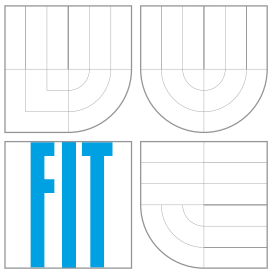
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

AUTOMATIC THERMOMETER

AUTOMATICKÝ TEPLOMĚR

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
JORGE CUADRADO SAEZ

SUPERVISOR
VEDOUCÍ PRÁCE
Doc. Dr. Ing. DUŠAN KOLÁŘ

BRNO 2016

Abstract

The aim of this thesis is trying a Raspberry Pi board as and standalone system. The system will measure the temperature and run a web server. The web server will show the data in a web application and also it will provide the configuration options of the device. Furthermore, the device will transfer periodically the data to a remote host. The main problems of this aproach are the power consumption and the privacity of the data stored in the device. The power efficiency carry many considerations. On the one hand, it is needed to develop a efficient system, avoiding not essential libraries and trying to use as much software already present on the system as possible. On the other hand is necessary to configure the system in order to disable all the unnecesary hardware and software. This document details some power saving configurations made in the hadware and in the software of the board. There are many possible software configurations in order to save battery. The problem is about hardware configurations, the hardware is quite simple and several times it is not possible to disable components and some configurations are not possible without add external hardware components. Finally, it is possible to low the power consumption of the device greatly. But even with this configurations, the power consumption will be higher than in a normal embedded system, which can perform the temperature measurement properly and will be able to continue running more efficiently and more time. But the real fact is that a Raspberry Pi device despite of its power consumption is capable of working in more fields, it is cheaper and also provide a faster develop than a traditional embbeded system which are specific for one specific task.

Abstrakt

Tato práce popisuje tvorbu automatického teploměru spolu s jeho konfigurací a prezentací na vlastním WWW serveru na platformě Raspberry Pi 2. Spolu se základními prvky měření a konfigurace se projekt zabýval i možnostmi automatického odesílání dat a aktivním snižováním spotřeby.

Keywords

IoT (Internet of the Things), embedded systems, standalone system, Raspberry Pi, web application, power saving

Klíčová slova

IoT (internet věcí), vestavěné systémy, samostatně pracující systémy, Raspberry Pi, Web, snižování spotřeby

Reference

CUADRADO SAEZ, Jorge. *AUTOMATIC THERMOMETER*. Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Kolář Dušan.

AUTOMATIC THERMOMETER

Declaration

I declare that I have created this thesis myself under the supervision of Doc. Dr. Ing. Dušan Kolář. I have cited all the bibliographic sources and publications used for the creation of this thesis.

.....
Jorge Cuadrado Saez
May 18, 2016

Acknowledgements

This project would not have been possible without the invaluable help of those people who helped me these last three months.

I want to specially thank Zbyněk Křivka and Dušan Kolář for the help and advice that was given to me. I also want to thank Zdeněk Vašíček for helping me with the hardware module and Marek Rychlý for the inconveniences caused for letting me share the office 229.

Also, I want to thank Alba Camazón Pinilla for her assistance on checking the grammar and ortography in this document. She made it possible to make it legible. Additionally, many thanks to Guillemo Román Ferrero for his aid regarding security and network issues.

Furthermore, I want to thank Samuel for his support and advice about Python. `Siempre nos quedarán las risas`.

And finally, I want to thank my parents, my brother, and the incredible `Hotties`, Vivi, Sergio and all the fantastic Erasmus people who `supported` me all this time.

Contents

Chapter 1

Introduction

1.1 Overview

This project consists on creating an intelligent device able to measure the temperature from a Raspberry Pi board and a temperature sensor. Moreover, the device has to be prepared to create a web server which provides a web platform which shows visitors the last temperature samples taken by the device and also allows logged users to change the configuration parameters of the device remotely.

The objective of this project is to use a Raspberry Pi board and some hardware components like a RTC module and a Temperature sensor create a device which will be able to get temperature samples of the surrounding place.

This temperature station has to be prepared to send the data to another target machine and be configured remotely from a web application which has to be provided for a web server hosted also in the same device.

That Web application will be composed by public area which will show the temperature samples and a graphic to the visitant users and also a private area protected with login which will allow authenticated users to change the device configuration like samples frequency, number of samples shown in the web page or also the target machine for the scp data transfers.

This project is framed in the field of Internet of the things and thought as a investigation about the capabilities of the well known Raspberry Pi board for host a complex embedded system in an efficient way as a data producer for other devices. The processing of that temperature data is not an objective of the designed system, leaving aside the web page provided by the system which will create a simple graphic with it.

The system will be designed as a standalone system which will provide all the mentioned features. The system is also thought for being moved to some different locations, reason why the device will be wireless, using a usb Wi-Fi module in order to get connected to Internet and a battery to get the power.

Furthermore, the power efficiency is another important matter in the project as a consequence of using a battery as a power source, so the system will allow the user to decrease the power consumption changing some device configurations and disabling device functionalities.

1.2 Internet of the Things

The Internet of Things (IoT) [8] is a relatively recent concept which refers to the interconnection of devices in the physical world like thermometers, washing machines or traffic lights. This network of devices permits us to create new machines which will do more complex things than we thought possible years ago.

They IoT network can gather and create a lot of data using the devices sensors in the network which is exchanged between devices. Also, these devices use other devices data to choose the best behaviour in every case permitting the machines to becoming more intelligent which means that it can automatice more tasks, give us more and more useful information and becoming more accurate and efficient making their tasks.

One of the facts that are responsible for the success of the IoT movement is the hardware requirement. The IoT promotes the use of little embedded hardware devices which limit CPU computational power, few memory and also low power consumption. This is a real advantage comparing to other kind of systems which require expensive and powerful machines and permit to integrate IoT systems in almost all the fields. Some of the most useful fields are in industry automating task, in the domestic field for example helping to consume energy in an efficient way or notifying a broken electrodomestic. Also it can be used to get environmental information which can help to get advised of a natural disaster before it happens or to control the water contamination. One more important field is set in urbanism, this technology permits us to get many useful information, for example, about traffic, contamination and pollen in the air. This examples are only a few comparing to the real possibilities of this technology.

Other of the important facts which permitted to start thinking of IoT as a serious alternative was the release of IPv6 addresses allowing this little devices to get access to Internet and permitting to interconnect these devices between them no matter the distance and location. And more important, permitting users to manage and get information of these devices from every place. This last advantage also carries an important matter concerning the security of the information which is generated by these devices and also to protect these devices against malicious people who can get advantage of using them for their own interest.

Between the reason which are making this technology grow, one is the scalability of these networks. It is relatively easy to add new devices to the network which will generate more data or use the existent data to increase the functionality of all the network.

Nowadays, many companies are becoming to release new IoT products which greatly improves the life of the users. There are many examples like intelligent fridges which notify the owner when there isn't any bottle of milk. Or the example of sensors which measure the toxicity of the river wate preventing us to get poisoned from a factory leak. And also systems which using a smartphone application allows a driver to know where is a place to park the car.

One particular example of these technologies can be found in Santander, Spain, where the parkings are intelligent. There are sensors under the floor which send the status information of the parking to a server which makes this information public to the inhabitants. [11]

Also in Zaragoza, Spain, sensors gather anonymous information about the places where people usually go, from where and at what time in order to improve the public transport.

This last example also carries some considerations about the privacy of the people when these systems are introduced in our lives, how this data can be used and which limits we

have to put.

1.3 The Aim of This Project

There are many alternatives to make an standalone system. from specific chips which perform particular and fixed tasks to more general chips like Arduino which allows to perform more complex tasks and manage other sensors. Recently, it appear a new kinf of devices like Raspberry Pi which are computers with full functionality and capabilities with low power consumption and a litle size.

This project has been planified in order to test devices like Raspberry Pi, which are becoming more powerful and efficient day by day. This devices allows to create new and more complex devices with more possibilities than traditional desktop computers, servers or embedded systems.

The problem of this topic is the power management. We need a extremely efficient power management in our system. A simple embedded system will consume much less power than a device like Raspberry Pi, which is composed by many chips. But the possibilities of a system deployed in a Raspberry Pi are incredibly bigger than the possiblities on a system deployed in an embedded chip or an Arduino board. That is bacause of the greater capabilities of Raspberry Pi board, that can be so useful in the IoT field which is growing fastly and demanding more complex functionalities every day.

It is important to be concerned about the security and safety of the data stored in the device. Credentials, certificates and also the samples. In the moment we think about designing a wireless system that can be placed in a public environment, we should start thinking also about protecting the device, not only from hackers attack, but also from possible steals and how to hide the stored data.

Chapter 2

The System: Hardware

2.1 The Raspberry Pi

Raspberry Pi is a well known micro computer which has become famous due to its low price and its high performance in a device of little size [9] [6]. After years of its releases the community behind it continues growing and growing and because of that fact combined to the possibilities that this device characteristic offer has created a great context to develop small smart devices at a low price, fact which becomes Raspberry Pi in a great choice for IoT projects.

2.1.1 History and Social Importance

The creator of this device was Eben Upton realized that new students of computer sciences in Cambridge University had lower computer skills. Upton arguments that this is because in the new game consoles, mobile phones and new devices are "fixed function devices", not like the old devices such as Commodore or Spectrum which users needed to learn how to programm it in order to use it. [17]

To solve this problem he proposed a new little machine that could be bought for a low price, like 25$, which could help child to learn about computation and programming. The success of this idea was completely unexpected for Embed Upton who thought about creating only 1000 devices. In the first day after the release, they sold 100.000 devices.

Nowadays, the Raspberry Foundation is supported by the University of Cambridge Computer Laboratory and Broadcom, which is an educational charity meant to promote the study of computer science topics using Raspberry Pi as a tool.

Right now, many organizations use Raspberry Pi with an education purpose in all education levels and every time more countries are using it in the schools as a tool to teach children not only computational skills but also mathematical concepts and other fields like physics with tools such as the Scratch programming language.

This cheap device has created a huge community of people around it who are interested in using it as an education tool, as a research device and also as an entertainment tool. And actually it is possible to learn how to build interesting and complex projects thanks to it. There are many examples and ideas which also started as a kind of entertainment and ended up as a commercial product.

