Internet of Things: WLAN Connected Thermostat

Auteur : Takougoum Fonkoua, Thierry
Promoteur(s) : Kraft, Michael
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil électricien, à finalité approfondie
Année académique : 2015-2016
URI/URL : http://hdl.handle.net/2268.2/1651

Faculty of Applied Science
Department of Electrical Engineering and Computer Science

Internet of Things: WLAN Connected Thermostat

Author:
Thierry Takougoum Fonkoua

A thesis submitted in fulfillment of the requirements for the degree of Master in Electrical Engineering

2015-2016 Academic Year

Faculty of Applied Science
Department of Electrical Engineering and Computer Science

Internet of Things: WLAN Connected Thermostat

Author:
Thierry Takougoum Fonkoua

A thesis submitted in fulfillment of the requirements for the degree of Master in Electrical Engineering

Panel of examiners:

Prof. Michael Kraft	University of Liège	Supervisor
Prof. Philippe Vanderbemden	University of Liège	Examiner
Prof. Benoit Vanderheyden	University of Liège	Examiner
Dr. Fabrice Axisa	University of Liège	Examiner
Dr. Philippe Laurent	University of Liège	Examiner

2015-2016 Academic Year

Abstract

In the context of the emerging Internet of Things, the project aims at investigating whether a standard programmable radiator thermostat can be converted into a WLAN connected device that can be connected to a standard home WiFi router. A software application is supposed to be developed to control the thermostat remotely.

Based on the main features of electronic thermostats on the market, it was possible to define a basic model . The ESP8266-01, a low-power WiFi transceiver was then added to the basic module to add the connectivity. The relatively low cost of the WiFi module has allowed to show that it is possible to develop low cost connected objects and free them from a central gateway that is found very often in this type of projects.

A web application was develop to control remotely the electronic board via the Internet. The database behind the web application allows the thermostat and users to easily exchange data. To reduce the power consumption, the micro-controller and WiFi module programs allow them to be in the sleep mode much of the time . They come out of this mode to execute an operation and then immediately go back in the sleep mode. In addition a small learning algorithm was develop to allow the thermostat to function fairly accurately.

The prototype test stages highlight the main features of the thermostat. It is a low power device powered by battery. It can be connected directly to the WLAN, can automatically get an IP address. It can be controlled from any place through the Internet. A comparative study between its operation and two other types of thermostat allowed to measure the potential energy savings it brings.

Acknowledgements

I would like to express my special thanks to Prof. Michael Kraft for giving me the opportunity to work in this thesis and for all his advices.

I would also want to express my gratitude to all the staff of the Montefiore Institute for all the knowledge and experience they brought me.

Finally I would like to thanks my family for their encouragements, especially my wife for the support for years.

Contents

List of Figures

List of Tables

List of Abbreviations

IoT	**I**nternet **o**f **T**hings
HVAC	**H**eating **V**entilation **A**nd **C**ooling
HMI	**H**uman- **M**achine **I**nterface
WLAN	**W**ireless-**L**ocal **A**rea **N**etwork
BPM	**B**usiness **P**rocess **M**anagement
BRM	**B**usiness **R**ules **M**anagement
BSS	**B**illing **S**upport **S**ystem
OSS	**O**perational **S**upport **S**ystem
MCU	**M**icro-**C**ontroller **U**nit
ICSP	**I**n-**C**ircuit **S**erial **P**rogramming
GPIO	**G**eneral **P**urpose **I**nput **O**utput
UART	**U**niversal **A**synchronous **R**eceiver/**T**ransmitter
RX	**R**eceiver
TX	**T**ransmitter

‘

Chapter 1

Introduction

1.1 Background

In recent years, the field of home automation has been subject of a particular interest to researchers, manufacturers as well as consumers who are recipients of new products. This interest can be explained primarily by the contribution in terms of comfort and time savings [2] associated to these products. Although it was destined to a bright future, home automation has never fully imposed itself in homes. Is this due to the technology? Is this due to a simple problem of acceptance? Today, thanks to the significant development of the Internet and telecommunications technologies, and particularly with the democratization of some technologies across smart-phones, one can witness a new orientation of a number of research in the field of home automation. For some researchers, the aim of technological development must not simply be the improvement of human comfort but should also help making a healthier environment.

Nowadays, people are more likely to be aware that the future of energy is based on efficient use of that energy today. This awareness about energy consumption comes from concern about its availability and its environmental impact, and has renewed interest in finding more efficient ways to use it. Everything now goes through ways to consume less and better. This efficient management should not be limited only to the industry but to all energy consumption sectors and therefore the applications and consumer equipments. Different scientific researches aim to contribute to the improvement of energy management in general and particularly in home HVAC (Heating Ventilation and Cooling) systems.

1.2 Problem

The standard of living today can not be sustained without a sustainable management of energy resources. Knowing the consequences of poor management of these resources (blackout, energy war, ...), professionals encourage the use of more energy-efficient equipments and also a responsibility of all users (industrial , large public ...). For Jiakang Lu [15], HVAC are the largest source of residential energy consumption. He demonstrates that using technology and investing only $25 in sensors, he can get to up to 28% energy saving on average. His study also notes that conventional systems often allow to reduce by around 7% and in some cases are also responsible for the increase of energy consumption.

Many electronic thermostats are based on the ability to program their operation in advance. In fact they usually offer the ability to preset the operation for different time periods (eg period of six hours, days, weeks,...). The problem is that the thermostat does not follow in case of weather improvement and therefore can cause unnecessary energy expenditure. Furthermore, practically all current "smart" radiator control valves require a central gateway box that connects to the Internet. This makes such solutions relatively expensive for the end user.

1.3 Goal

In the context of the emerging Internet of Things (IoT) also called Internet of Objects, the project aims at investigating whether a standard programmable radiator thermostat can be converted into a WLAN connected device that can directly be connected to a standard home Wifi router. The idea is to build a digital thermostat prototype similar to those found on the market but which will add a module to connect to the Internet. A software application is supposed to be developed that can control the thermostat remotely.

One problem with many numeric thermostats on the market is that they often lack accuracy. The project also involves the implementation of a learning algorithm that will allow thermostat to set the temperature with more accuracy and facilitate its adaptation to any local.

