-ldap php5-mhash php5-mysql php5-odbc`



To verify that everything is correct we create a basic php file with the path `/var/www/rasp.php`



Figure 31: PHP running

Right now we can use PHP, but we do not have the software to use it with a database. For installing it we need to run the command: `sudo apt-get install mysql-server mysql-client php5-mysql`. During the installation the terminal will ask us for the password for the user root.

After the installation we need to start the MySQL. To do this we write in the terminal: `sudo service mysql start`

```

pi@raspberrypi: /
pi@raspberrypi / $ sudo service mysql start
[ ok ] Starting MySQL database server: mysqld already running.
pi@raspberrypi / $ mysql -uroot -praspberry
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 44
Server version: 5.5.46-0+deb7u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
  
```

Figure 32: MySQL running

For using MySQL we have to write `mysql -uroot -ppassword` and we are working with the database. As we can see working with the database in the terminal is difficult.

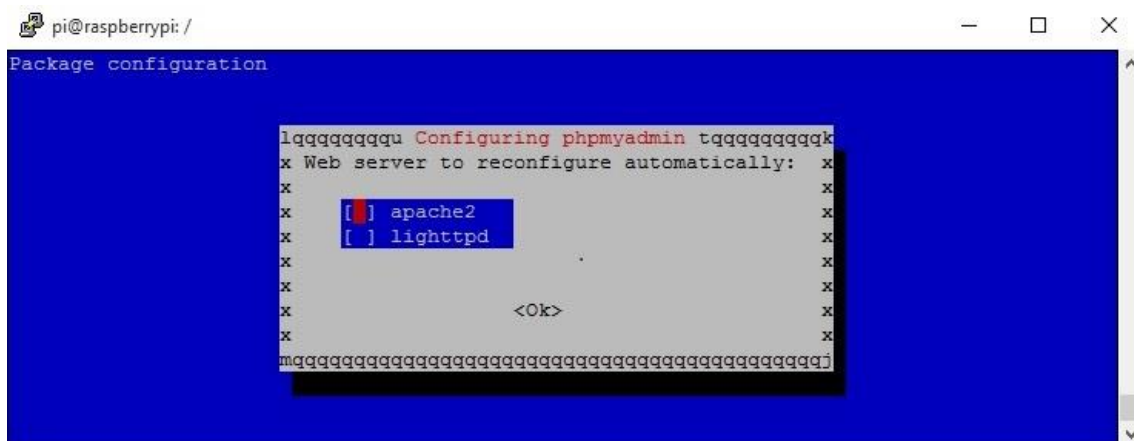


Figure 33: Installing phpMyAdmin



For an easy management of the database we are going to install phpMyAdmin, a web interface for working with the database. For installing it, we have to write in the terminal: `sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin`

After the installation the Raspberry will ask us for the installed server (figure 33) and in our case we select apache2.

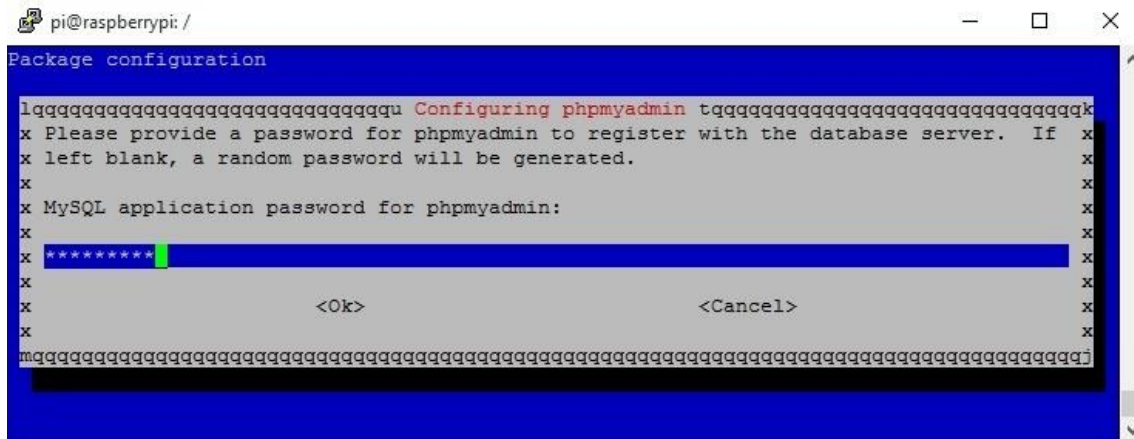


Figure 34: Setting a password for phpMyAdmin

On the next step it will ask us, if we want to configure our database and here we press yes. Then we choose the password for the MySQL and we confirm it.

When we finish the installation, we need to modify one file in the Raspberry, because the PHP has support for MySQL, but it is not enabled by default. We have to add `extension=mysql.so` to the file on the path `/etc/php5/apache2/php.ini`