2.1.2 Hardware

The Raspberry Pi is a fully functional computer with also a low power consumption. Since the first one was released, several new models have appeared. The first one was the model A. Few months later the model B was released. After some time appeared the A+, B+ models as a revision of the models A and B. Not so much time ago was released the model 2 B. And few months ago the models Zero and 3 were marketed. Every model has a different hardware design. There had been some improvements among them. The model used in this project is the Raspberry Pi 2 B+ [3], which is not the choice that fits better in this project as I explain in the 2.1.3 subsection.

In this section i will explain briefly the main hardware characteristics of the model used in the project.

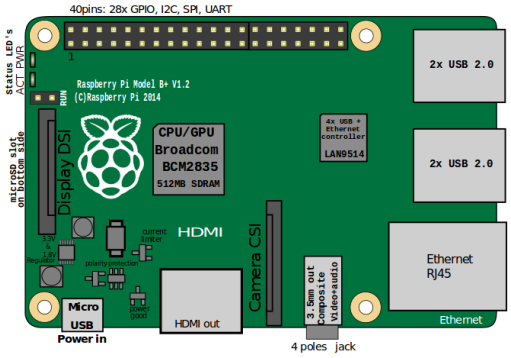


Figure 2.1: Raspberry Pi main components.

The board, by default, builds four USB ports which allow it to handle multiple peripheral connections like keyboard, mouse, Wi-Fi usb module and others. There is also a micro USB port but is designed to power the device with a standard smartphone charger.

In order to allow the Internet connection the device mount an ethernet socket. In our project it was needed to buy also a USB Wi-Fi module to allow the device to be moved everywhere without depend of any unnecessary wire.

Both, the USB ports and the ethernet socket, are controlled by the same hardware hub. This fact was important in the develop of the power saving modes as it is explained in section 2.1.3.

There are also other connections which are not used in this project like the 3.5mm audio jack and composite video port, the Camera interface (CSI) and the Display interface (DSI). Also there is a HDMI port which was used for the development and it will probably use for the maintenance in the future but it is not used while the device is running normally.

The principal data storage of the device is in a microSD card, the board also counts with a micro SD card slot.

The CPU mounted in the Raspberry Pi 2 B is a ARMv7 quad core at 900 MHz and the principal memory is a 1 GB SDRAM at 400 MHz.

Finally, the board has 40 GPIO pins, these pins are one of the most important features in the machine, which permit to add a new hardware, sensors and controllers to the board allowing to develop complex systems as robots, media centers or weather stations.

2.1.3 Power Save Works

As a micro computer the Raspberry Pi board has a low consume of power. Because of that, this device is a great option for build systems which has to work full time without stop. But in our system, the power is one of the strongest limitations since the device will use an external battery as power supply.

For our project, the device has some hardware components which we do not need, like the HDMI socket or all the USB ports less one for the Wi-Fi usb module. After the research, it is possible to disable these components saving some battery and some of the power modes disable it as we have explained in the 4.5 section.

Although there is a limitation, USB ports are not independent between them and with the ethernet socket [10]. This is because all of these ports are controlled by the same hub. So when we disable the power we can not disable only one of them and we will lose any possibility of having Internet connection because our two ways to get this are the ethernet socket and a USB Wi-Fi module which uses one of these four USB ports.

This dessign was made thinking in the mobile systems and it is simpler than a normal desktop computer USB hub and it does not allow to only let one of the usb ports enabled. It is possible to disable the data transmission of the ports but the device will continue giving them energy so it is not useful to disable them. As i will explain in the seccion 6.2, the disable of the data transfer would be useful as a security functionality.

Additionally, so many Internet blogs write about downclock the CPU frequency [18], in the practice downclock the Raspberry Pi CPU saves a despicable amount of power but the heat of the CPU decreases and this fact can avoid corrupt the temperature samples. [12]

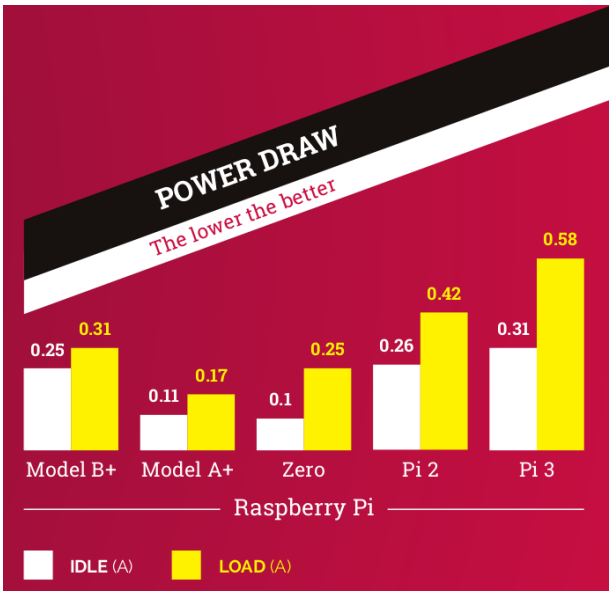


Figure 2.2: Power consumption diagram from Raspberry Pi Magazine [4].

Other saving method is to disable the status led which decrease a bit more the power consumption but at the end is also a despicable amount.

As a final consideration, there is a Raspberry Pi model, the model A, which is more suitable for this project. That model has a lower regular power consumption than the B model and only had one USB between other facts as show in the 2.2 figure.

2.2 The Module

2.2.1 Installation

The provided module comes prepared to directly plug in the Raspberry Pi GPIO pins. The module has to be plugged in the 1, 3, 5, 7 and 9 pins according to the module installation manual.

This will allow us to access and install the two components pf the module. The first one is a RTC module model PCF85163 which can be configured to keep the right time between reboots and also to release alarm signals in the right moment. The second chip in the module is a temperature sensor model DS18B20 which is a one-wire digital sensor that measure the temperature.

2.2.2 PCF85163

The PCF85163 is a CMOS Real-Time Clock (RTC) and a calendar optimized for low power consumption [2]. Although, this chip provides other functionalities, the two functions which the system use of pcf85163 are the clock output which permit to get the actual date in the pcf85163 and the interrupt output which permits to wake up the device from a sleep state.

Why a RTC?

Our system needs to still work after several days but the Raspberry Pi device despite being a device which consumes so few battery comparing to a normal computer, consumes so much battery comparing to a hardware specialized for embedded systems. For this reason we need to low the consume of the device in order to save battery to maintain working out the device more time.

Carrying this to an extreme, one of our partial objectives is to make the device consume just the necessary battery to make its job.

In order to achieve this partial objective, the way to consume the less battery as possible is putting the device to sleep in the intervals when the user does not need to interact with the device and it is supposed not to run a task.

Normal computers have a special hardware which permits them to wake up automatically when they are sleeping. Unfortunately, the Raspberry Pi does not count with the necessary hardware for wake up automatically. That is the reason why we need a RTC, which allows the Raspberry Pi device to wake up using his interrupt output which sends a signal to device when a configured alarm occurs.

PCF85163 Chip

How is represented in the figure 2.3. The chip count with 8 pins.

The first and the second pin are the input and output of the chip oscillator. The third pin is the alarm output which only will be activated when the alarm happeds. The fifth and sixth are the serial data I/O and the serial clock input. The seventh pin is the clock output. The fourth and the eight pin are from the ground and the power.

Installation

The pcf85163 comes also built on a prepared module for Raspberry Pi. For this reason once the module is plugged on the Raspberry Pi is necessary to add `dtoverlay=i2c-rtc,pcf8563`

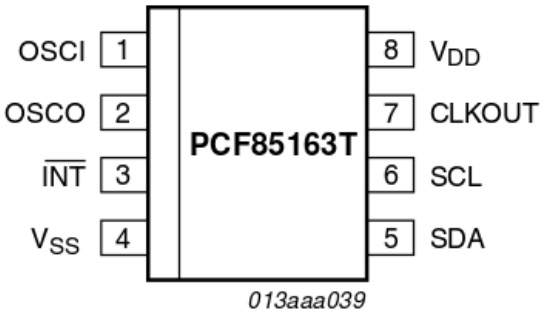


Figure 2.3: PCF8563 pins diagram.

in the `/etc/config.txt` file [19].

This line in the device configuration will mount the modules to manage the RTC modules in the system and the specific drivers of the pcf8563 family.

Then, after reboot the device, it is necessary to configure the actual time stored in the chip. For this task we can synchronize with the system time if it is updated or set up the time manually. Some Linux distributions as Raspbian offer an interface for this use with the command `hwclock`. To set the system time in the chip we can use `hwclock -w` command. To set it up manually we can use `hwclock -time -set=„date"`.

After setting up the correct time, we have to configure the device to get the time of the RTC module on boot time. To get this we need to edit `/lib/udev/hwclock-set` and change all the `"-systz"` apparitions for `"-hctosys"`.

Now it is possible to access to the RTC time and apparently the time is kept between reboots but it is not the reality. In this kind of device like raspberry pi, which does not have any hardware clock to keep the actual time. There is a process called `fake-hwclock` which calculates the actual time.

For force the system to use the RTC module time we have to remove the `fake-hwclock` process from the system using the following commands.

```
1        sudo apt-get remove fake-hwclock
2        sudo rm /etc/cron.hourly/fake-hwclock
3        sudo update-rc.d -f fake-hwclock remove
4        sudo rm /etc/init.d/fake-hwclock
5        sudo update-rc.d hwclock.sh enable
```

Now the system will get the time from the RTC however the time of the system in every reboot is always “Jan 1st 1970”. This happen because before setting up the hour from the RTC in the system, the RTC time is overwritten in `/lib/udev/hwclock-set`. to fix this it is necessary to remove or comment the following lines in the script.

```
1        if [ -e /run/systemd/system ] ; then
2        exit 0
3        fi
```

Now the RTC keeps the right date and the system gets the time from the RTC properly.