In addition to the main features already defined , the following must apply to the system:

- low power device
- battery powered

- WLAN connected
- automatically obtain an IP address
- remote control
- similar size as those on the market
- low cost

The goal is achieved when the above requirements are met in the design.

1.4 Outline of the work

The thesis describes the design of the hardware and software parts of a connected thermostat for home radiators. Results of tests on the designed board are presented and discussed. The thesis is organised as follow:

- Chapter 2 describes the main concepts of interest (IoT, thermostats) and presents a short review of the state of the art about connected thermostats.
- Chapter 3 is devoted to the design of the hardware part. A design by functional blocks is implemented and main components are chosen.
- In Chapter 4 we discuss the software part of the project. Some approaches of IoT software will be presented. A suitable one for the project will be chosen.
- Chapter 5 presents the experimental results and a discussion about the performances of the thermostat.
- The last chapter, Chapter 6 gives the general conclusion of the thesis and presents some ideas for future researches.

Chapter 2

Background Iinformation and Theory

2.1 IoT: Intenet of Things

2.1.1 History and Definitions

The Internet is nowadays the most powerful tool for information sharing. It has led to profound changes in almost all aspects of human life. It is a global network of billions of computers through which one can share information, communicate and more.

A thing in the context of IoT can simply be any object. It can be a chair with a pressure sensor, a smart fridge, a tree in a farm, an animal with a transponder, an automobile that has a built-in sensors to sense crashes and alert the emergency services, a bio-chemical or implantable medical sensors, etc...

Kevin Ashton, cofounder and executive director of the Auto-ID Center at MIT, first mentioned the Internet of Things in a presentation he made to Procter & Gamble in 1999. Here is how Ashton explains the potential of the Internet of Things: “Today computers - and, therefore, the Internet - are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1,024 terabytes) of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture or scanning a bar code.

The problem is, people have limited time, attention and accuracy – all of which means they are not very good at capturing data about things in the real world. If we had computers that knew everything there was to know about things – using data they gathered without any help from us – we would be able to track and count everything and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling and whether they were fresh or past their best.” In the following Figure 2.1, we can see the evolution of technology related to objects: we left the RFID

which was adapted for counting and tracking objects to move towards communication and control objects, of course with the addition of the connectivity and embedded intelligence.

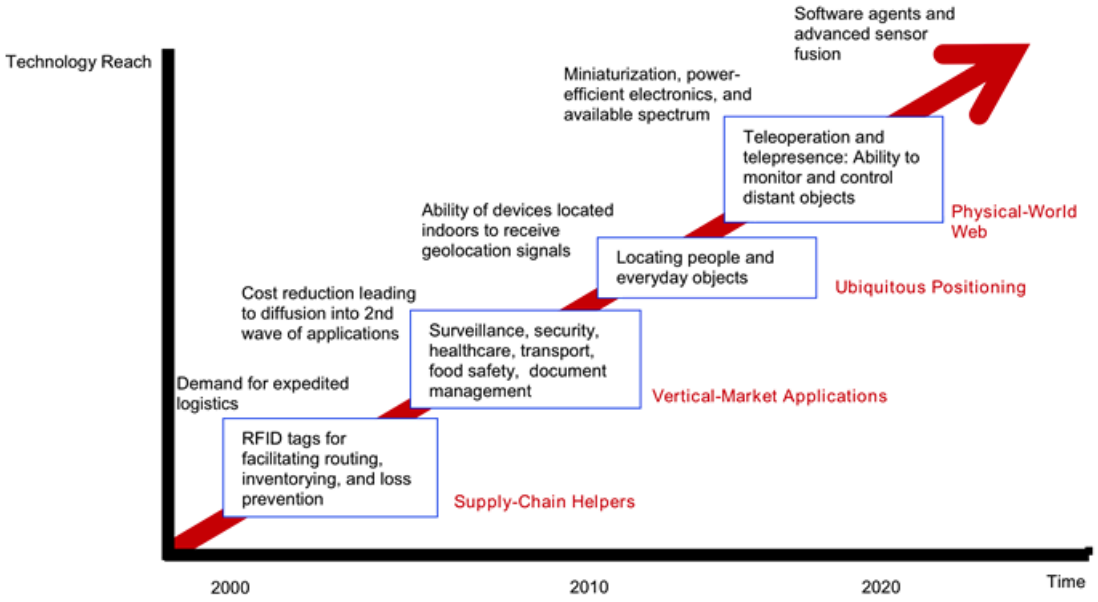


FIGURE 2.1: Development and Outlook of object related technology [4]

For Margaret Rouse [19], "The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction." The Figure 2.2, presents another view of what IoT is. It can be simply view as a network where anything or anyone from anywhere can be connected at anytime and share any information.

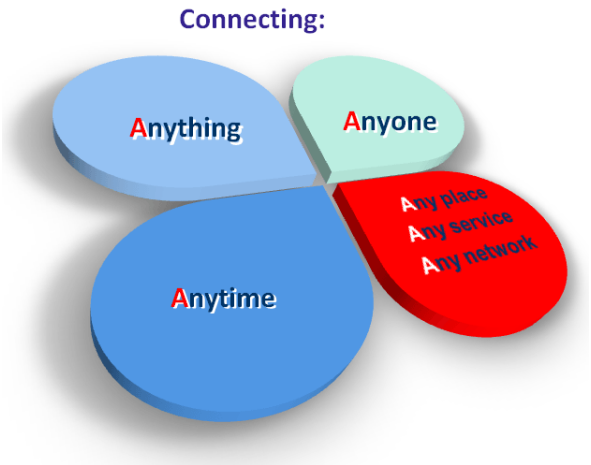


FIGURE 2.2: A view of what IoT is. [23]

A more physically, the Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

2.1.2 Fields of Application

As it is now known, any object can be a part of an IoT network, it is obvious that the fields of application are unlimited since one can turn any object into a smart one. The diversity of applications will affect almost all sectors including home and consumers, transport and mobility, health and safety, buildings and infrastructure, energy, etc...Figure 2.3 gives a view of some of the application fields of the IoT.