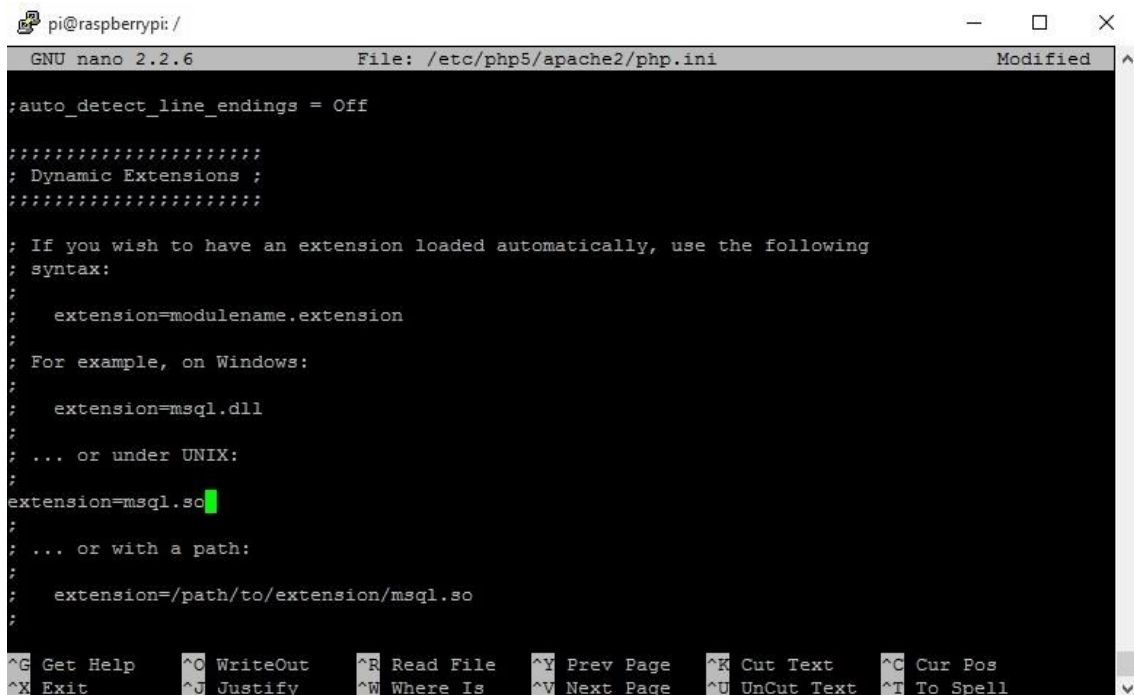


Figure 35: Enabling support for MySQL



5.5.- Creating the database

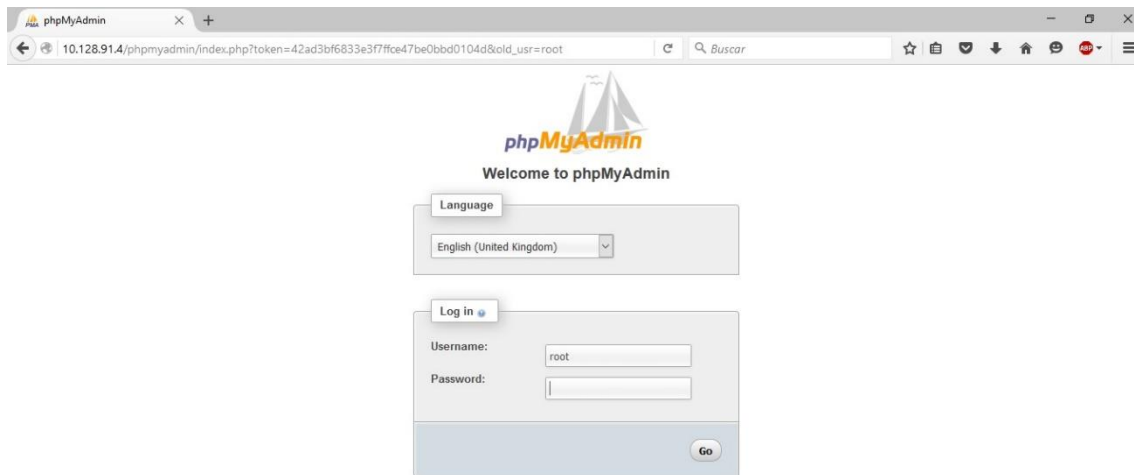


Figure 36: phpMyAdmin main page

In the figure 36 we have the login page of the phpMyAdmin interface. All the data from the sensor is going to be stored in the database. We will also store users and corresponding passwords for accessing the interface.

To do this, we need to create a database. We choose the databases tab and we write the name of the database in "create new database". We create one database called "user1"

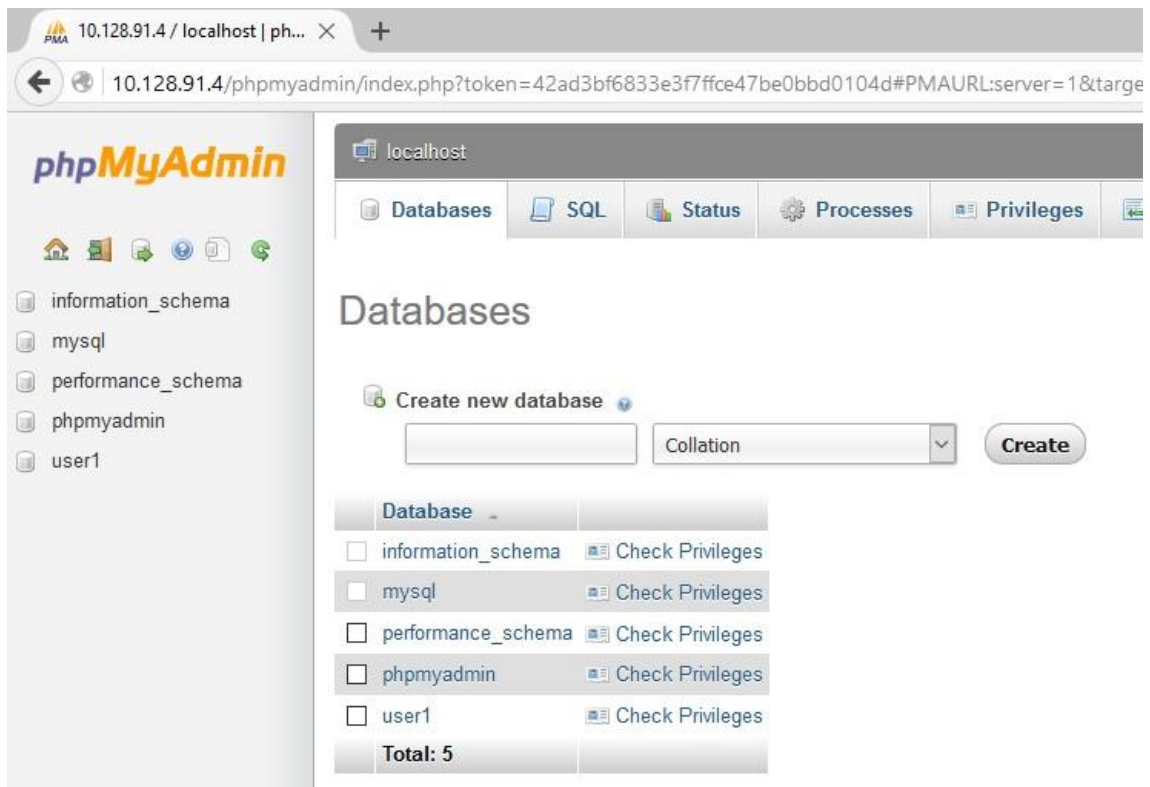


Figure 37: Creating a database



When we have the database, we need to create a table to insert the data into it. We have to click on the database “user1” and then press create table.

The figure 38 shows what appears on the screen. We can name the table and also define the number of columns. In our case the name is login and it will have 2 columns. This table will host the user and the password for the login system.

Also we need another table for the data received from the sensor. It is going to be called data with two columns: value and time.

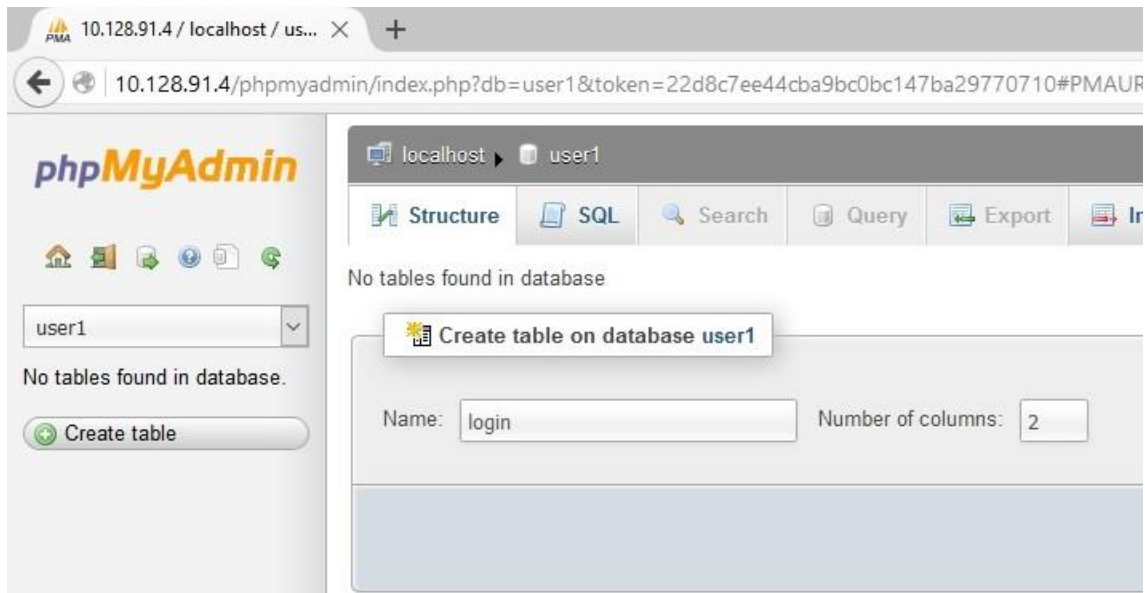


Figure 38: Creating a table

The two columns are of type “VarChar”, because both contain numbers and letters. We will set them to a length of 20 characters and they will be marked as primary, because they are unique.