How to Use It

There are multiple ways to use it, in our project get the date from the RTC is made automatically from the system once it is configured so we do not need to manage with it. Although the time can be gotten using the command `hwclock -r` or getting the value of the file `/sys/class/rtc/rtc0/time`

We also need to configure the alarms in the RTC in order to wake up the Raspberry pi in the right moments. This last task is not trivial and needs low level programming so that the development os this was considered as out of the scope of the project. To achive this the systems uses a library called "AlarmPi" [5] which provides the necessary interface to manage alarms in the RTC module. This library uses Python3 instead of Python 2.7, this is not a real problem because Raspbian and most of the Debian flavours bring by default both.

"AlarmPi" has a useful script called `setAlarm.py` which permits to set the alarm in the chip for a specified day in the month, recovers the scheduled alarm time from the chip and also erases the alarm from the device. The command for setting up the alarm is `python3 setAlarm.py -d day -h hour -m minute` allowing us to set an alarm in a specific day of the actual month and for a specified hour and minute. The hour must be set up in a 24 hours format. Finally to cancel the actual alarm is necessary to use the "-c" flag ("–cancel") without any value and to recover the alarm from the chip the "-s" flag ("–status").

Problems During the Development

The RTC installation consumed so much time, the instructions summed up here are the product of a hard investigation. The main problem was the lack of information about this module not only in Internet blogs because it is not a well known module, but also in the store web page there is not referent to the producer and the documentation provided does not contribute enough information for a developer.

Also the first module provided has a manufacturing defect which allows to get the temperature value of the sensor but does not permit to get the values from the pcf85163 RTC returning by the standard output of the `hwclock` command this following non explicit error.

```
hwclock: The Hardware Clock registers contain values that are either invalid
(e.g. 50th day of month) or beyond the range we can handle (e.g. Year 2095).
```

After searching the causes and also solutions to this error, we try to solve it in some other distributions without any modification and we try alternative installation methods. It was impossible to determine why the RTC fails. Then we tried a copy of the module which worked properly so we assumed that was a hardware error.

Also this module was not designed to wake up the raspberry pi and the board of the module does not connect the pcf85163 alarm output pin to the module output so i managed to connect it wiring the alarm output pin to the "run" input of the Raspberry Pi [16] board with the hope of emulate this function. As for today i could not manage to achieve this so the third power saving mode, described in the section 4.5.7, will remain as a demo feature, the code of the power mode will be provided as normal but in the script the called mode will be the second one.

Actually, it is possible to set up and recover the alarms from the chip using the "AlarmPi" library but it is not certain whether the RTC is capable of waking up the device using the alarm signal when it happens. The only source of information concerning this topic was the company which created the "AlarmPi" library for their own products.

There are also other modules more complex than the provided one which are specifically created to manage and optimize the power consumption in Raspberry Pi and also to enable hibernation and suspend modes in the device like Sleepy Pi shield board [13].

2.2.3 DS18B20

The main sensor in the system is the DS18B20 temperature sensor [1]. This is a 1-wire sensor and permits to connect various DS18B20 sensors in parallel using the same pins, one for the data and another one for the power. This sensor measures the temperature directly in a digital format permitting to get the data directly from the sensor. This sensor can work in a range of temperature between -10ºC and 60ºC according to the module specifications.

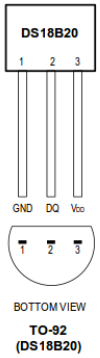


Figure 2.4: Sensor diagram.

Moreover, this sensor uses a parasite power mode, which permits it to drain energy from the data bus.

Installation

First, we need to connect the sensor to the Raspberry Pi. The DS18B20 has three pins as shown in the figure 2.4. The first and third ones are for the 3v3 pin and for the ground pin, the second one, connected to the 4 GPIO transfers the data. The following figure show the connection diagram necessary for this sensor.

The ds18b20 needs to be wired also with a 4.7K resistor as is written in the documentation of the sensor. In our case the sensor was built and ready to use on a RTC module so it was not needed to modify the hardware provided wiring a resistor.

Once the sensor is physically installed, it is necessary to change some configuration files in the Raspberry Pi allowing the device to detect the sensor and read from it.

In order to make this possible we need to add `dtoverlay=w1-gpio` in the `/etc/config.txt` configuration file.

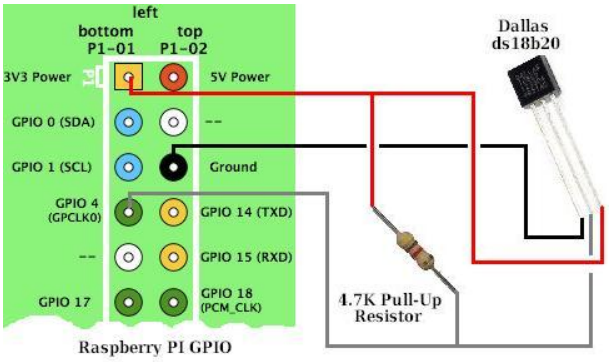


Figure 2.5: Connection diagram for ds18b20 sensor.

How to Use It

Once the sensor is connected and detected by the device, there should be a file in the `/sys/bus/w1/devices/` folder called `28-*` being the star a string identificator of the sensor which permits to distinguish between this sensor and another ds18b20 temperature sensors. Inside the sensor folder, if we make a `cat` of the `w1_slave` we will get dump of the sensor data as it is shown in the figure 2.6.

```
pi@dhcps181:/sys/bus/w1/devices/28-000005f9b4b0 $ cat w1_slave
d1 01 4b 46 7f ff 0f 10 6c : crc=6c YES
d1 01 4b 46 7f ff 0f 10 6c t=29062
```

Figure 2.6: Dump of the ds18b20 sensor data.

As we can see in the figure 2.6 our temperature sensor is the 28-000005f9b4b0 being the identificator of our sensor the `000005f9b4b0` one. Also in that figure we can see there is written a "YES" which means that the data of the sample is not corrupted. Also we can see "t=29062" which represents the value of the temperature. To get the right value we have to divide that value between 1000 getting a value of 29.062 °C.

Chapter 3

Software Decisions

3.1 The Operative System

There are so many options available in the market. Everyone has his own pros and cons but at the end I decided to bet for compatibility and i chos the official Debian flavour designed specially for the Raspberry Pi.

3.1.1 Why Raspbian Lite?

There are so many reasons to choose Raspbian. But the most important is because it is a distribution made by Raspberry.org, specially designed for the Rasperry Pi. It means that all the community behind Raspberry Pi is more likely to use this distribution. That is so useful in order to find solutions, tutorials, documentation and libraries specially made to be used in this distribution with a Raspberry Pi device. Also it is so useful because the Raspbian distributions count with the called Lite version. This version is offered without graphical interface support, so thanks to this, it does not have to store several unnecessary and expensive drivers in memory and also the image in memory is lighter than the normal one.

3.1.2 Consequences During the Develop

Working without graphical interface is less comfortable than working with it but after getting used to work only with the shell, the develop was faster. Choosing Raspbian Lite was a really great decision because all the official documentation concerning Raspberry Pi is thought to use this software distribution and even more important it contains all the necessary drivers to work with the Raspberry Pi hardware.

Wi-Fi Setup

VUT University provides two Wi-Fi networks, one is VUTBRNO network and the other one is Eduroam. I prefered use Eduroam because it has Wi-Fi spots in several universities and organisations attached to the project Iris around all Europe and VUTBRNO authentication uses a web formulary which is difficult to fill from the command line or a script.

The Wi-Fi setup from command line is not as easy as one made from a graphical user interface. After searching the commands or configurations needed to configure the Wi-Fi connection in a distribution without graphical interface as Raspbian Lite i found the

wpa_supplicant [7] command line utility which permits to configure a connection and is used in every reboot for the system so is was perfect to connect the device to Eduroam.

The setup of the wpa_supplicant configuration is not so difficult. It is needed a configuration file with the network credentials and security configurations and a couple of commands. The configuration file for the wpa_supplicant is available in the anexus D.

Once the configuration file `wpa_supplicant.conf` is created and placed in `/etc/wpa_supplicant/` it is only necessary to execute the following commands:

```
1  sudo wpa\_supplicant -Dwext -iwlan0 -c /etc/wpa\_supplicant.
       conf
2  sudo dhcpcd wlan0
```

Components Disable

One of the important points in the project is the possibility of moving the device everywhere only with a battery suppliying power. This means that we have an important need of saving all the power possible and developing an efficient system. One of the most important actions was the disable of all the hardware components that are not needed in the device. Using this distribution, the disabling of all the components that the hardware design permits to be trivial only using some scripts provided by the distribution.

3.1.3 Other Alternatives

Raspberry Pi can run several operating systems so that it offers huge possibilities. The most well known possibilities which are offered are Raspbian, Ubuntu Mate, Windows 10 IoT core and Openelec offered in the official Raspberry Pi foundation web page. Also there are some general alternatives not specifically offered for Raspberry Pi like Debian, Linux Mint, Slackware and Archlinux.

After reading about the options I prefered to take out Windows 10 IoT because i am not familiarized with the windows environment, Openelect since it is specialized in multimedia and the general distributions since they do not offer anything special, have so many things that i do not need and also could need special drivers for interact with the Raspberry Pi.

Between the specialized options, even if each one offers its own special capabilities, I prefered to use the official distribution for the Raspberry Pi in order to avoid hardware incompatibilities and lacks of documentation.

3.2 The Language

One of the most important decisions I had to take was which language was the most desirable and fitted better in this project to finish it successfully. There was an important matter to chose one language which provided me a fast development and worked efficiently in the Raspberry Pi.

3.2.1 Why Python?

Finally one of the best alternatives was Python. The main reason is that it is already included in the operating system chosen, Raspbian. Thanks to this we avoided occupying more space in memory with another language libraries. Furthermore, Python provides a

great support and a huge amount of libraries to interact with the operating system and the Raspberry Pi hardware. For these reasons Python grants a fast development.

3.2.2 Consecuences During the Develop

It was difficult to get used to the Python because of my inexperience. Despite of the fact that Python solved so many problems simple and efficiently using the default python libraries like the cron library, the daemon library and the cryptographic library.