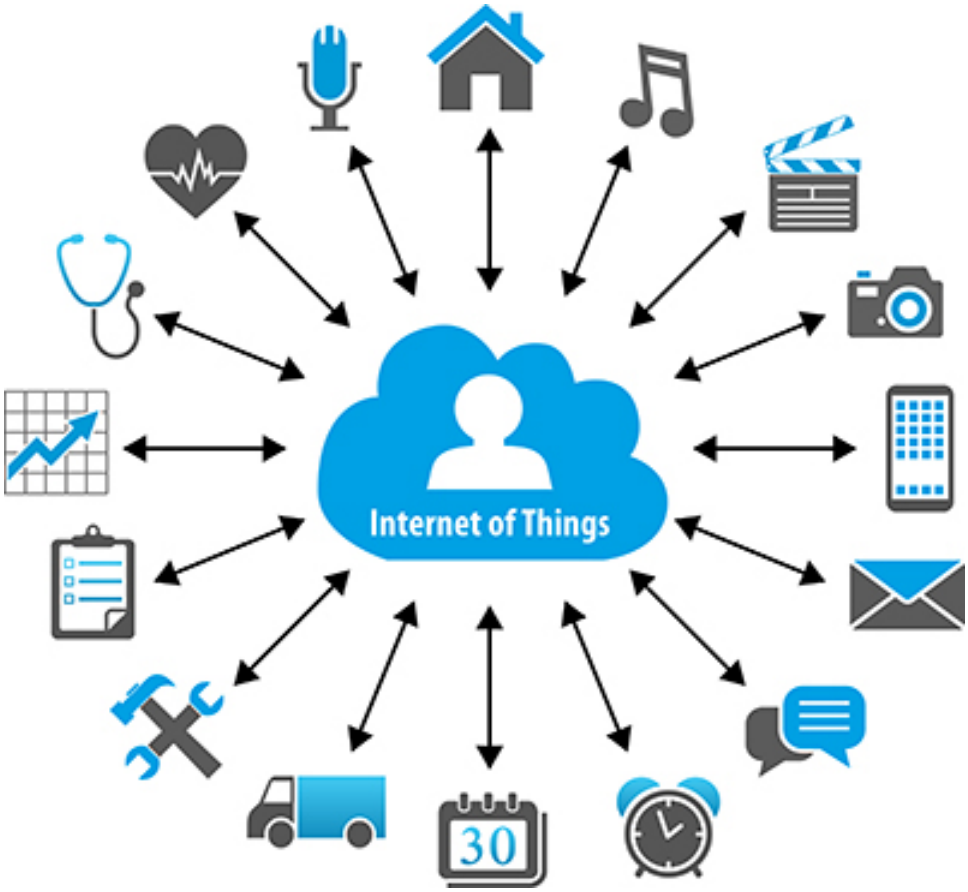


FIGURE 2.3: Several fields of application of IoT [5]

The future of IoT sector is promising and there is a growing interest of many large companies (automotive, energy , telecommunications, electronics, IT, biotechnology, etc...) that are diversifying by opening up this market segment. The economic weight and interest of some applications help to boost investment in research and development and

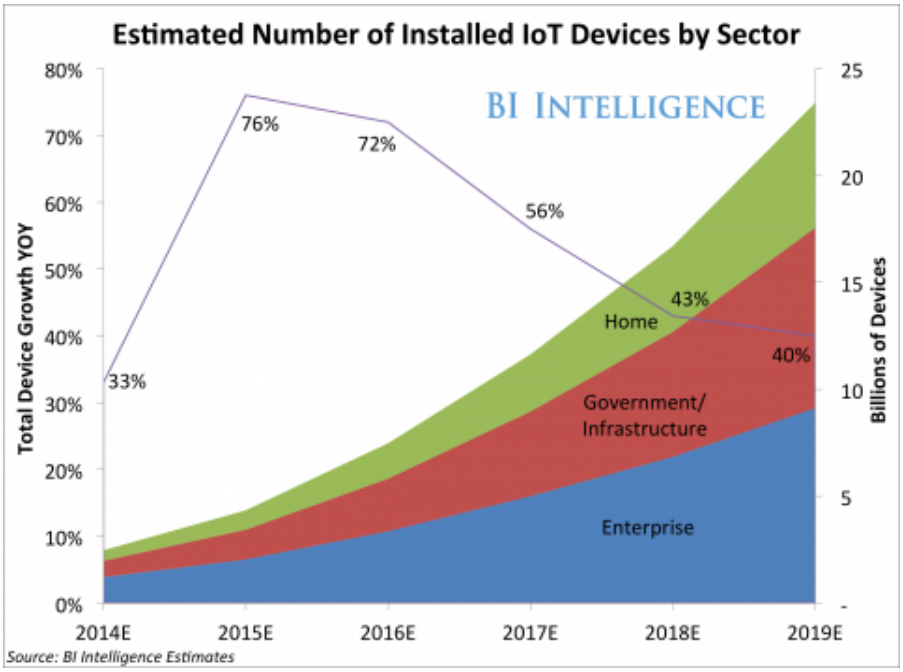


FIGURE 2.4: A view of the IoT market evolution. [10]

sustainably install the uses of the IoT. Figure 2.4 shows a projection of market trends in this sector.

One can observe a rapid change in the number of connected objects for years to come. This can be explained by highlighting the benefits (increase of efficiency, reduction of costs) of the connection of inert objects on the Internet. In one report in 2015, BI Intelligence predicted that:

- IoT will be the largest market for devices in the world with tens of billions of devices.
- As well as residential areas, business and government will be the main recipients of the products.

Factors that could slow down the infatuation in the sector concern the cost and security vulnerabilities. Indeed, the sector is still in the beginning of its development and all is not yet standardized. Grant and Al [11] show that on a particular brand of smart thermostat, it is possible for a hacker to use a smart-thermostat as an access point and bypass all security barriers. The hacker then has access to the entire infrastructure. The concern might be more pronounced because the manufacturer of this equipment is often presented as the global industry leader and even a leader in the field of IT.

2.2 IoT Architecture Layers

An in-depth look at the technologies accompanying the IoT allows to identify an architecture model that has 4 main layers.Figure 2.5 represents a structure and some technologies of a 4 layers architecture model.