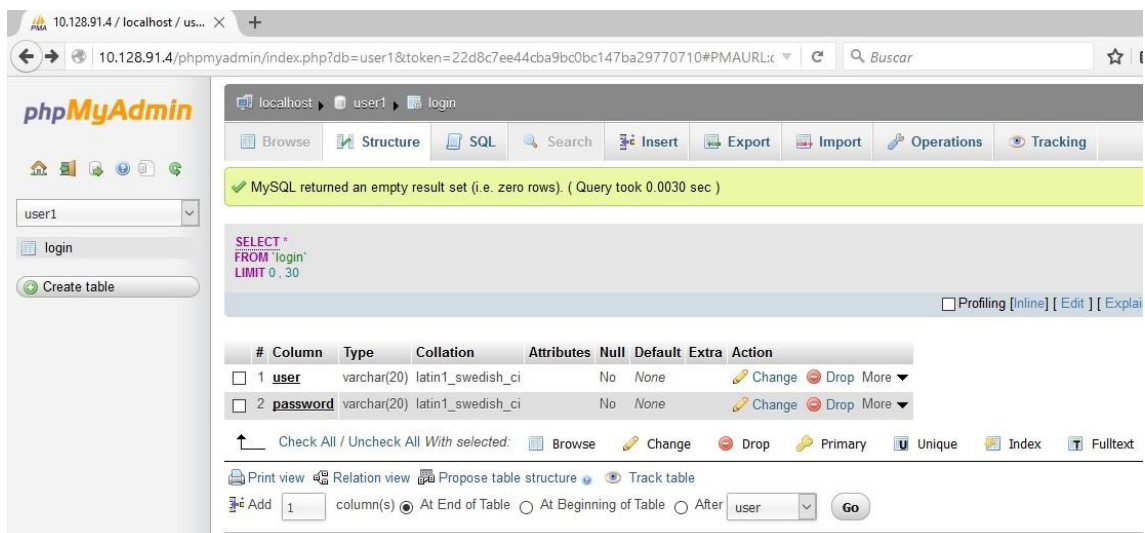


Figure 39: Creating columns

If we want to add data into the tables, we can do it using the SQL tab with the command INSERT or use PHP code or do it with the terminal in the Raspberry.



5.6.- Security

The security is an important field in this kind of devices, because they are connected to the internet. If we do not protect it properly, someone can access the device and control it.

To avoid this we have implemented some security mechanisms:

- A login system to the Raspberry: the Raspberry has all of the project on it, if someone accesses it and searches for the files, he could discover the password for our database in the code and then he will have a complete access to the device.
- A codified and anti SQL injection login system in the webpage: [14] an experienced SQL programmer can send some special characters in the user and password input to trick the database. This type of attack is called SQL injection. To protect against this we have to prevent the use of special characters.
Beside this when we send the password to the database, we do it by hashing it using md5 algorithm. If someone intercepts the message, it is hashed and you cannot use it to access to the interface, because you cannot write it into the login input.
- A header in the interface webpage: If you try to bypass the login screen and go directly to the interface without logging in, it denies you the access to it.

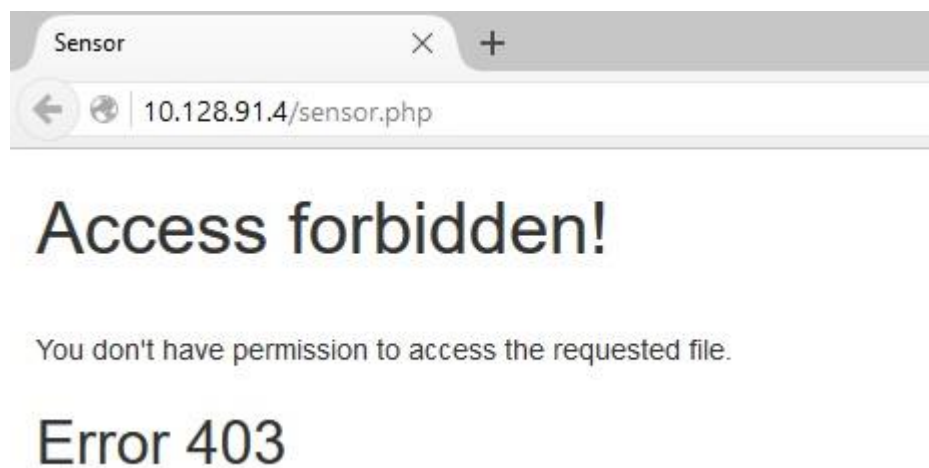


Figure 40: Interface security

We also need some passive security for our device, in our case if something bad happens in one of the programs or the user uses the device improperly.

For having this type of security we use the Crons [15]. The Crons are applications running in the background stored in the Crontabs (figure 41). We can run them once or do it periodically. To enter inside the Crontabs, we need to write in the terminal: `sudo crontab -e`. If we want to check the Crons running we have to type: `sudo crontab -l`

Our security file is squared in red in the figure 41. The security.py runs every 10 minutes and it close the relay when the temperature is higher than 35 degrees.



```

pi@raspberrypi: ~
GNU nano 2.2.6 File: /tmp/crontab.b3DRzy/crontab
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * sh storedata.sh
0 0 * * * sh cleanDB.sh
10 * * * * sudo python security.py

```

Figure 41: Crontabs

5.7 Summarize of the code

We have posted all of the code in the appendix, but we are going to write a brief explanation of every file to get a better idea about the program.

- Login.php: Here we have the login system to the device, the hashing of the password and the anti SQL injection protection.
- Sensor.php: This one is the interface of the device. A user will be able to access this two files only.
- Storedata.php: Opens the file with the data from the sensor and uploads it to the database.
- Storedata.sh: The script in bash to execute the Storedata.php as a Cron. It upload the data every minute to the database as we can see in the figure 41.
- CleanDB.php: If we upload one value every minute, we have 10.080 records per week. That means 40.320 per month. To reduce this, the code executes at midnight and takes the average temperature per hour, stores it and deletes the records of that day, saving only the averages per hour.
With the usage of this we have 672 records every month with high enough precision for the future use of the data.
- CleanDB.sh: Executes the file CleanDB.php every midnight.
- ShowTemperature: This part of the code is the responsible for showing the actual temperature in the interface. If we want to have real-time data on the temperature we need this. We cannot take this data from the database, because it changes every minute.
- TurnOn.sh: Turns on the relay.
- TurnOn.php: Every time we press the button on in the interface executes the TurnOn.sh
- TurnOff.sh: Turns off the relay.
- TurnOff.php: Every time we press the button off in the interface executes the TurnOff.sh
- Status.php: Reads one file in the Raspberry to know, if the relay is turned on or off
- Security.py: As we mentioned before, this is a dead man's switch in case something goes wrong.



6.- Economic study

The aim of this chapter is to evaluate the economic part of developing this project. We are going to list the material resources needed and the indirect costs of the project.

The table 2 has the economic investment in the hardware part of the project. We must consider that the device is a prototype and if we produce it in large quantities the price will be lower.

Item	Quantity	Price
Raspberry pi Model B+	1	20,67€
Tinker kit relay	2	13,97€
DS18B20 temperature sensor	1	1,38€
WiFi USB adapter	1	7,86€
ABS box	1	12,46€
Schuko plug	1	3,22€
Socket	1	2,74€
Screw M3	8	0.05€
Screw M2	6	0.05€
Screw M7	4	0.1€
Cable grommet	2	0.94€
10 Meter power cable	1	1€
10 Jumper wire 150mm male	1	2,66€
2,5 Meter Ethernet cable	1	2,78€
SD Card	1	3,99€
Total price of the hardware		87,60 €

Table 2: Economic study of the project

The indirect cost are not included in the costs of the physical device, but we need them to create the device. They include the electricity consumed, maintenance of the workplace and tools used for creating the device, such as drillers, screwdrivers and a computer.