3.2.3 Other Alternatives

Other great alternatives were C and JavaScript. The first one provides a low level and so efficient approximation to the problem and it is also by default in the operating system. Despite of this, the develop using C requires more time and I am not familiarizated with it. On the other side JavaScript would be a great option using also NodeJS. Also JavaScript provides great libraries and it is so comfortable for the developer to program the backend and frontend in the same language but it is not focused on interaction with the operating system, there is not information about performance and power consumption in Raspberry Pi and it is not included in the operating system by default.

3.3 The Web Server

This decision was conditioned by the code language chosen. The problem was deciding between all the possibilities based in python assuming also that we need a so efficient and lightweight server. The web server has to provide static web pages. Moreover, the web application needs to have two different sections. On one hand, a public part with the produced data and some graphics. On the other hand, a private part where it is possible to change the device configuration like the power saving mode, the scp address and the frequency of the sensor actions. The scp address is the target machine where the user want to transfer the stored data.

3.3.1 Why Http.server

At the end Http.server was the chosen option. It is a really simple web server included in the standard Python library. It has only the really necessary to works and do not provide anything outside the basic functionalities of a web server. The main feature of this web server that matches perfectly in this project is that it is really lightweight, other lightweight servers are based on Http.server also. In this project we provide a static and so simple web page so we do not need anything else.

3.3.2 Consecuences During the Development

The http.server demonstrated to be so simple to use. The first approach to the coding of the server was a bit difficult, the official documentation was not easy to understand and the necessary functions in the server class were not clear. Anyway, there is so many useful information about how to develop a server using HTTP.server in internet. At the end, to get the basic code run was not so difficult.

One of the challenges using this server was developing a session to ensure the post and to get requests made only from authorized users in order to keep the system safe from

malicious actions. There were not official libraries or capabilities in the HTTP.server to get this and at the end i had to develop a simple session system which provided the necessary functionality to keep the server safe.

3.3.3 Other Alternatives

Python offers a great amount of alternatives to make a simple and lightweight server. So many of these are also based on Http.server. This is the main reason why I prefered to choose http.server.

Between the alternatives we can find Apache, Gunicorn, CherryPy and some more. But many of them spent many resources, others have much more things than needed and a number of them hasn't official support. Kepping in mind that our web application is a static web page I decided finally to choose the simplest and lightest one.

3.3.4 Web Server Parts

The web server is composed by the web server module. It was programmed using the HTTP.server library. The Web_Maker module was one of the first modules created. It uses the html templates to update the content of the static web pages. This mechanism emulates the behaviour of a dynamic web page in the simplest way. Some time later I realized the server was not safe without a session. For this reason I created a new module to manage sessions and keep safe the system from malicious request. The sessions were created using cookies. Furthermore, during the development and test of the server in the Raspberry Pi device I realized that the IP was not static. Every time the IP changed I had to fix all the links in the web application. For this reason and as a complementary work i created another module it. This module changes the static links of the templates and remakes all the html files. The objective of the module is make development and deployment faster. Moreover, it was used later for the installation script, making the installation easier. This script would be even more useful supposing multiple installations of the software. The web server uses this module when it detects an IP change. If the IP changes, the server updates the device name and also the domain name in the html file links. With this action the web application can be visible out of the VUT University without using a static IP.

Chapter 4

The System: Software

4.1 Subsystem: The Temperature

The main objective of the system is to measure the temperature. This was implemented as a subsystem using a deamon. For this purpose, I used the YapDi library, developed by Kasun Herath. As we have explained in section 2.2.3, the used sensor is the digital thermometer DS18B20.

The Subsystem store the samples in plain text data files stored in the data folder of the application. Every sample is in a different line and it is composed by the measure, in celsius degrees, and the time stamp.

This subsystem uses also a frequency configuration file which allows the daemon to update the frequency while it is running and a file size configuration that determined the number of samples in each new file created.

4.1.1 How It Works

Due to the importance of this system, it is always running. Every time the system is booted this daemon is launch after checking if the configuration file existed, else the configuration file is created with a default frequency value of 10 minutes.

When the daemon starts running it gets the frequency from the configuration file. When the last file is created, if it is completed, it means the file has the maximum number of samples permitted by the user and it writes in a configuration file. The daemon will create a new data file using the actual timestamp as part of the name. Otherwise, the daemon will continue using this not full data file for store the samples.

4.2 Subsystem: The SCP

One of the requirements of the system is the necessity of sending periodically the data to a machine. The address of the machine must be configurable by the user in the web application. For this purpose i implemented a daemon which runs in background sending the data periodically.

The daemon was implemented using Python and using the YapDi library, developed by Kasun Herath, which simplifies the configuration of the daemons and permits to program it in python. The Paramiko library simplify the SCP connection development and permit to initialize the SCP connection without write directly the password of the user in the shell.

4.2.1 How It Works

This subsystem only starts working when the configuration is set up in the web page application. For security reasons the SCP data is not stored in the device. Despite this fact, if the system is rebooted this configurations will get lost. This could be undesirable by the user if he wants to enable the power saving mode 3. This mode hibernates the device in a schedule set up by the user. For this reason, the web application permits the user to allow the system to store the scp connection data. In this case the device will check if the scp configuration is stored in the configurations folder and in this case the system will start the daemon after the reboot.

The subsystem wrap an SSH connection to create a SCP connection. If the connection can not be created because the configuration data is not valid the daemon will kill his own process.

Once the connection is created, the daemon will check which files were not sent to the real target machine using a configuration file where the daemon write all the sended files. The daemon will send also the not completed data files but it will update the data file in the target machine once it is completed.

When the file is sent, the daemon will mark the file as "sent" in a configuration file and then it will create a backup of the data file in another directory.

The daemon has a reset function. This function reset the sended files configuration. The daemon use a file where are marked the sended files. When this file is erased the daemon will send all the files another time and mark them as sended again. This configuration file is erased when the target machine changes.

4.3 Subsystem: The web Server

The web server was designed with the intention to show the data in a graphic and in a more understandable way. And furthermore it allows the user to set up the configurations for the rest of the subsystems and also it permits to change the power mode of the device and set up the schedules for the power saving modes.

The web server is made by the use of the standard python library. I used the Http.server library, which provides a lightweight web server with the basic functionality for a static web page.

4.3.1 How It Works

The web server wakes up every time the device is booted in power saving mode 0 or 1. If it is in power saving mode 2, it will wakes up only in the time outside the time intervals setted up by the user.

The web server allows HTTPS connections. To get this, it wraps the HTTP connection with a SSL encryptation. Also I implemented a session system which permits to keep the security in the request made to the server only allowing the logged users to change the configuration.

This is a static web server which means that only it can serve static web pages. This is desirable in our project because that way we do not need a complex web page and also the resources consumed by the server are smaller than in a more complex web server.

The problem of using a static web server is when we need to provide some simple dynamic content. This is not a problem for the configuration section but it was problematic in the

public part of the web application because we needed to show the last samples and update the content periodically.

For this reason, the web server uses a template where it adds the new samples on every get request of the homepage. At first sight, that seems not so efficient but in a web page like this one with a few rate of visits allow us to grant dynamic content without using a dynamic server which is definitely heavier and spends more resources than our static server.

Moreover the server will detect if the Ip has changed since the last boot and in this case, it will remake the web pages links and the device hostname. This is because the device had not any static IP during the developtment. In order to mantain the web page online was needed to remake all the static links of the html templates when the IP changed.

4.3.2 Data Validation

All the data received from the user is validated before the changes of the configuration. For this purpose i used regular expression which only would match with the correct type of data. After checking if the data type was correct there was a second validations. It checked whether the data was between the ranges permitted and if the configurations were consistent and not contradictory.

4.3.3 Session Implementation

To allow secure get and post request, it is needed to implement a session. The problem of using a so simple static web server like http.server is that there is not also a default library that allows to keep sessions. For this reason, I implemented a session using cookies.

Every time a user is logged, the server will generate a cookie which will be sent in the answer headers of the http request.

The value of the cookie is a hash of the current timestamp.

This cookie is stored in the session subsystem and when a request to the server is made, the server will check if the cookie was created in the server and if the cookie still alive.

4.4 Web Application

The objective of the web application is to show the last temperature samples and to allow the configuration of the device parameters from any location at any time.

The web page is composed by two main sections. There is a public section made to show to the visitants the last samples and the graphics which make easier the understanding of the data. And there is also a private section where the users can change the device configurations and control the device behaviour.

One of the most important things when the device has published its address is protect them from intruders and any other attack like man in the midle, service denegation etc.

For this reason, it was necessary to add a login page in order to protect the private section. This section requests the users to introduce their credentials before granting them access to the private section and also permits the server to create a session with a cookie in order to prevent also any request not sent from the web application.

The web application has been designed to be simple and light with the aim of being the most lightweight possible in order to save battery sending the less amount of files possible to answer the requests of the server.

4.4.1 Pages

Home

In the home page, the server shows the last samples in a table and a graphic with the samples shown.

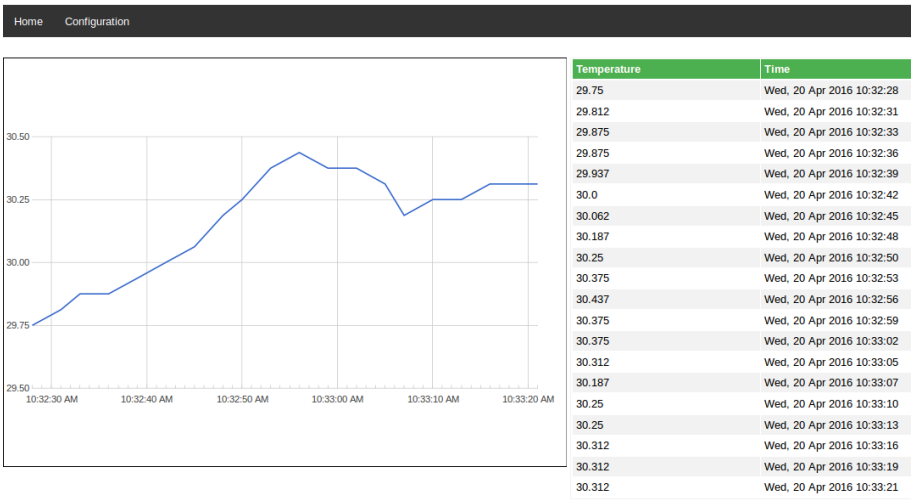


Figure 4.1: Home page in the web application.