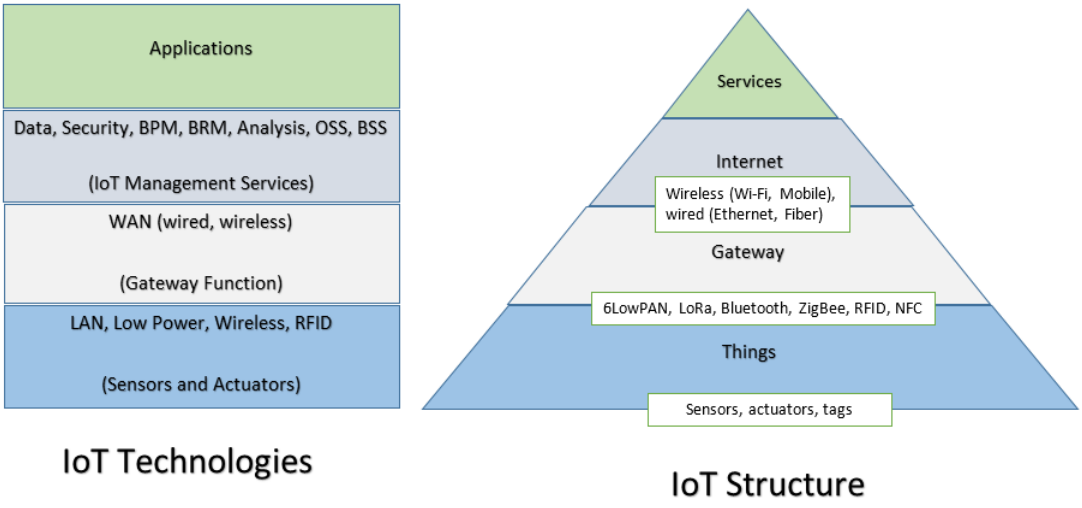


FIGURE 2.5: IoT architecture with 4 layers

From bottom to top:

- **Sensing and Identification Layer**: this layer includes sensors, actuators, tags(RFID, Barcode) The sensors are usually of low power, low data rate and real time.
- **Network Construction Layer**: it manages the local network and the connectivity. Some devices can be connected directly to the Internet without passing through a local gateway. The choice made here is efficient energy management oriented since the devices are generally low power.
- **Management Layer**: create and manage services (configuration, performances, data flow, security control, etc...) to satisfy users.
- **Application Layer**: provides interaction methods to users and applications.

2.3 State of the Art

2.3.1 The Radiator Thermostat

The heating system of residential buldings assesses for more than 2/3 of the energy consumption. In every day life, a few simple actions enable everyone to contribute to the reduction of its energy consumption. To reduce energy costs related to heating systems, it is advisable to adjust the heating on lower but comfortable temperatures. For example it is shown that heating a room at 19°C instead of 20°C can reduce energy consumption by nearly 7% while maintaining a good level of comfort. The thermostat is therefore the key element to achieve this goal.

Mechanical Valve

The main function of the radiator manual valve is to open or close the supply of hot water. It does not regulate the temperature therefore may lead to waste of energy. The temperature sensor in a mechanical thermostatic valve (Figure 2.6) is made up of two pieces of metal that are laminated together (bi-metal). Each type of metal has a different rate of expansion when heated and cooled, which is what controls the thermostat temperature. This regulation of the temperature allows substantial savings. For example in case of improved weather conditions, the valve will reduce the amount of water flowing through the radiator, reducing the amount of energy consumed .

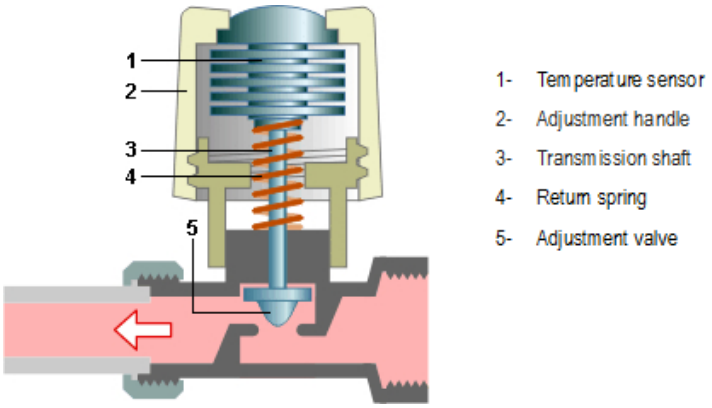


FIGURE 2.6: schematic of a radiator valve [8]

The Electronic Thermostat

Most electronic thermostats, has features that allow them to better operate than mechanical valves. These features include:

- a better regulation, because electronic temperature sensor are more accurate than the bi-metal used in mechanical thermostat;
- a digital display that is a better HMI (human-machine interface);
- an electronic temperature sensor whose output can be directly used for enslavement;
- a program mode that enables the user to predefine the operations over a long time period.

Most of the time, electronic thermostats use as actuator DC motors to move the transmission shaft. Those motor are often associated with a gearbox to reduce the rotation speed and manage the transmission shaft of the valve easily.

One of the main disadvantages of electronic thermostats is that they require a source of electrical power (usually batteries which level should be monitored and replaced when it is necessary), which is not the case of mechanical thermostats.

2.3.2 Smart-Thermostats market

The future of the connected thermostats market is bright because users are increasingly sensitized on the gain from the use of this technology. It is clear that thanks to the energy saving potential of up to 30% announced by most manufacturers, the target audience will respond in favor. Another very convincing argument is the return on investment because the potential energy gain and money saving allow in a few years (usually 1 or 2 years) to fully amortize the equipment. It is then understandable that the enormous economic potential of this sector leads large multinationals to invest in the sector. Figure 2.7 shows a classification of some smart thermostat leading manufacturers and their position in the market.

This classification done by Navigant Research [18] examines the strategy and execution of several smart thermostat manufacturers and software providers that are active in the global smart thermostat market and rates them with an objective assessment of these companies relative strengths and weaknesses in the global smart thermostat market.

2.3.3 Examples of Connected Thermostats

On the market, there is a multitude of connected thermostats which mostly uses the electric grid to power the Internet gateway. Below are given two examples, one of which is a product of a leader and one that also meets success.