The indirect cost amount to 125 €. This is an approximation, because we do not know the exact price of every tool and the power consumed.



7.- Future lines

In this chapter we are going to talk about the future improvements of this project. As we have done in the project already, we are going to divide this into the hardware and the software improvements.

For the hardware the main goal is to reduce the cost of the device. Creating a PCB with all the components inside will reduce the cost of the components and also reduce the size of the box.

Including a GSM module for the internet connection in case we do not have internet or WiFi at the place, where we use the device.

Another future possibility is to start a new branch of the project with a wireless temperature sensor with an Arduino connected to it. This could be useful for certain environments, where you cannot connect it with wires, because of physical difficulties, or the distances are long and therefore inefficient to be connected with wires.

For the software part the future plans include the creation of a programmed schedule, which would turn the heater on and off at specific times and enable us to exactly configure at what time what temperature should be in that place.

For example we could configure a workplace to be a bit warmer during the day, when there are employees inside, and let the place be a couple of degrees colder during the night, when there is no one there. Doing that we could save energy and save money.



Bibliography

- [1] A. Kamilaris and A. Pitsillides, "Towards interoperable and sustainable smart homes," *IST-Africa Conf. Exhib. (IST-Africa), 2013*, pp. 1–11, 2013.
- [2] "Netatmo device." [Online]. Available: <https://www.netatmo.com/es-ES/producto/termostato/installation>. [Accessed: 25-Jan-2016].
- [3] "WeMo - insight." [Online]. Available: <http://www.belkin.es/f7c029eaes-interruptor-wemo-insight.html>. [Accessed: 26-Jan-2016].
- [4] "GSM heater controller." [Online]. Available: <http://www.4ucontrol.com/controlador-de-caldera-gsm/>. [Accessed: 26-Jan-2016].
- [5] "Arduino introduction." [Online]. Available: <https://www.arduino.cc/en/Guide/Introduction>. [Accessed: 26-Jan-2016].
- [6] "Raspberry pi documentation." [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/>. [Accessed: 26-Jan-2016].
- [7] "DS18B20 Datasheet." [Online]. Available: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>. [Accessed: 26-Jan-2016].
- [8] "Tinker kit relay." [Online]. Available: <https://store.arduino.cc/product/T010010>. [Accessed: 26-Jan-2016].
- [9] "Verified USB adapters." [Online]. Available: http://elinux.org/RPi_USB_Wi-Fi_Adapters. [Accessed: 26-Jan-2016].
- [10] "Getting started with Bootstrap." [Online]. Available: <http://getbootstrap.com/getting-started/>. [Accessed: 26-Jan-2016].
- [11] "NOOBS download." [Online]. Available: <https://www.raspberrypi.org/downloads/noobs/>. [Accessed: 26-Jan-2016].
- [12] "Cambridge University - Computer laboratory." [Online]. Available: <https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/temperature/>. [Accessed: 26-Jan-2016].
- [13] "Web servers." [Online]. Available: <http://www.rpi.uroboros.es/servers.html>. [Accessed: 26-Jan-2016].
- [14] "SQL injection." [Online]. Available: http://www.w3schools.com/sql/sql_injection.asp. [Accessed: 26-Jan-2016].
- [15] "Scheduling tasks with Cron." [Online]. Available: <https://www.raspberrypi.org/documentation/linux/usage/cron.md>. [Accessed: 26-Jan-2016].



List of Figures

Figure 1: Thermostat by NETAMO	4
Figure 2: WeMo insight by belking.....	5
Figure 3: GSM controller by 4UControl.....	6
Figure 4: Block diagram of the project.....	7
Figure 5: Block diagram of the device.....	8
Figure 6: Arduino Uno	9
Figure 7: Raspberry Pi B+	10
Figure 8: Temperature sensor DS18B20	11
Figure 9: Tinker kit relay.....	12
Figure 10: Optocoupler diagram	12
Figure 11: Wifi USB adapter	13
Figure 12: Box, socket and plug	13
Figure 13: Hole for the socket.....	14
Figure 14: Holes for the cables.....	14
Figure 15: Installation of the socket and plug.....	15
Figure 16: Installation of the relays.....	15
Figure 17: Installation of the Ethernet cable and sensor	16
Figure 18: Installation of the Raspberry.....	16
Figure 19: Raspberry terminal.....	18
Figure 20: Putty interface.....	18
Figure 21: Fillezilla interface	19
Figure 22: SDFormatter Interface.....	19
Figure 23: Netbeans interface.....	20
Figure 24: Installation of the OS.....	21
Figure 25: Configuring the Raspberry	21
Figure 26: Internet settings.....	22
Figure 27: Configuring the sensor	23
Figure 28: Updating the kernel	23
Figure 29: Getting the temperature from the sensor	24
Figure 30: Successful installation of LAMP.....	24
Figure 31: PHP running.....	25
Figure 32: MySQL running.....	25
Figure 33: Installing phpMyAdmin	25
Figure 34: Setting a password for phpMyAdmin	26
Figure 35: Enabling support for MySQL	26
Figure 36: phpMyAdmin main page.....	27
Figure 37: Creating a database.....	27
Figure 38: Creating a table	28
Figure 39: Creating columns.....	28
Figure 40: Interface security	29
Figure 41: Crontabs	30
Figure 42: Login webpage	37
Figure 43: Top view of the interface	37
Figure 44: Daily graphic.....	38
Figure 45: Heater controller and actual temperature	38



List of tables

Table 1: Bill of materials.....	17
Table 2: Economic study of the project	31



List of abbreviations

GSM	Global System for Mobile
SMS	Short Message Service
WiFi	Wireless Fidelity
AC/DC	Altern Current/Direct Current
EMI	ElectroMagnetic Interference
OS	Operating System
PCB	Printed Circuit Board
LED	Light Emitting Diode
USB	Universal Serial Bus
ABS	Acrylonitrile Butadiene Styrene
SSH	Secure SHell
FTP	File Transfer Protocol
IDE	Integrated Development Environment
SD	Secure Digital
GMT	Greenwich Mean Time
IP	Internet Protocol
LAN	Local Area Network
SQL	Structured Query Language
PHP	PHP Hypertext Preprocessor



A.- User Guide

In this appendix we are going to explain the interface and the login system for the user.

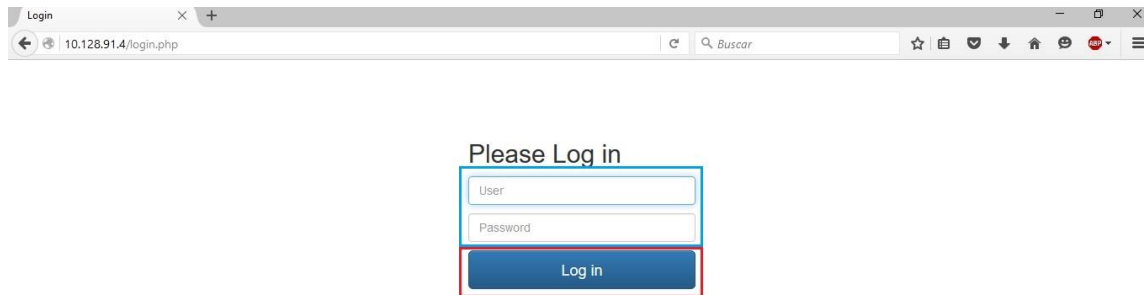


Figure 42: Login webpage

Inside the blue square of the figure 42, we have two input fields, the upper one for the user and the lower one for the password.