Login

The login page allows the user to get access to the configuration page. The implementation of the login was difficult to design. Finally, in the device, the data of the web page credentials for the login is not stored directly. Instead of store the data, the device store a hash of the login credentials. When the user make a post with the login and password, the server hash both and compare with the stored hashes. If the hashes are the same the server will grant access to the user and create a cookie with the session. By this way, if the device is stolen, the login credential will be difficult to get.

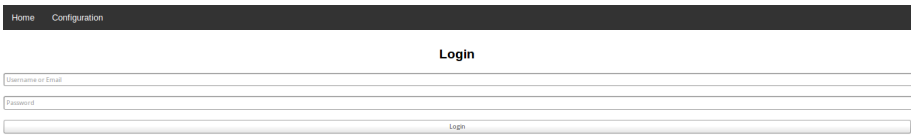


Figure 4.2: Login page in the web application.

Configuration

The configuration page allows to change the device behaviour. The configurations that user can change are the scp target address, the frequency of the samples, the number of samples the web page shows, the size of the data files stored in the device and the power saving mode. In this last configuration, the user can change between four power saving modes, in the mode 2 and mode 3 the user also have three ways of inputting the schedule of the

power saving modes, one where the interval of time is the same for all days. Other where the user can input a different interval for every day of the week. The last one where the user can input multiple intervals for each day.

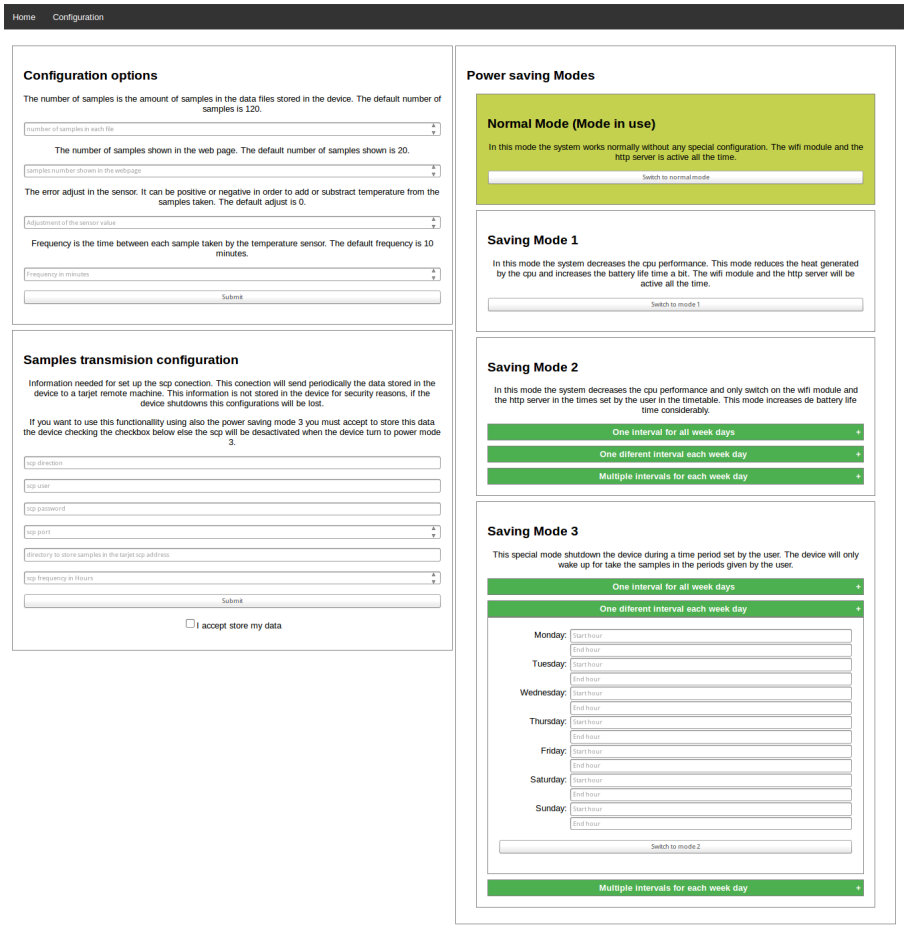


Figure 4.3: Configuration page in the web application.

The configuration page shows always what power saving mode is actually set up in the device configuration. Because the web server only serves static content. There is four html files for the configuration stored in the device. Each one of these files change the actual selected mode. This is not a clever solution for big and complex web pages but in our case there are only four options which only change the CSS class of one element whitouth any other collateral consequence. Thanks to this solution, we can save battery of the device avoiding to add programs which precompile the html files or another complex solutions.

Error Page

Another important matter was giving feedback about errors to the user when the data input was not correct or when an internal problem happens. This is a main topic in a user interface and a static web server which is extremely limited to give feedback to user.

Because of this problem, it is not possible to use the solution implemented in the configuration section to show the current power saving mode because the number of possible states is too big and during the implementation the number of errors handled will grow for

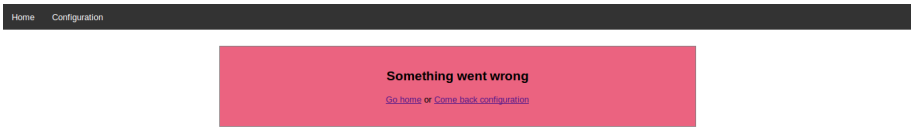


Figure 4.4: Error page in the web application.

sure.

The web application redirects the user to a generic web page with a text explaining the problem.

4.4.2 Scripts

With the aim of saving battery, when we sent this amount of data, the web application was designed as simpliest as posible. The web page try to use all the html5 functions to decrease the number of scripts needed.

For this reason, there are only two scripts used in the web application.

configuration.js

The configuration script is only used in the configuration section. This script has two main functions. On the one hand, the script creates a pop-up asking the confirmation of the user. This confirmation pop-up will appear when the inputed values on a form are possible but not desirable to save battery or the values are in conflict with another configuration value. On the other hand, this script creates the accordions in the second and third power saving mode forms. This accordions provide three different ways to input the power saving mode schedules.

chart.js

This script gets the values of the samples directly from the html table of the html file and using that data, the script generates the graphic in the home page.

Also, the script control the size of the window resizing the graphic for keep the responsive dessign.

4.4.3 Styles

The web application is concerned about the functionality. But also the dessign was made to keep in mind the user experience. Thee dessign was made also to avoid send more data than necessary in every server request answer because there is a strict requirement of energetic efficiency. For that reason, there is no big dessign libraries stored in the device and the design strictly respect the KISS principle.

The style is provided only with a simple css style sheet for all the web application which control the position of the elements and provide some media queries which implement a responsive design.

Responsive

Nowadays, the way of using Internet is so different from four or five years ago. The smart phones changed completely the way to use web pages and now it is mandatory to dessign not only a desktop web page but also a mobile dessign prepared for small screens.

To achieve this objective, there are media queries implemented in the style sheet which permit to adapt the content to smaller screens changing what is showed. This modification of the content structure keeps the web dessign understandable and clean.

The Style Sheet

The style sheet is stored in the css folder, in the project folder. In the style sheet there is coded a generic dessign for all the web application and a specific dessigns for each web page.

4.4.4 Libraries

The web application has two dependencies with external libraries. One of the libraries JQuery and the other one is Google charts tool.

The web page uses this two libraries because are stored in remote servers. That is a great advantage because we don't need to store it. And also, and the most important reason, because the device doesn't need to waste energy sending the library files on each client request to the server which is a way to save battery and also computational power.

4.5 Power Saving Modes

The system implements four power saving modes which change the configuration of the device. These power saving modes enable or disable some device configuration, hardware components and running programs [15]. These changes lower the device functionalities but decrese the power consumption in order to increase the battery life time.

4.5.1 Why?

The project is designed to stay working using a battery as a power supply during the most time as possible. The big problem of this kind of system is the battery life time because they are thought to be working during long periods. Normally, these kind of systems work in devices with few computational power but with lower power consumption. One example of these devices is Arduino. But with the arrival of the Internet of the things (IoT) it arrives also the necessity and also the possibility of more powerful devices like Raspberry Pi.

It is true that the Raspberry Pi is a device which consumes so few power, and a big part of the Raspberry Pi community thinks that trying to decrease the power consumption is a non sense. Contrary to this point of view, this thesis tries to optimize the power consumption of the device as much as possible in order to increase the battery life time of the device trying in the same process to test the raspberry pi as an embedded system.

This is not an easy task because even if Raspberry Pi has a lower consumption for a computer, it continues consuming so much if we compare this device to other like the before mentioned Arduino.

Making a Raspberry Pi to consume the same or less power than systems like Arduino is of course impossible and the aim of this thesis is not achieve it. The two types of systems

are extremely different and the objective of both systems is not the same. The Raspberry pi is for sure more powerful and it is a multipurpose device which can work as a multimedia station or like an embedded system like this. But the chips like Arduino are designed to be only embedded systems and for sure are better option if we only think of the power consumption. The problem of these chips is the lack of computational power which does not permit them to create a web server or to work above a complex operating system like Raspbian. This fact is not trivial since a complete operating system grants lots of services, security systems and also makes the develop of applications significatively faster.

With this facts in mind the next power saving modes were developed.

4.5.2 Description

All the power modes were implemented using bash scripts which can be used from any directory of the device if they are placed in the /bin folder of the device user.

This scripts also call another python scripts if it is necessary and principally enable or disable the power of the hardware components which are not needed in the device.

The power mode scripts not only disable the ethernet interface saving computational resources, but they also disable the bus which gives power to the component even when the interface is down saving in this way the power destinated to that interface.

There are hardware components which are always disabled because they are not useful for this system like the HDMI port or the ethernet port. But for security reasons there is a power saving mode which enables all the disabled components if necessary.

4.5.3 When

Every power mode is designed to work in some specific part of the lifecycle of the device. The default power saving mode, which will be active if there is not a configuration about that. This is the first power saving mode.

The zero power saving mode is only thought for special moments where the hdmi port or the ethernet wire could be necessary but it is not thought for the normal lifecycle of the device.