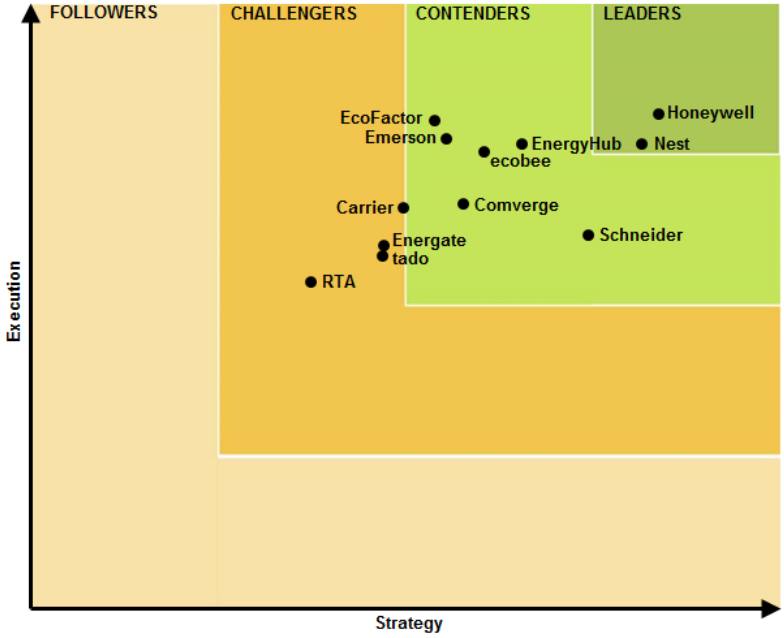


FIGURE 2.7: Main manufacturers and market positions [18]

eQ3-MAX!

The eQ3-MAX!(Figure 2.8) is a package consisting of a MAX!Cube which is the central gateway for the local network, thermostatic valves and a wall thermostat.

FIGURE 2.8: eQ3-MAX! [1]

It allows the control of the heating system via the Internet. It is able to control up to 50 devices in 10 rooms. This set allows to control the central heating system through a personal computer or a smart-phone. One of the main positive points of the EQ3-MAX!

is that it is easy to install and configure through its own software. It works based on a schedule programing and can be operated both manually and remotely. Its sensors that detect opening windows inform the user and can adapt the heating operation. The ECO switch reduces the temperature of all parts of a building upon exit and restore last temperatures when it is reactivate.

The Honeywell Evohome THR992GRT

The Honeywell Evohome THR992GRT pack (Figure 2.9) consists of:

- a central programmable thermostat which is the core of the system
- a RFG100 gateway to control the system via a smart-phone
- one radiator valve

FIGURE 2.9: The Honeywell evohome THR992GRT [12]

The Evohome allows to independently adjust the temperature in different rooms (up to 12) with the possibility to preset in advance various periods schedules. it is not only used to control the heat in a room but it can also control the production of hot water for sanitary. Like other smart thermostats, Evohome learn how long it takes to heat a room to define the optimal periods of start-up and shutdown. By using the app IFTTT (If-This-Than-That), it can turn the heaters when a user leave a postal code.

Chapter 3

Hardware Design

The previous chapter set the foundation of a connected thermostat project. This chapter will focus on the design of the hardware part of the project. To carry out an IoT project, it is always necessary to define in advance some parameters to build a prototype and estimate costs. Among those parameters, the most relevant are:

- the type of sensors/actuators
- the type of communication interface(s)
- amount of data to be captured and transmitted
- frequency of the data transmission

The methodology is to select the components and modules based on predefined selection criteria. This chapter is organized into seven sections. The first presents an overview of the system. The next two are respectively interested in the micro-controller unit (MCU) and the radio module which together form the core of the project. Sections 4 and 5 deal with Inputs/Outputs. The last two parts are interested in the power supply and design of the PCB.

3.1 Block Diagriam

The electronic thermostat valve regulates the temperature by adequately controlling a transmission shaft. Its HMI allows the user to manipulate it more easily. To perform its functions, the thermostatic valve must have the following modules:

- a MCU to manage all system functions.
- connectivity module which is the interface between the electronic board and the LAN.
- sensors(temperature, battery level).

- an actuator (DC-motor)
- a LCD Display to display information relevant to the user.
- an In-Circuit Serial Programming to program the MCU without having to remove it from the board.
- a power supply to supply all the parts of the module.

Figure 3.1 shows the block diagram of the hardware part of the project.

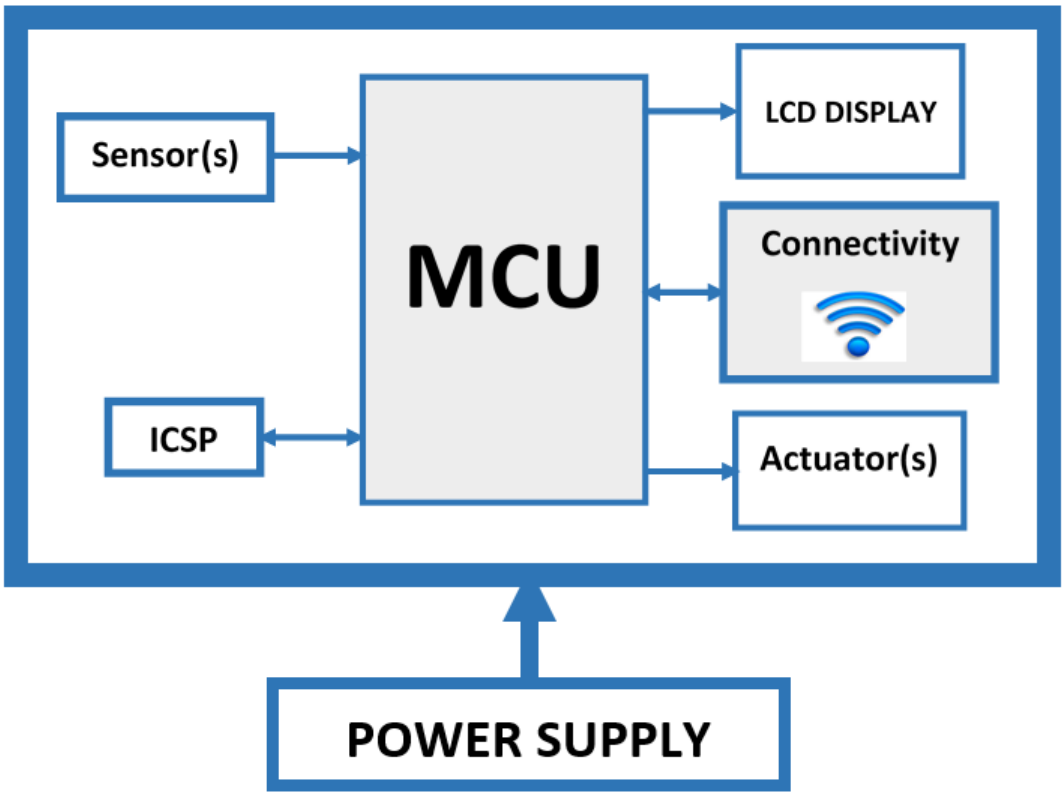


FIGURE 3.1: Block diagram of the hardware

The next sections aim to choose the components of the various modules of the block diagram.