When we write the username and the password, we continue with the press the button on the red squared button named “Log in”.

In the figure 43 we have the top part of the interface page. In the green square we have the actual time of the Raspberry. Under it we have a radio button for different graphical displays of the graph: Daily, Weekly and Monthly.

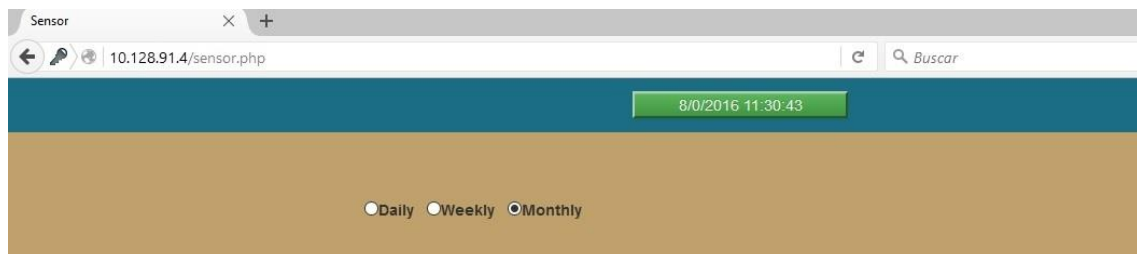


Figure 43: Top view of the interface

In the figure 44 we have the daily plot. The Blue colour in it is when the temperature is under 20 degrees, soft brown for temperatures between 20 and 30 and red for temperatures higher than 30 degrees.

In the figure 45 we have the actual temperature indicator with the same colours as before.

Also there is the on and off buttons for the heater. The brighter button indicates, which one is active at that moment. In the figure 45 the heater is off.



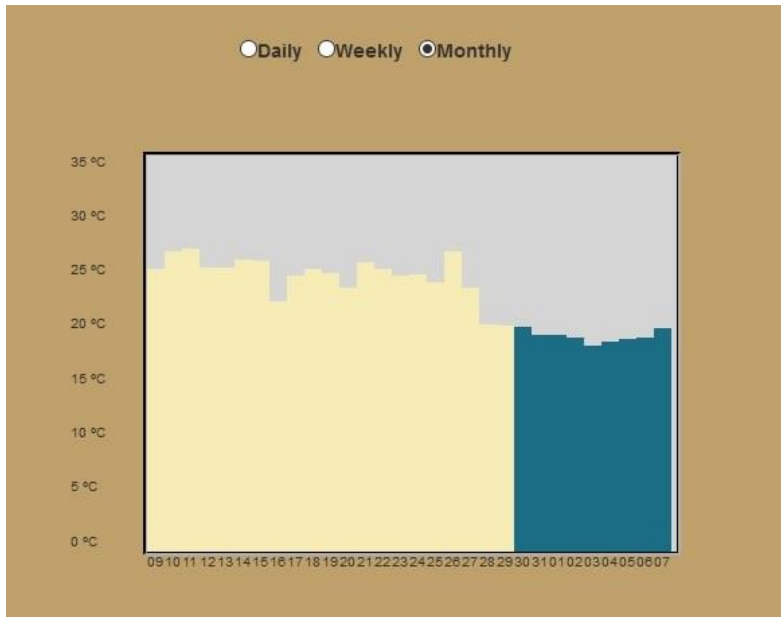


Figure 44: Daily graphic

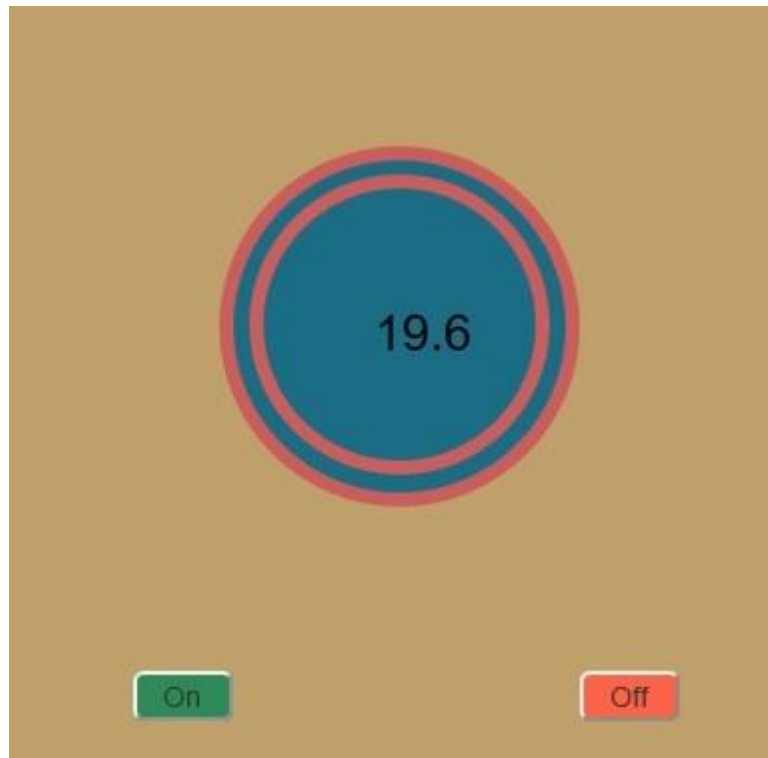


Figure 45: Heater controller and actual temperature



B.- Useful commands for the Raspberry

<code>sudo <command></code>	Execute the command as root.
<code>raspi-config</code>	Configuration menu of the Raspberry.
<code>apt-get update</code>	Download the new actualizations.
<code>apt-get upgrade</code>	Install the new actualizations.
<code>apt-get install <name></code>	Installs the program for the terminal.
<code>mkdir <name></code>	Create the <folder>.
<code>rm -r <name></code>	Delete the <folder>.
<code>cd <path></code>	Get inside the <folder>.
<code>cd ..</code>	Go back to the previous folder.
<code>ls</code>	List of the files and folders of the actual folder.
<code>nano <name></code>	Opens the text editor with the <name>.
<code>reboot</code>	Reboots the system.
<code>shutdown now</code>	Turns down the system.



C.- Code

Login.php

```

<!DOCTYPE html>

<html lang="en"><head>

  <meta charset="utf-8">
    <title>Login</title>
    <script src="bootstrap/js/jquery.js"></script>
    <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="bootstrap/css/bootstrap-theme.min.css">
    <script src="bootstrap/js/bootstrap.min.js"></script>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href=" ../favicon.ico">
    <!-- Custom styles for this template -->
    <link href="bootstrap/css/estilos.css" rel="stylesheet">

</head>

<body>
  <!-- Heading of the document -->
  <div class="container" style="margin: 75px auto; width:300px; height:300px;">

    <form class="form-signin" action="#" method="post">
      <h2 class="form-signin-heading">Please Log in</h2>
      <input id="inputEmail" class="form-control marged_inputs" placeholder="User" required=""
autofocus="" type="text" name="name_login">
      <input id="inputPassword" class="form-control marged_inputs" placeholder="Password"
required="" type="password" name="pass_login">
      <button class="btn btn-lg btn-primary btn-block" type="submit" name="send_login">Log
in</button>
    </form>
    <!-- Connection to the database -->
    <?php
    session_start();
    $login = filter_input(INPUT_POST,"send_login");
    $conector_bbdd = new mysqli('localhost', 'root', 'raspberry','user1');
    if(isset($login)){
      //Anti SQL-Injection
      $name = filter_input(INPUT_POST,"name_login");
      $pass = filter_input(INPUT_POST,"pass_login");
      $name_seguro = htmlspecialchars($name);
      $pass_segura = htmlspecialchars($pass);
      // Hasing
      $pass_segura_cifrada = hash('md5',$pass_segura);
      $consulta = "SELECT user,password FROM login WHERE user ='".$name_seguro."' AND
password ='".$pass_segura_cifrada."'";