The other two power saving modes are designed as hard saving power modes. These modes decrease the device functionalities but increase the battery life time. Moreover, they will disable the Internet connection and also the web server in order to save the most battery possible. These modes are designed to work following a schedule set up by the user. The schedulle can not ocuppy a complete day in order to let the user change to other modes and recover the control of the device at least a couple of hours every day.

4.5.4 PM0

The zero power mode is thought as a special mode which will enable all the default hardware functionalities. The default hardware functionalities are the ones which are enabled in a clean operative system. For examples USB ports, the HDMI port and more. This mode will increase the battery consumption until the maximum. This is the reason why this mode is not designed to be working in the device.

This mode is thought for maintenance of the device when it is necessary to connect the device to a screen and the HDMI port is needed to be enabled.

It is not recommendable to use this mode in the normal lifecycle of the device.

4.5.5 PM1

The first power consumption mode is the default power saving mode in the device, it will be set if there is not another configuration and this mode will be automatically enabled when the device goes out of the second or third power saving mode.

This power saving mode decreases the frequency of the CPU in order to low the power consumption. The truth is that the power consumption only decreases a bit but the collateral fact is that the heat generated for the CPU decrease also.

Even if it low the power consumption only a few with the lower CPU frequency. It is better than nothing and the heat decreasing is useful in order to increase the life of the device and in order to avoid affecting the temperature sensor samples.

Of course, this power saving mode also disables the HDMI and the ethernet port. It also would be so useful to disable the USB hub letting active only the USB port used by the Wifi module but the way it is designed. the hardware does not allow to make this because the USB ports are not independent ones from others, they are all connected to the same master usb root port like it is explain in the 2.1.3 section.

4.5.6 PM2

This power saving mode was designed to think in extend the battery life lowering the device functionalities even if the connection with the device is lost in order to make possible to get a device which can be working during longer periods.

The main fact in this power saving mode is that the Wi-Fi module is disabled and the user won't be able of connect with the device to change the configurations. If this power saving mode was working without stopping, it would not be useful because the user would need in some moment to be physically in front of the device, to connect it to a screen and change the power mode manually.

If this mode is enabled for a complete day, it would not be possible to change the power saving mode. In that case the user would need to get physically the device to change the configuration. In order to avoid this, the second power mode can not be running the entire day.

The user must set up a schedule for this power saving mode in the configuration page of the web application. This schedulle is implemented using a crontab. Crontabs are managed by the operating system which will execute the crontab commands when the right time arrives. Every job in the crontab will execute a script which sets up the power saving mode configurations. The schedule will restrict the maximum time which the device can be in the second power saving mode. 23 hours is the maximum time for a day, it permits the user to change the configuration at least one hour every day.

This power saving mode will disable the same things disabled in the first power saving mode and it will also disable the USB hub, the Wifi interface and the web server.

As an exception, if the scp subsystem is running, it will detect if the second power saving mode is enabled. In this case, the subsystem will enable the Wi-Fi interface in order to send the data to the target machine address. Then, it will disable the Wi-Fi interface again.

4.5.7 PM3

In a harware system like Raspberry Pi is not possible to let running only the temperature subsystem and nothing more. To extend the battery life time, we can try to use a special

operating system prepared only to execute our subsystem. The problem of this kind of operating systems is their capabilities. If it is only prepared to run our subsystem and nothing more, it would not support another general functions or the web server. In addition develop a system in this plataforms would be longer and more difficult.

An alternative to this is use a Arduino board connected to the Raspberry Pi. The Arduino board, which consume less power than the Raspberry Pi, would run the temperature subsystem. In this way, the Raspberry Pi would wake up only when the web server or the scp subsystem is needed.

With the objective to increase the battery life time extremely the third power saving mode will shutdown the device setting an alarm in a RTC module which will wake up the device just before take a temperature sample.

When the device wakes up, we first take the sample and then, we check if in the schedule set up by the user it is time to change to another power saving mode or if the system has to shut down itself another time until next sample time. If it has to shutdown again, before this, the system set up a new alarm in the RTC which only can store one alarm at the same time.

Like the second power saving mode, the third power saving mode follows a schedule set up by the user with the same rules and reasons as the second power saving mode.

In this power saving mode, the SCP subsystem will work only if the user marks the checkbox of the configuration web application. So he will accept store the scp configuration data in the device. If not, the data will be lost on every shut down the device will do. Also if the scp configuration is stored in the device, the scp data transfer frequency will not be followed and the subsystem will send the new data every time the device wakes up in order to do not lose data.

4.6 Boot

As one important part of the system, there is a boot script on which every boot of the device will check the configurations and will configure the device in order to keep the device working in the proper way.

This script is written in bash and it is called from /etc/profile configuration file on every boot. when the script is called, it will change the actual location to the system folder and then, it will execute the weaterStation.py python script. This last script is the one which will set up the actual configurations in the subsystems and also change the hostname of the device and the links of the html files in order to keep the device accessible from outside the university network.

Chapter 5

Life Cycle

The system is designed to work in an unstoppable way. During the working time, the system will determine the state according to the power saving modes and the user new configurations.

Also, the system is conformed by three semi-independent subsystems which will perform different tasks. Every subsystem has its own lifecycle that will be conditioned by the general system lifecycle and the user inputs.

5.1 Life Cycle Attending to Power Modes

The lifecycle of the system is determinated by the actual power mode, the change between power modes modifies the behaviour and the functionalities of the system. Also, the change between power modes will determine directly the movement between states in the subsystems.

Initially, the system will boot, in this moment the system will execute the boot script, which will make the system jump to the actual state checking up the real power mode to set up the right configurations in the system according to the actual power mode.

The number of possible initial states where the system can jump after the boot checks are four and represent the four different power saving modes.

5.1.1 Change Between States

The are two possible ways to change between states. One is manually; the user will modify the state of the system directly, the state of the system will not change until the user modify it again. The other one is automatically, the user will input some configuration data. This data will determinate the next state changes witouth require the user actions.

Manual Changes

In this system the user is capable of changing the device state manually during certain intervals of time.

For this aim, the user can use the web application in order to change the configuration of the device. The web configuration page has two types of possible configurations. On one side, there are the system state configurations which will change the system state. On the other side, there are the subsystem configurations which will affect the subsystems behaviour, but not their state.

The configurations that will affect the system state are the power saving mode configurations. Setting up the different power saving modes the system will jump between states affecting the behaviour and the functionalities of the system.

Depending on the state, it is possible that the user will not be able to access the web application. In this case the user will not be able to change the state. This happens in the second power saving mode. In that state the system disable the network interface to save battery. This happens also in the third power saving mode, which shutdown the device until the time of take another sample.

Schedulled Changes

The scheduled or automatical state change happens in the second and third power saving mode when the user had set up a schedule which will determine the state of the system depending on the actual date of the device.

For this reason, the device will write a crontab which will automatically change the state of the device following the user input.

The change between states is limited on this mode and also depends if the enabled power saving mode is the second one or the third one.

In the scheduled changes of the second power saving mode the possible jumps are from the second power saving mode state to the first power saving mode state and from the first state to the second.

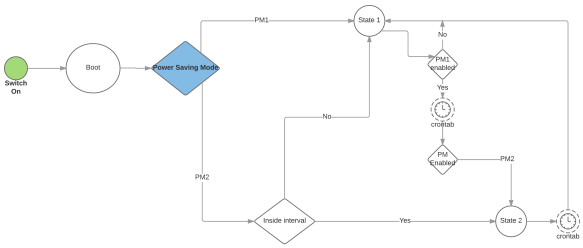


Figure 5.1: Second power mode state diagram.

Also in the scheduled changes of the third power saving mode, the possible jumps between states are from the third state to the sleep state and from the sleep state to the first state.

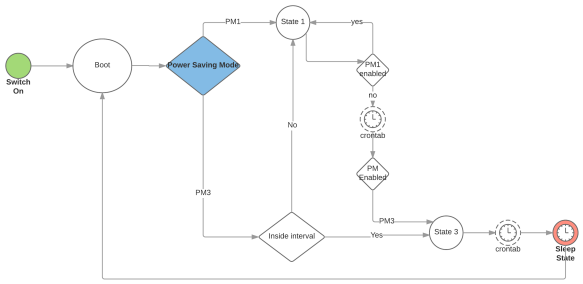


Figure 5.2: Third power mode state diagram.

5.1.2 States Diagram

How it is possible to watch in the figure 5.3 once the device is switch on it enter in the boot state where the system checks the configuration files. In function of the actual power saving mode state, the system can jump to 4 different states. The state zero corresponds to the normal power saving mode which represents the device without any special configuration in order to save battery life was explained in section 4.5.4. In the same way, the state one corresponds to first power saving mode, state two to the second power saving mode and state three to the third power saving mode.

From the state 0 it is not possible to change to another state without the interaction of the user in the web application.

From the state one, it is possible to jump to state two if the second power mode is enabled or jump to the state three if the third power saving modes is enabled. Both jumps happen when a scheduled cron job occurs. If the enabled power saving mode is the first one, it is not possible to jump to another states.

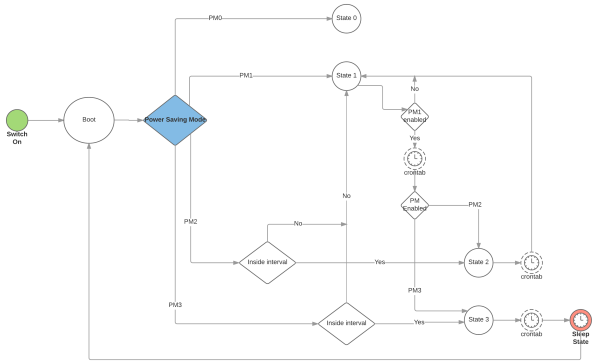


Figure 5.3: Complete state diagram.

From the state two, is possible to jump to the state onw when the scheduled cron job happens.

And from state three when the scheduled cron job occurs, the system will jump to a sleeping state. In this state, the device is shut down waiting for the RTC clock module alarm to wake up. When the RTC alarm occurs, the device wakes up and it enters in the boot state where the state will redirect the system to the state zero or to the state three, depending on the device time and the scheduled cron jobs for the device.