3.2 The Microcontroller Unit

A MCU is a self-contained system with peripherals, memory and a processor that can be used as an embedded system. All IoT devices have a MCU to manage the main functions. For recall, an IoT device can be seen as a classical embedded system based on a MCU with added connectivity.

MCU manufacturers in their catalogs usually have many different product ranges. The choice of the most suitable MCU for a particular application is mainly based on technical specifications provided by manufacturers in the device datasheets. The main criteria used in the selection are:

- required hardware interfaces
- the software architecture
- the architecture
- the memory needed
- processing demand
- power constraints
- availability of development tools (hardware and software)
- speed

When it comes to design IoT devices, particular emphasis must be put on the power and the memory needed.

Most well-known MCU manufacturers because of the enormous potential that represents the progress of the IoT are opened to this market segment. The logical consequence is the entrance into their catalogs of many oriented IoT devices among which low and extra low power MCU. Indeed, most of those devices are battery-powered and must work as long as possible without requiring a battery replacement.

It is not good enough to only look at main current consumption specifications while selecting a low-power MCU. One should not only look for the MCU which has the lowest power but should also look under what circumstances that occurs. Parameters that can help to better assess the current consumption of a low-power device are:

- supply voltage range
- disabled peripheral(s)
- functioning mode (active, idle, sleep, deep sleep)
- operating speed
- state retention
- wake-up time
- wake-up sources

TABLE 3.1: Low Power Family examples from 3 manufacturers

	STMicroelectronics Ultra-low-power STM8L	**Microchip eXtreme Low Power**	**Microchip Atmel picoPower Technology**
Supply Voltage Range (V)	1.65 to 3.6	1.8 to 5.5	1.8 to 5.5
Max Flash Size(kB)	64	256	256
Internal RAM Size (kB)	1 to 4	0.064 to 16384	
Functionning Features	Lowest power mode: 0.30 μA Dynamic run mode: 180 $\mu A/MHz$	sleep currents down to 9 nA run currents down to 30 $\mu A/MHz$	$<35\mu A/MHz$
Data EEPROM	256 to 2048 bytes	256 to 1024	4096
Hardware Interfaces	ADC DAC USART, SPI, I2C RTC Timers Temperature sensor Comparators etc...	ADC DAC USART, SPI, I2C, USB, PWM OPAMP RTCC Timers Comparators etc...	ADC DAC USART, SPI, I2 C, PWM OPAMP RTCC Timers Comparators etc...
Max Operating Speed (MHz)	16	16	64

Table 3.1 presents some low-power MCU familly from three main manufacturers while Table 3.2 compares some features of three MCU from these three manufacturers.

When working on battery-powered applications, one of the most important issue to be taken into account is the battery life. Regular replacement of the battery is not wanted thus its life needs to be as long as possible. This also presents an economic and environmental advantages. As in most IoT applications, sensors repetitively collect data at regular time intervals. It is no longer necessary to use the microprocessor all the time. The power consumed by a MCU is the sum of a constant part (modules such as regulators) independent of the frequency and a portion which varies with the frequency.

TABLE 3.2: MCU Power Management comparison

	PIC18LF25K50	**ATMEGA168P**	**STM8L151F1**
Flash(kB)	32	32	8
RAM (kB)	2	2	1.5
Pins	28	32	up to 48
Sleep (nA)	20	100	800
Deep Sleep (nA)	-	100	300
WDT (nA)	300	3500	450
$\mu A/MHz$	110	300	150
Wakeup (μs)	5	-	5

To reduce power consumption, the operating frequency should be reduce to the lowest possible value that is sufficient for the proper functioning of the application. The use of the sleep mode also helps to significantly reduce consumption. As shown in Figure 3.2 the processor is used only a short time (active mode) to execute the code and then disabled with all non necessary modules (sleep or deep sleep mode). The time taken to go from sleep mode to active mode is an important feature because in this period energy is lost: why the wake-up time is one of the criteria for selecting low-power MCU.