```



```

    $resultado = $conector_bbdd->query($consulta);
    if($resultado->num_rows == 1){
        $rows = array();
        while($row = $resultado->fetch_array(MYSQLI_NUM))
        {
            $rows[] = $row;
        }
        // Starting a session to avoid the bypass
        $_SESSION["username"] = $rows[0];
        header("Location: sensor.php");
    }else{
        echo '<div class="alert alert-danger" role="alert">The user and the password does not
match</div>';
    }
}
$conector_bbdd->close();
?>

</div> <!-- /container -->

<!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
<script src="../../assets/js/ie10-viewport-bug-workaround.js"></script>

</body>
</html>

```

Sensor.php

```

<!DOCTYPE html>

<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Sensor</title>
    <script src="bootstrap/jquery/jquery.js"></script>
    <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="bootstrap/css/bootstrap-theme.min.css">
    <script src="bootstrap/js/bootstrap.min.js"></script>
    <script src="bootstrap/js/myFunctions.js"></script>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href="../../favicon.ico">
    <!-- Custom styles for this template -->
    <link href="bootstrap/css/estilos.css" rel="stylesheet">
</head>

<body>
    <?php
        // Security header to avoid the bypass

```



```

session_start();
if(!isset($_SESSION["username"])){
    echo "<div style='margin-left:15px;'><h1>Access forbidden!</h1>"
        . "<br />"
        . "<p>You don't have permission to access the requested file.</p>"
        . "<h2>Error 403</h2></div>";
}
else{
    $conector_bbdd = new mysqli('localhost', 'root', 'raspberry','user1');
    $altura_maxima = 35;
    ?>
    <div class="row" style="background-color: #1b6d85; height: 50px;"> <!-- Top bar with the time and
date -->
        <div class="col-sm-12">
            <button id="btn_time" class="btn-success btn-md center-block" style="height: 25px; width:
200px; margin-top: 12px;">Conecting...</button>
        </div>
    </div>
    <div class="row fill" style="background-color: #c0a16b; padding: 60px; padding-left: 0px;"> <!--
radio button for selecting plot -->
        <div class="row">
            <div class="col-sm-12">
                <div class="center-block col-sm-6" style="float: none;">
                    <input type="radio" name="graficas" value="D" checked id="grafica_diaria"/><label
for="grafica_diaria">Daily</label>&nbsp;&nbsp;&nbsp;
                    <input type="radio" name="graficas" value="S" id="grafica_semanal"/><label
for="grafica_semanal">Weekly</label>&nbsp;&nbsp;&nbsp;
                    <input type="radio" name="graficas" value="M" id="grafica_mensual"/><label
for="grafica_mensual">Monthly</label>
                </div>
            </div>
        </div>
        <div id="div_grafica_diaria" class="col-sm-8" style="padding: 60px; padding-left: 0px;"> <!-- Daily
graphic -->
            <div class="row">
                <div class="col-sm-1 col-sm-offset-3" id="numero_temperaturas">
                    <div style="margin-bottom: 26px; font-size: 10px;" >35 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >30 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >25 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >20 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >15 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >10 °C</div>
                    <div style="margin-bottom: 26px; font-size: 10px;" >5 °C</div>
                    <div style="font-size: 10px;">0 °C</div>
                </div>
                <div class="col-sm-8" style="width: 400px; height: 300px; background-color: #d5d5d5; padding-
left: 0px;padding-right: 0px; border: groove black 3px;">
                    <?php
                        $tiempoActual1 = time();
                        $tiempoFormateado1 = date('Y-m-d H:i:s', $tiempoActual1);
                        $arrayTiempo1 = explode(" ",$tiempoFormateado1);
                        $x = $arrayTiempo1[1];

```



```

$arrayhorax = explode(":", $x);
$horax = $arrayhorax[0];
$hora_menos_uno = $horax."00:00";
$horaformateada = $arrayTiempo1[0]." ".$hora_menos_uno;
$mediaNoche1 = "00:00:00";
$arrayTiempo1[1] = $mediaNoche1;
$tiempoFormateado2 = implode(" ", $arrayTiempo1);
// Selecting the daily temperatures from the database
$selectTemperatures1 = "SELECT valor,time FROM data WHERE time >
".$tiempoFormateado2." and time < ".$horaformateada."";
$result = $conector_bbdd->query($selectTemperatures1);
$contador_barras = 0;
$minuto = 0;
$primer_tiempo;
$array_temperaturas_medias = array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
$array_horas = array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
$num_filas_total = $result->num_rows;
$rows = array();
while($row = $result->fetch_array(MYSQLI_NUM))
{
    $rows[] = $row;
}
// Getting the average temperatures for every hour
for($numero_fila = 0; $numero_fila < $num_filas_total; $numero_fila++){
    $row = $rows[$numero_fila];
    if($minuto==60){
        $array_temperaturas_medias[$contador_barras] =
$array_temperaturas_medias[$contador_barras] / $minuto;
        $array_horas[$contador_barras] = $primer_tiempo;
        $minuto = 0;
        $contador_barras++;
    }
    if($minuto==0){
        $primer_tiempo = $row[1];
    }
    $array_temperaturas_medias[$contador_barras] =
$array_temperaturas_medias[$contador_barras] + $row[0];
    if($numero_fila==($num_filas_total-1) && $minuto!= 0){
        $array_temperaturas_medias[$contador_barras] =
$array_temperaturas_medias[$contador_barras] / $minuto;
        $array_horas[$contador_barras] = $primer_tiempo;
    }
    $minuto++;
}
// drawing the graphic
foreach($array_temperaturas_medias as $temperatura_en_hora){
    $altura_porcentaje = ($temperatura_en_hora * 294) / $altura_maxima;
    $resto = 294 - $altura_porcentaje;
    $color_fondo = "#d5d5d5";
    if($temperatura_en_hora > 30){
        $color_fondo = "indianred";
    }
}

```



```

        }else if($temperatura_en_hora > 20){
            $color_fondo = "#f7ecb5";
        }else{
            $color_fondo = "#1b6d85";
        }
        echo "<div style='background-
color: ".$color_fondo.";width:16.3px;height:".$altura_porcentaje."px;float: left; margin-
top:".$resto."px;'></div>";
    }
    echo "</div><div class='row'><div class='col-sm-8 col-sm-offset-4'>";
    foreach($array_horas as $time){
        if($time != 0){
            $fecha_hora = explode(" ", $time);
            $tiempo = $fecha_hora[1];
            $array_tiempo = explode(":",$tiempo);
            $hora = $array_tiempo[0];
            echo "<div style='width:16.3px;font-size:8px;float:left;'>".$hora."</div>";
        }
    }
    echo "</div></div>";
?>
</div>
</div>
<div id="div_grafica_semanal" class="col-sm-8" style="padding: 60px; padding-left: 0px;"> <!--
weekly graphic -->
    <div class="col-sm-1 col-sm-offset-3" id="numero_temperaturas">
        <div style="margin-bottom: 26px; font-size: 10px;">35 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">30 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">25 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">20 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">15 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">10 °C</div>
        <div style="margin-bottom: 26px; font-size: 10px;">5 °C</div>
        <div style="font-size: 10px;">0 °C</div>
    </div>
    <div class="col-sm-8" style="width: 400px; height: 300px; background-color: #d5d5d5; padding-
left: 0px;padding-right: 0px;border: groove black 3px;">
        <?php
            $tiempoActual2 = time();
            $tiempoFormateado3 = date('Y-m-d H:i:s', $tiempoActual2);
            $arrayTiempo2 = explode(" ",$tiempoFormateado3);
            $mediaNoche2 = "00:00:00";
            $arrayTiempo2[1] = $mediaNoche2;
            $tiempoFormateado4 = implode(" ", $arrayTiempo2);
            // Selecting the weekly temperatures from the database
            $selectTemperatures2 = "SELECT valor,time FROM data WHERE time <
"". $tiempoFormateado4. "" and time > "" . $tiempoFormateado4. "" - interval '10080' minute";
            $result2 = $conector_bbdd->query($selectTemperatures2);
            $contador_barras2 = 0;
            $hora = 0;
            $primer_tiempo;