5.2 Life Cycle Attending to Subsystems

The system is composed by three main subsystems as it is written in the chapter 4. These principal subsystems are the temperature subsystem which measure the temperature every certain time set up by the user. The scp subsystem transfers the raw data to a target machine. And also the web server subsystem, which shows the data and allow the user to change the device configuration. Every subsystem has its own lifecycle directly influenced by the general system lifecycle and the user configurations.

5.2.1 Temperature and SCP Subsystems Lifecycle

This two subsystems share the basic structure of their code. That is the reason why I prefer to explain their lifecycles as the same, because the states where they can stay are the same. The only difference between their lifecycles is the event which starts the lifecycle.

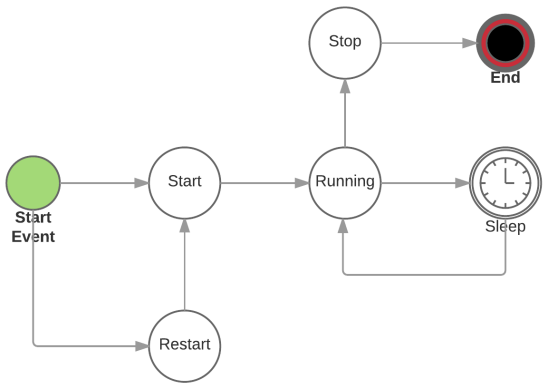


Figure 5.4: SCP and Temperature subsystems state diagram.

Firstly, the start event in these two subsystems is a call from the boot state of the system. The boot state checks the configurations and depending on this configuration value it will wake up the subsystems or not. In the temperature sensor case, the start event will always come from the boot state of the system. The boot state will wake up the temperature subsystem always. On the other hand, the boot state will wake up the scp system only if any configuration file which contains the necessary data for it. If this configuration file does not exist, the only other event which can start the scp lifecycle is the setup of the scp data in the configuration section of the web application.

After this start event, the lifecycle diagram is completely the same for both. Once the initial event wakes up the subsystem, it jumps to the start state. In this state the subsystem checks some configurations and make some initial task in order to initialize itself. Then the subsystem jumps to the running state where it performs it task and jumps to a sleep state during a certain time set up by the user in the configurations. When the subsystem wake up from sleep it make another time the same task and return to the sleep state in a infinite loop. This loop only will be break when the system kill the subsystem directly or in a indirect way when it shut down the device.

5.2.2 Web Server Subsystem Lifecycle

The web server subsystem has a quite easy lifecycle diagram. As a server, when it starts running, it will be working non-stoping until the system will kill it.

The subsystem will be waken up by the system only when the system jumps to the state zero or the state one. That means that the web server will be wake up in all the power saving modes when they are in state one, or in state zero.

And also, the subsystem process will be killed when the system jumps to state two or state three which means that there is no network interface enabled. For this reason, there is no sense in keeping the process alive.

Chapter 6

Conclusions

6.1 Conclusions

Despite of the short period that this thesis last, it is easy to substract conclusions of the work made. Talking about the hardware, it is clear that the Raspberry Pi is not an embedded system. Try to use it as a traditional embedded system is possible but it will be a waste of resources. In addition, it will be dificult since the configurations needed to try emulate a real embedded system are not a few. Even so, the best Rasperry Pi model for emulate it is the A+ one or the new Zero model. These modeles are more suitable than the rest of the models because the hardware characteristics are simplier and because of this the power consumption is also lower.

The Raspberry Pi is a flexible device. It is capable of work in most of the fields since its little size and cheap prize. Even more, this device permits to think in a new kind of proyects because of it CPU, powerful than any other embedded system.

Furthermore, it was clear that a Respberry Pi is not a suitable device for dessign a system which depends on a battery for get power. The Raspberry Pi consume few power comparing on a normal computer, but it continues consuming much power if we compare on a embedded chip. At the end, it is not capable of being working for long terms with not more hardware than the default one in the board. As maximum, the device will be able to run for many days or a few months.

Depending on the project, it can be enough, but for out project is a quite sort period. Anyway, using external hardware, which improve the power management of the Raspberry [20], Pi it could be get. But unfortunately, the external hardware for get this point was not capable of let us test if it is possible to keep running our Raspberry Pi for long terms using an hibernation power mode. On one hand, there are examples of how to make this with expensive hardware shields for Raspberry Pi. But on the other hand, there was not enough information about how to wake up the Raspberry Pi using only a RTC module alarm.

Now, changing the topic to the software part of the project. The system was designed as a standalone system which is prepared for make all the task in a unique device. Raspberry Pi is a dessirable device to be working as a standalone system. But the dessign of a wireless standalone system in a little device which will be in public places creates to many new problems. Firstly, the battery use, a standalone system needs to do all the task by itself. This fact supposses spend more energy with every new functionality that the system is capable to do. Second, the web page does not provide a good user experience since the web page is online only few hours every day. In the second and third power saving mode the configuration was chosen thinking in save energy instead of provide a good user experience.

For these reason in some periods the configuration is not accesible. And finally, keep safe the data inside the device is not possible in a standalone system like ours. The device can be physically stealed, that is because the system needs to create keys and keep them safe to encrypt and decrypt private information. Also, this device does not have password because needs to wake up automatically. At the end, anyone with the device in his hands can find the keys for decrypt the private information.

To conclude, we have dessign a complex standalone system. This system is capable to make all the task by itself providing information in many ways. This system also grants the user many possible configuration and optional functions. Even so, the system is robust and can endure a non-stopping working without errors. The main points cover by this thesis are the power efficiency management, the secureness of the stored data and the connection between devices. For get this points, we have fight agains all the difficulties present in the IoT devices. And at the end, it was possible to cover all these points in order to dessign a efficient and powerful system.

6.2 Improvements

As all software developments, always there are new task to do and new improvements to implement. One easy improvement is allow user to set up more than one scp address in order to send the temperature data. In IoT networks, there are devices which create new data and devices that use the data generated by others. Allowing set up more than one SCP address would allows to create a bigger network allowing different devices to perform different task each one.

Moreover, our device is only a temperature station, but it is possible to add more sensors [14] to detect also the humidity levels, wind force and other weather parameters which can provide new data. Treat all of these different data can provide new useful information.

During the development, I realiced that it is not possible to dissable the energy provided to the USB ports of the Rasperry Pi, if we want to keep the Wi-Fi connection. But it is possible to disable the data transfer in these USB ports. The device will continue spending the same energy, but steal physically the data will be imposible using the ports which are not in use.

Also, it could be interesting to continue investigating in the wake up using only a RTC. There is also expensive options which can make this, but this options also has many components that are not useful for projects like ours. Because of this, to get a automatic wake up using only a RTC module like pcf85163 would be a great advantaje.

Other way to improve the system is connecting a solar panel to the device. With a solar panel the device will recharge the battery during the day. It is a simple solution to keep the device working more time.

Finally, it would be useful to get advantaje of the computational power of the Raspberry Pi. The device is capable to generate more complex graphics. It can treat the data in order to provide more useful information and it can generate stadistical predictions.

6.3 Real World Uses

One of the best points of use a Raspberry Pi is the flexibility that it provides to develop any kinf of systems. A temperature station like in our project can be used in many fields. The most basic example is only generate data and send it to any other device. Then the

device which recives the data will treat it and decide what to do with it depending of the last values.

There are also more examples, our system could be part of a inteligent thermostat. In the same way that the device changes the system configuration, it will change the heating in a room depending on the last temperature samples and current time. An intelligent thermostat is a great way to make our heater more efficient and also to save money.

Lastly, our system also could use the data of the temperature samples to control the traffic screens which notify about the dangers in the road. If we give the control of the LED screen to the Raspberry Pi board it will writte notifications in function of the sensor values. For example, if it is a minus zero degrees, the system will write that it is dangerous to drive fastly in the road because there is danger of frozen road.

Bibliography

[1] *DS18B20*. Maxim Integrated. [Online; navštíveno 25.2.2016].

[2] *PCF85163, Real-time clock and calendar*. Phillips, July 2010.

[3] *Raspberry Pi 2 Model B*. The Raspberry Foundation, March 2016.

[4] *Raspberry Pi 3 is out now! Specs, benchmarks & more*. Magpi, March 2016.

[5] Abelectronics. *Demo files for Alarm Pi board from AB Electronics UK*. abelectronicsuk. [Online; navštíveno 15.5.2016].

[6] Wikipedia Community. *Raspberry Pi*. wikipedia. [Online; navštíveno 15.3.2016].

[7] Wikipedia Community. *Wpa_supplicant*. wikipedia. [Online; navštíveno 20.4.2016].

[8] Wikipedia Community. *Internet of the Things*. wikipedia, May 2016. [Online; navštíveno 14.5.2016].

[9] Raspberry Pi Foundation. *About us*. Raspberry Pi Foundation. [Online; navštíveno 15.3.2016].

[10] Raspberry Pi Foundation. *USB*. Raspberry Pi Foundation. [Online; navštíveno 7.4.2016].

[11] Javier Lacort. *Así son las primeras ciudades inteligentes de España que dibuja Telefónica*. hipertextual, November 2014. [Online; navštíveno 12.5.2016].

[12] Mahjongg. *RPiconfig*. Ellinux, May 2016.

[13] Markt Marshall. *Experimenting with Raspberry Pi power management*, August 2014. [Online; navštíveno 7.4.2016].

[14] Javier Pastor. *Las 13 mejores ideas que hemos encontrado hechas con Raspberry Pi*. Xataka, June 2015. [Online; navštíveno 10.5.2016].

[15] samirsogay. *Power Saving Tips for Raspberry Pi*. Baba AweSam, January 2014. [Online; navštíveno 15.3.2016].

[16] Aaron Shaw. *Adding an On/Off switch to your Raspberry Pi*, November 2013. [Online; navštíveno 14.5.2016].

[17] Olivia Solon. *Raspberry Pi's Eben Upton: we need to create a generation of producers not consumers*. Wired, October 2013. [Online; navštíveno 25.2.2016].

[18] Stocksy. *Raspberry Pi Setup*, October 2012.