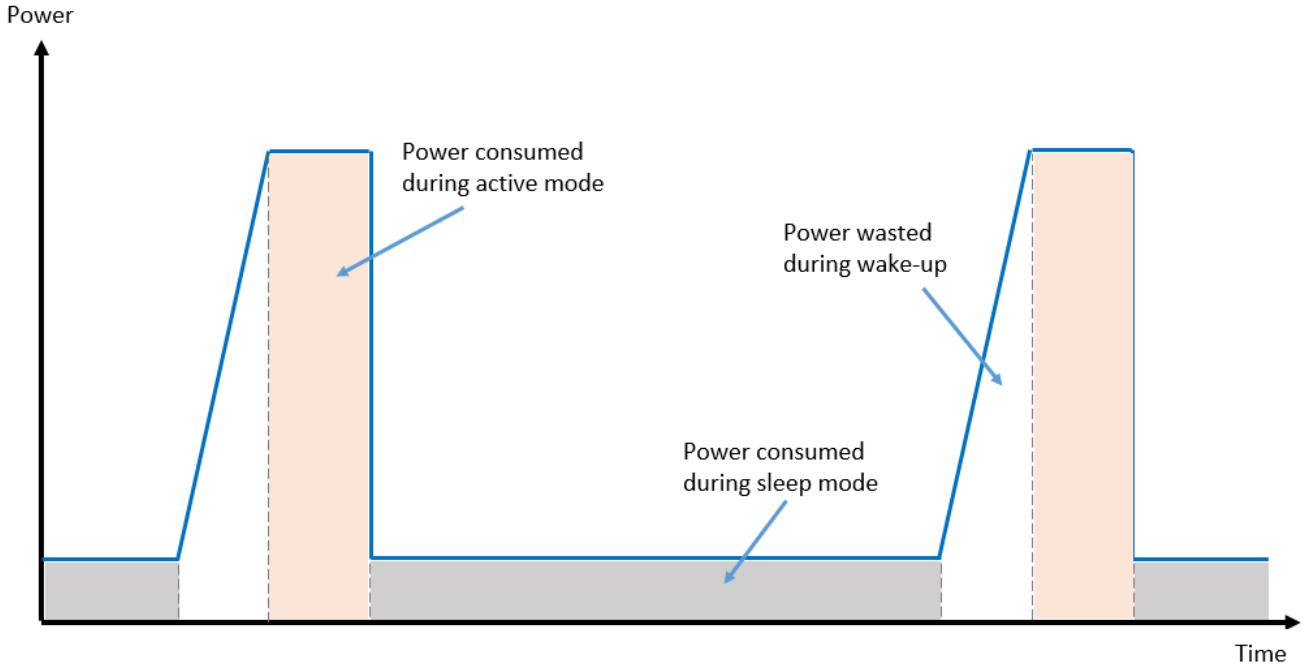


FIGURE 3.2: Active and Sleep mode of the CPU

Taking into account the selection criteria listed previously and the availability of development tools (software, hardware, free MCU samples), the choice goes to the PIC18LF45K50 whose family block diagram is shown on Figure 3.3. The results of this work will be based

on this MCU whose main features are presented in AppendixA. The principles will be the same for the other MCUs.

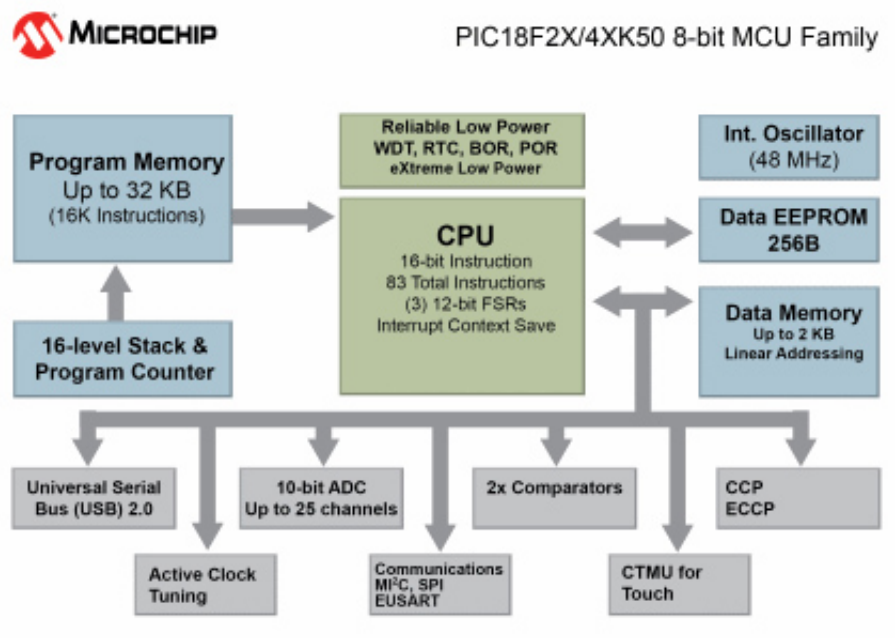


FIGURE 3.3: The PIC18LF45K50 family block diagram [17]

3.3 The Connectivity Module

The way an IoT object sends or receives data is a key aspect of the design of IoT devices. There are many wireless connectivity technologies used in the IoT field (WiFi, LoRa, Bluetooth, Infrared, Zigbee, LTE, NFC, BLE, Z-Wave, etc ...). To choose one specific technology, some features should be taken in account among which:

- the range of the network
- the maximum data rate, to be sure to have enough bandwidth
- the frequency
- the available security protocols
- authentication

3.3.1 Choice of the Module

Connectivity is one of the intrinsic characteristics of IoT devices. The choice of a connectivity (or transceiver) module must be done with special attention because with the

MCU, it is one of the core elements of the device. It provides connectivity between the unit and the rest of the network. It should be noted that it is usually the largest source of energy consumption in IoT devices. Some wireless communication integrated MCUs often incorporate architectures to reduce power consumption but are not always suitable for small projects.

In addition to the particular weight given to the electric characteristics, the criteria for choosing a connectivity module, highlight other relevant characteristics including:

- the cost
- interface(s)
- protocol(s)
- sizes
- modes
- etc...

Table 3.3 makes a comparison of WIFI modules from some main manufacturers on the consumer market. Since the main idea of the project is to add connectivity to existing systems, criteria such as cost, electrical, interfacing, sizes, number of pins, leads us to choose the ESP-01 module.

Table 3.3: WiFi modules comparison

	Wifi Shield	Wifi Shield	Huzzah CC3000	ESP-01	ESP-12	Huzzah ESP8266	RN131	SPWF01SA	Smart Connect
Company	**Sparkfun**	**Arduino**	**Adafruit**	**Espressif**	**Espressif**	**Adafruit**	**Microchip**	**ST**	**Microchip Atmel**
Chip module	RN-131C	HDG204	CC3000	ESP8266	ESP8266	ESP8266	RN131	SPWF01SA	ATWINC1500
Wifi Standards	802.11 b/g	802.11 b/g	802.11 b/g	802.11 b/g/n	802.11 b/g/n	802.11 b/g/n	802.11 b/g	802.11 b/g/n	802.11 b/g/n
Packets	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP	TCP, UDP
Modes	Client/Server	Client/Server	Client/Server	Client/Server	Client/Server	Client/Server	Client/Server	Client/Server	-
Concurrent sockets	-	4	4	5	5	5	-	8	-
Size	66x53mm	66x53mm	27x41x3 mm	21x11 mm	24x16 mm	25x38 mm	37x20x4 mm	27x16x3 mm	34x15 mm
Interface	SPI	SPI	SPI	UART	UART	UART	UART	UART	SPI or UART
Encryption	WPA2-PSK	WPA2-PSK	WPA2-PSK	WPA2-PSK	WPA2-PSK	WPA2-PSK	AES128	AES256	AES
Supply Voltage	3.3V	5V	3.3V or 5V	3.3V	3.3V	3.3V	3.3V	3.3V	3 to 4.2 V
Sleep Current	4uA	-	-	10uA	10uA	10uA	4 uA	15 mA	4uA
TX Current	210 mA	210 mA	350 mA	215 mA	215 mA	215 mA	210 mA	243 mA	172 mA
RX Current	40 mA	-	-	60 mA	60 mA	60 mA	40 mA	105 mA	70
Number of pins	44	30	9	8	22	22	44	29	40
Prog MCU	No	Yes	No	Yes	Yes	Yes	No	No	No
Cost (EUR)	74.32	82.11	37.13	4.30	3.85	9	27.38	18.43	7.5