```



```

$array_temperaturas_medias2 = array(0,0,0,0,0,0,0);
$array_dias2 = array(0,0,0,0,0,0,0);
$num_filas_total2 = $result2->num_rows;
$rows2 = array();
while($row = $result2->fetch_array(MYSQLI_NUM))
{
    $rows2[] = $row;
}
// Getting the average temperatures for every day
for($numero_fila = 0; $numero_fila < $num_filas_total2; $numero_fila++){
    $row = $rows2[$numero_fila];
    if($hora==24){
        $array_temperaturas_medias2[$contador_barras2] =
$array_temperaturas_medias2[$contador_barras2] / $hora;
        $array_dias2[$contador_barras2] = $primer_tiempo;
        $hora = 0;
        $contador_barras2++;
    }
    if($hora==0){
        $primer_tiempo = $row[1];
    }
    $array_temperaturas_medias2[$contador_barras2] =
$array_temperaturas_medias2[$contador_barras2] + $row[0];
    if($numero_fila==($num_filas_total2-1) && $hora!= 0){
        $array_temperaturas_medias2[$contador_barras2] =
$array_temperaturas_medias2[$contador_barras2] / $hora;
        $array_dias2[$contador_barras2] = $primer_tiempo;
    }
    $hora++;
}
// drawing the graphic
foreach($array_temperaturas_medias2 as $temperatura_en_dia){
    $altura_porcentaje = ($temperatura_en_dia * 294) / $altura_maxima;
    $resto = 294 - $altura_porcentaje;
    $color_fondo = "#d5d5d5";
    if($temperatura_en_dia > 30){
        $color_fondo = "indianred";
    }else if($temperatura_en_dia > 20){
        $color_fondo = "#f7ecb5";
    }else{
        $color_fondo = "#1b6d85";
    }
    echo "<div style='background-
color: ".$color_fondo.";width:56.1px;height: ".$altura_porcentaje."px;float: left; margin-
top: ".$resto."px;'></div>";
}
foreach($array_dias2 as $time){
    if($time != 0){
        $fecha_hora = explode(" ", $time);
        $tiempo = $fecha_hora[0];
        $array_tiempo = explode("-", $tiempo);
    }
}

```



```
        $dia = $array_tiempo[2];
        echo "<div style='width:56.1px;float: left;font-size:10px;text-align:
center;'>".$dia."</div>";
    }
}
?>
</div>
</div>
<div id="div_grafica_mensual" class="col-sm-8" style="padding: 60px; padding-left: 0px;"> <!--
monthly graphic -->
<div class="col-sm-1 col-sm-offset-3" id="numero_temperaturas">
<div style="margin-bottom: 26px; font-size: 10px;">35 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">30 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">25 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">20 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">15 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">10 °C</div>
<div style="margin-bottom: 26px; font-size: 10px;">5 °C</div>
<div style="font-size: 10px;">0 °C</div>
</div>
<div class="col-sm-8" style="width: 400px; height: 300px; background-color: #d5d5d5; padding-
left: 0px;padding-right: 0px;border: groove black 3px;">
<?php
    $tiempoActual3 = time();
    $tiempoFormateado5 = date('Y-m-d H:i:s', $tiempoActual3);
    $arrayTiempo3 = explode(" ", $tiempoFormateado5);
    $mediaNoche3 = "00:00:00";
    $arrayTiempo3[1] = $mediaNoche3;
    $tiempoFormateado6 = implode(" ", $arrayTiempo3);
    // Selecting the weekly temperatures from the database
    $selectTemperatures3 = "SELECT valor,time FROM data WHERE time <
'".$tiempoFormateado6."' and time > '".$tiempoFormateado6."' - interval '43200' minute";
    $result3 = $conector_bbdd->query($selectTemperatures3);
    $contador_barras3 = 0;
    $hora2 = 0;
    $primer_tiempo;
    $array_temperaturas_medias3 =
array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    $array_dias3 = array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    $num_filas_total3 = $result3->num_rows;
    $rows3 = array();
    while($row = $result3->fetch_array(MYSQLI_NUM))
    {
        $rows3[] = $row;
    }
    // Getting the average temperatures for every day
    for($numero_fila = 0; $numero_fila < $num_filas_total3; $numero_fila++){
        $row = $rows3[$numero_fila];
        if($hora2==24){
            $array_temperaturas_medias3[$contador_barras3] =
$array_temperaturas_medias3[$contador_barras3] / $hora2;
```



```

        $array_dias3[$contador_barras3] = $primer_tiempo;
        $hora2 = 0;
        $contador_barras3++;
    }
    if($hora2==0){
        $primer_tiempo = $row[1];
    }
    $array_temperaturas_medias3[$contador_barras3] =
$array_temperaturas_medias3[$contador_barras3] + $row[0];
    if($numero_fila==($num_filas_total3-1) && $hora2!= 0){
        $array_temperaturas_medias3[$contador_barras3] =
$array_temperaturas_medias3[$contador_barras3] / $hora2;
        $array_dias3[$contador_barras3] = $primer_tiempo;
    }
    $hora2++;
}
// drawing the graphic
foreach($array_temperaturas_medias3 as $temperatura_en_dia){
    $altura_porcentaje = ($temperatura_en_dia * 294) / $altura_maxima;
    $resto = 294 - $altura_porcentaje;
    $color_fondo = "#d5d5d5";
    if($temperatura_en_dia > 30){
        $color_fondo = "indianred";
    }else if($temperatura_en_dia > 20){
        $color_fondo = "#f7ecb5";
    }else{
        $color_fondo = "#1b6d85";
    }
    echo "<div style='background-
color: ".$color_fondo.";width:13px;height: ".$altura_porcentaje."px;float: left; margin-
top: ".$resto."px;'></div>";
}
foreach($array_dias3 as $time){
    if($time != 0){
        $fecha_hora = explode(" ", $time);
        $tiempo = $fecha_hora[0];
        $array_tiempo = explode("-", $tiempo);
        $dia = $array_tiempo[2];
        echo "<div style='width:13px;float: left;font-size:10px;'>".$dia."</div>";
    }
}
?>
</div>
</div>
<div class="col-sm-4"> <!-- layer with the actual temperature and on/off button -->
<div class="row" style="padding: 40px; padding-left: 0px;">
<div class="col-sm-12" style="padding: 40px; padding-left: 0px;"> <!-- actual temperature -->
<div class="center-block" style="width: 180px; height: 180px; border-radius: 180px;
background-color: #1b6d85; font-size: 25px;padding: 7px; border: solid indianred 7px;">
<div id="temperature" class="center-block" style="width: 150px; height: 150px; border-
radius: 150px; color: black; padding: 55px; font-size: 25px; border: solid indianred 7px;">