[19] swarren, eshizhan, lauraclay, lurch, bennuttall, and pelwell. *Device Trees, overlays and parameters*. Raspberry Pi Foundation, May 2016.

[20] Tom. *Reducing power consumption of a raspberry Pi*. Bitwizard, August 2014.

Appendices

List of Appendices

Appendix A

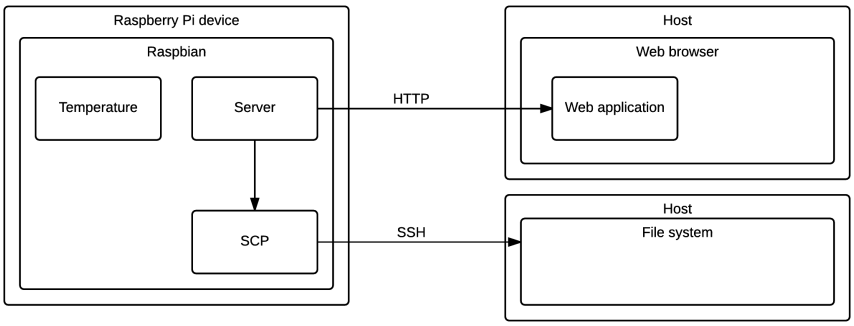
CD Content

```
CD
 |-- weatherStation
 |      |-- lib
 |      |    |-- CookieStorage.py
 |      |    |-- transfer.py
 |      |    |-- utils.py
 |      |    `-- web_maker.py
 |      |-- README.md
 |      |-- scpdaemon.py
 |      |-- scripts
 |      |    |-- boot.sh
 |      |    |-- pm1
 |      |    |-- pm2
 |      |    |-- pm3
 |      |    |-- pmnormal
 |      |    |-- setup.sh
 |      |    `-- wpa_supplicant.conf
 |      |-- server.py
 |      |-- tempdaemon.py
 |      |-- weatherStation.py
 |      `-- web
 |      |-- css
 |      |    `-- style.css
 |      |-- html
 |      |    `-- templates
 |      |        |-- configuration-changed.html
 |      |        |-- configuration-fail.html
 |      |        |-- configuration.html
 |      |        |-- configuration_mode1.html
 |      |        |-- configuration_mode2.html
 |      |        |-- configuration_mode3.html
 |      |        |-- empty_web.html
 |      |        |-- login-fail.html
 |      |        |-- login.html
 |      |        |-- session-fail.html
 |      |        `-- web_bone.html
 |      `-- js
 |          |-- chart.js
 |          `-- configuration.js
 `-- thesis
```

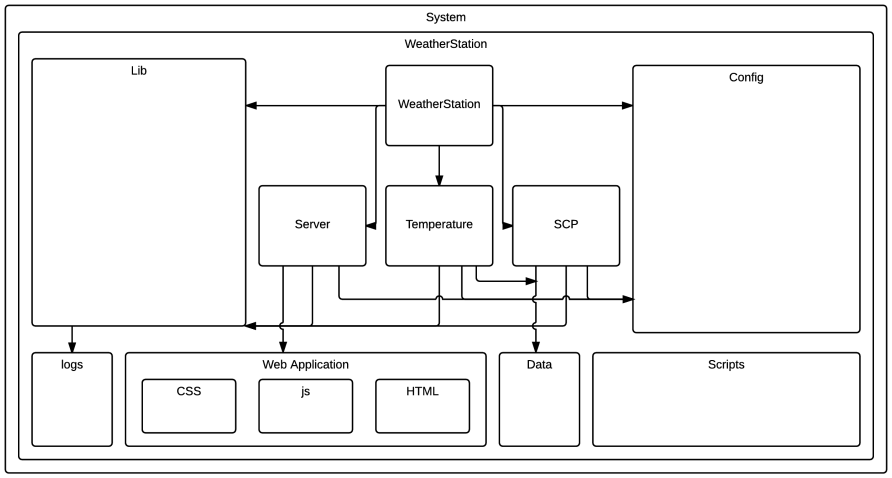
Appendix B

Design Diagrams

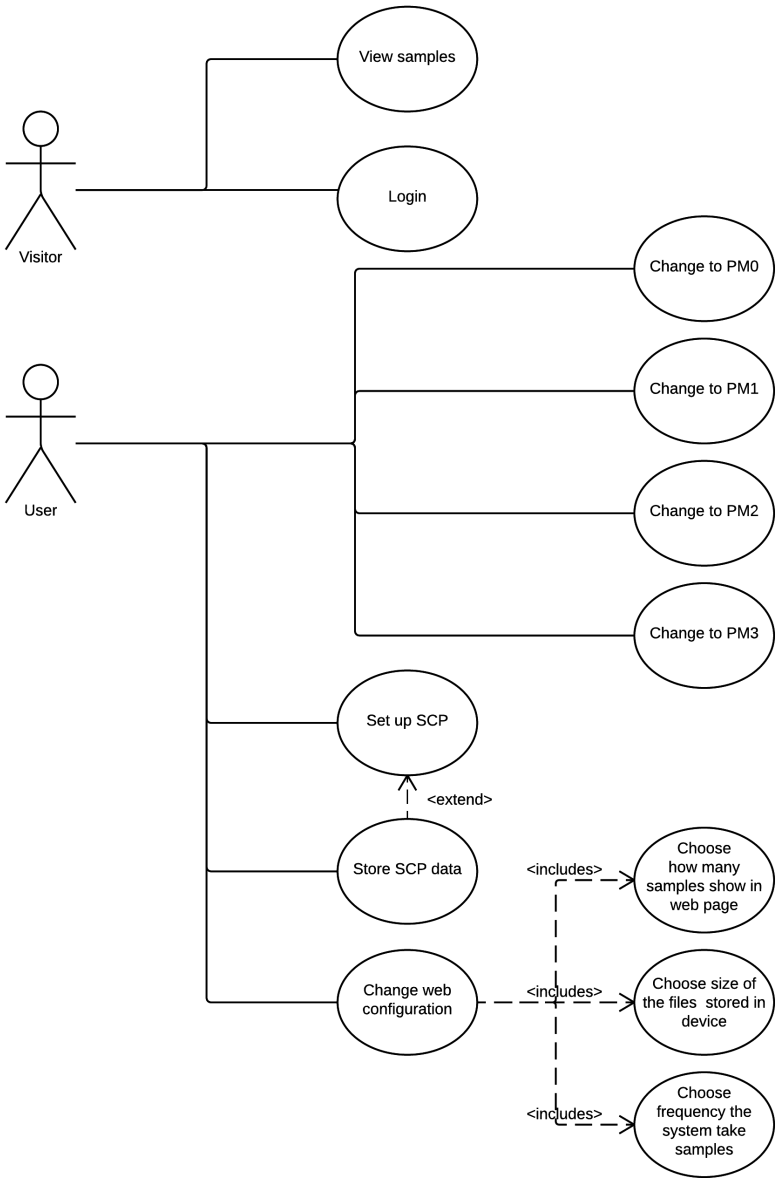
B.1 Deploy Diagram



B.2 Dependencies Diagram



B.3 User Case Diagram



Appendix C

Installation Script

Firstly, it is needed to configure the Raspberry Pi device. First change the user name. Then in the raspi-config enable the ssh and the autologin options. After that, it is necessary to activate the Wi-Fi connection as explained in the subsection 3.1.2 and install the module as explained in section 2.2 before exececute the installation script. This script will install all the dependencies with libraries. Is was thought for work properly in Raspbian Jessie Lite. It will propably works properly in most of the Linux flavour but it has been tested only in Raspbian Jessie Lite.

```
1  #!/bin/sh
2
3  ACTUAL=$(pwd)
4
5  sudo apt-get update
6  sudo apt-get install git
7  sudo apt-get install python-pip python-crypto python-dev libgmp-dev
       cpufrequtils
8  sudo apt-get install build-essential libffi-dev libssl-dev
9  sudo pip install python-crontab
10 sudo pip install cryptography
11 sudo pip install paramiko
12
13 #Download project
14 echo "Downloading weather station code from Github."
15 sudo git clone https://github.com/coke727/embeddedServer.git
16
17 #Creating power saving modes script folder
18 mkdir /home/$USER/bin
19 cp -a $ACTUAL/embeddedServer/scripts/pmnormal /home/$USER/bin
20 cp -a $ACTUAL/embeddedServer/scripts/pm1 /home/$USER/bin
21 cp -a $ACTUAL/embeddedServer/scripts/pm2 /home/$USER/bin
22 cp -a $ACTUAL/embeddedServer/scripts/pm3 /home/$USER/bin
23 chmod 777 /home/$USER/bin/*
24
25 #Adding server execution to boot
26 if grep -q boot.sh "/etc/profile"; then
27   echo "[Warning!] The init weather station script is already in /etc/
         profile"
28 else
29   echo "Adding weather station to /etc/profile"
30   sudo echo "" >> /etc/profile
31   sudo echo -n "sudo sh "$ACTUAL"/embeddedServer/scripts/boot.sh" >> /
         etc/profile
```

```
32    sudo echo -e "\n" >> /etc/profile
33    fi
34
35  #Create temporal data dirs.
36  echo "Creating temporal data directories."
37  sudo mkdir -p $ACTUAL"/embeddedServer/logs" $ACTUAL"/embeddedServer/
        config" $ACTUAL"/embeddedServer/data/backup"
```

After install the system it is needed to download the YapDi library manually.

```
1  sudo git clone https://github.com/kasun/YapDi.git
2  sudo python YapDi/setup.py install
```

Before finish the setup, we need to add our script folder to the `/etc/sudoers` file. In order to get this, it is needed to add `/home/$USERNAME/bin` to the following line in the file. Realize that "$USERNAME" is the name of the current user in the system.

```
1  Defaults   secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
       bin:/sbin:/bin"
```

The result will be the following:

```
1  Defaults   secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
       bin:/sbin:/bin:/home/\$USERNAME/bin"
```

Finally, reboot the system.

```
1  sudo reboot
```

Appendix D

Wpa_supplicant Template

```
country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
        ssid="eduroam"
        eap=TTLS
        key_mgmt=WPA-EAP
        anonymous_identity=""
        identity=""
        password=""
        phase2="auth=PAP"
}
```