```




```

        </div>
    </div>
</div>
<div class="row" style="">
    <div class="col-sm-4" style=""> <!-- on and off button -->
        <button id="btn_on" class="btn-md center-block" style="height: 25px; width: 50px;
margin-top: 2px; border-radius: 5px;">On</button>
    </div>
    <div class="col-sm-4 col-sm-offset-2" style=""> <!-- Columna que contiene el botón de off -->
        <button id="btn_off" class="btn-md center-block" style="height: 25px; width: 50px;
margin-top: 2px; border-radius: 5px;">Off</button>
    </div>
</div>
<div class="row" style="background-color: #1b6d85; height: 50px;"> <!-- bottom of the page -->
    <div class="col-sm-12" style="color: white;">
        <p style="text-align: center; font-size: 15px;">Site created by Jonatan Rubio Bueno -
Fachhochschule Technikum Wien</p>
    </div>
</div>
<?php
}
?>
</body>
</html>

```

Storedata.php

```

<?php
// opening the temperature file from the Raspberry
function tempSensor(){
    $tfile = file("/sys/bus/w1/devices/28-00000755e6a5/w1_slave");
    $line = $tfile[1];
    $tempdata1 = explode(" ",$line)[9];
    $tempdata2 = explode("=", $tempdata1)[1];
    $tempdata3 = floatval($tempdata2);
    $temperature = round($tempdata3 / 1000,1) ;
    return $temperature;
}
// connecting to the database to insert the data
$temperature = tempSensor();
$conector_bbdd = new mysqli('localhost', 'root', 'raspberry','user1');
$insert = "INSERT INTO data (valor) VALUE(".$temperature.")";
$resultado = $conector_bbdd->query($insert);
$conector_bbdd->close();
?>

```



CleanDB.php

```

<?php
// connet to the database
$conector_bbdd = new mysqli('localhost', 'root', 'raspberry', 'user1');
// take the data from today
$select = "SELECT valor,time FROM data WHERE time > current_timestamp - interval '1440' minute";
$result = $conector_bbdd->query($select);
$minuto = 0;
$hora = 0;
$array_temperaturas_medias = array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
$array_horas = array(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
$primer_tiempo;
$num_filas_total = $result->num_rows;
$rows = array();
while($row = $result->fetch_array(MYSQLI_NUM)){
    $rows[] = $row;
}
// Getting the average temperatures for every hour
for($numero_fila = 0; $numero_fila < $num_filas_total; $numero_fila++){
    $row = $rows[$numero_fila];
    if($minuto==60){
        $minuto = 0;
        $array_temperaturas_medias[$hora] = $array_temperaturas_medias[$hora] / 60;
        $array_horas[$hora] = $primer_tiempo;
        $hora++;
    }
    if($minuto==0){
        $primer_tiempo = $row[1];
    }
    $array_temperaturas_medias[$hora] = $array_temperaturas_medias[$hora] + $row[0];
    if($numero_fila==( $num_filas_total-1) && $minuto!= 0){
        $array_temperaturas_medias[$hora] = $array_temperaturas_medias[$hora] / $minuto;
        $array_horas[$hora] = $primer_tiempo;
    }
    $minuto++;
}
// delete the previous records selected before
$delete = "DELETE FROM data WHERE time > current_timestamp - interval '1440' minute";
$resultado = $conector_bbdd->query($delete);
// insert the average temperatures back to the database
for($hora = 0; $hora < sizeof($array_temperaturas_medias); $hora++){
    $insert = "INSERT INTO data (valor,time) VALUE
('".$array_temperaturas_medias[$hora]."', '".$array_horas[$hora]."'");
    echo $insert."<br /><br />";
    $resulta = $conector_bbdd->query($insert);
}
$conector_bbdd->close();
?>

```



Showtemperature.php

```
<?php
// Get the temperature from the file
function tempSensor(){
    $file = file("/sys/bus/w1/devices/28-00000755e6a5/w1_slave");
    $line = $file[1];
    $tempdata1 = explode(" ",$line)[9];
    $tempdata2 = explode("=", $tempdata1)[1];
    $tempdata3 = floatval($tempdata2);
    $temperature = round($tempdata3 / 1000,1) ;
    return $temperature;
}

$temperature = tempSensor();
echo $temperature;
?>
```

myFunctions.js

```
$(document).ready(function(){

    $.ajax({ //Checking the previous state of the heater
        url: "status.php",
        success: function(resultado){
            if(resultado.indexOf("on")!=-1){
                $on = true;
                $("#btn_on").addClass("enabled_green");
                $("#btn_off").addClass("disabled_red");
            }else{
                $on = false;
                $("#btn_on").addClass("disabled_green");
                $("#btn_off").addClass("enabled_red");
            }
        }
    });

    // the top date and hour
    $timer = function(){
        $date = new Date();
        document.getElementById('btn_time').innerHTML =
        $date.getDate()+"/"+$date.getMonth()+"/"+$date.getFullYear()+
        "+$date.getHours()+":"+$date.getMinutes()+":"+$date.getSeconds();
    };

    // Turn on and off the heater
    $isOn = function(){
        if($on){
            $("#btn_on").removeClass("disabled_green");
            $("#btn_off").removeClass("enabled_red");
            $("#btn_on").addClass("enabled_green");
            $("#btn_off").addClass("disabled_red");
        }
    };
});
```



```
$.ajax({
  url: "turnOn.php",
  success: function(resultado){

  });
}else{
  $("#btn_on").removeClass("enabled_green");
  $("#btn_off").removeClass("disabled_red");
  $("#btn_on").addClass("disabled_green");
  $("#btn_off").addClass("enabled_red");
  $.ajax({
    url: "turnOff.php",
    success: function(resultado){

    });
  }
};

$("#btn_on").click(function(){
  $on = true;
  $isOn();
});

$("#btn_off").click(function(){
  $on = false;
  $isOn();
});

setInterval($timer, 1000);

// Different colours for different temperatures

$getDatos = function(){
  $.ajax({
    url: "showTemperature.php",
    success: function(result){
      $("#temperature").html(result);
      $("#temperature").removeClass(function() {
        return $( this ).prev().attr( "class" );
      });
      if(result>30){
        $("#temperature").addClass("calor");
      }else if(result > 20){
        $("#temperature").addClass("normal");
      }else{
        $("#temperature").addClass("frio");
      }
    }
  });
};

// Selecting the graphics
setInterval($getDatos, 1000);
```



```

$("#div_grafica_semanal").addClass("hide");
$("#div_grafica_mensual").addClass("hide");

function ocultarGráficas(){
    $("#div_grafica_diaria").removeClass("show");
    $("#div_grafica_semanal").removeClass("show");
    $("#div_grafica_mensual").removeClass("show");
    $("#div_grafica_diaria").addClass("hide");
    $("#div_grafica_semanal").addClass("hide");
    $("#div_grafica_mensual").addClass("hide");
}

$("#input[name='graficas']").change(function(){
    ocultarGráficas();
    $("#div_"+this.id).removeClass("hide");
    $("#div_"+this.id).addClass("show");
});
});

```

Estilos.css

```

.marged_inputs{
    margin-top: 10px;
    margin-bottom: 10px;
}

@media (min-width: 1367px) {
    html{
        height: 905px;
    }
    .fill {
        height: 805px;
    }
}

@media (max-width: 1367px) {
    html{
        height: 640px;
    }
    .fill {
        height: 540px;
    }
}

.enabled_green{
    background-color: springgreen;
}

```



```
.enabled_red{
  background-color: tomato;
}

.disabled_green{
  background-color: seagreen;
}

.disabled_red{
  background-color: sienna;
}

.frio{
  background-color: #1b6d85;
}

.normal{
  background-color: #f7ecb5;
}

.calor{
  background-color: indianred;
}

.hide{
  display: none;
}

.show{
  display: block;
}
```