3.3.2 The ESP8266-01

General Overview [22]

Espressif Systems' Smart Connectivity Platform (ESCP) is a set of high performance, high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed WiFi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

Figure 3.4 presents a view of different modules of the ESP8266 family while Figure 3.5 shows the ESP8266-01, target module of the project and its pin-out. Recall that its size (roughly the size of 1 euro coin), cost (a few euros), its interface (series communications found on all MCUs) and the number of pins (only 8, which simplifies interfacing) have contributed to this choice. The main differences between the members of the ESP8266 family are:

- the memory sizes (Flash, RAM)
- the number of GPIOs
- the type of GPIO (digital, ADC, PWM)
- antenna type

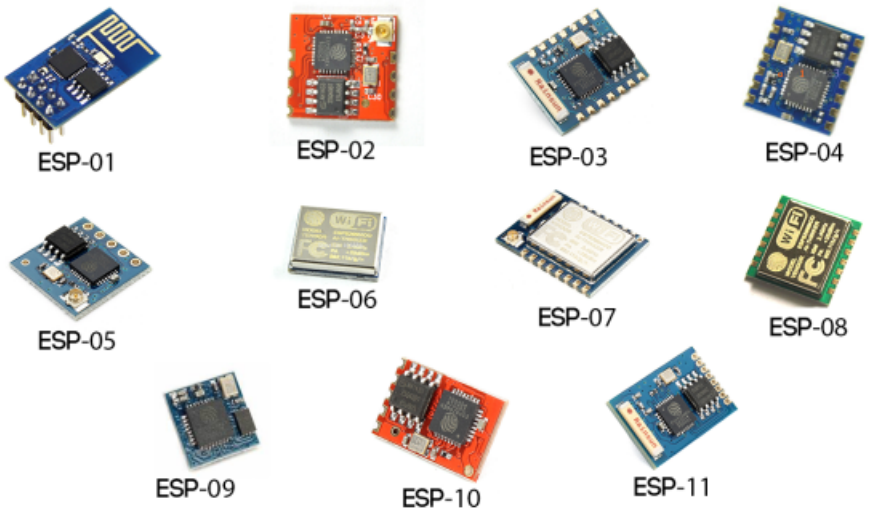


FIGURE 3.4: The ESP8266 family [20]

ESP8266EX offers a complete and self-contained WiFi networking solution; it can be used to host the application or to offload WiFi networking functions from another application processor. When ESP8266EX hosts the application, it boots up directly from

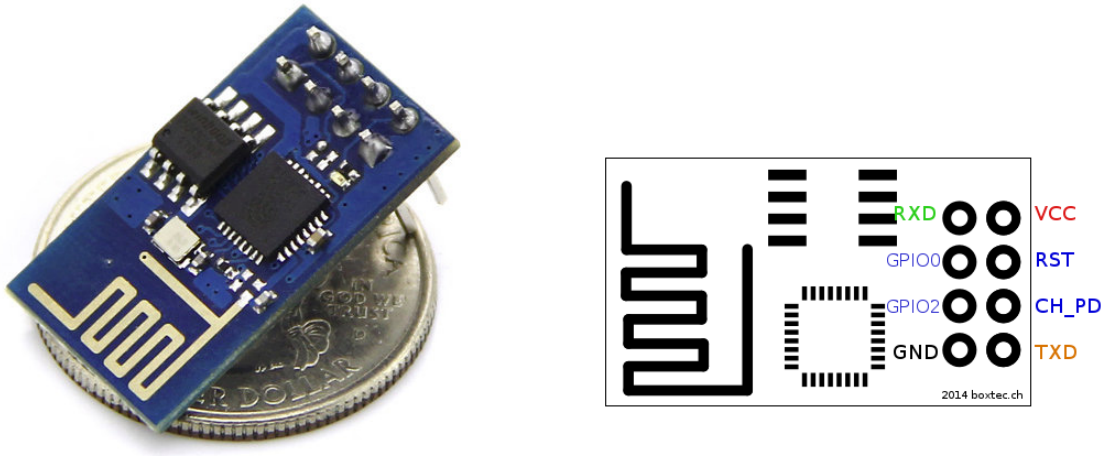


FIGURE 3.5: The ESP8266-01 module and pinout [14]

an external flash. It has integrated cache to improve the performances of the system in such applications. Alternately, serving as a WiFi adapter, wireless internet access can be added to any micro controllerbased design with simple connectivity (SPI/SDIO or I2C/UART interface). ESP8266EX is among the most integrated WiFi chips in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area. ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the WiFi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; sample codes for such applications are provided in the software development kit (SDK). Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

Specifications

The main technical features to be considered by designers using the ESP8266EX are the following:

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- 8 pins header includes 2 GPIO pins

- serial/UART baud rate: 115200 bps
- integrated TCP/IP protocol stack
- supply voltage: 3 to 3.6V (3.3V typical)
- I/O voltage tolerance: 3.6V Max
- regular operation current draw: $\sim 70mA$
- peak operating current draw: $\sim 300mA$
- power down leakage current: $< 10\mu A$
- +19.5dBm output in 802.11b mode
- flash Memory Size: 1MB (8Mbit)
- WiFi security modes: WPA, WPA2
- wake up and transmit packets in $<$ 2ms
- standby power consumption of $<$ 1.0mW (DTIM3)

Major Applications

Fields of application of ESP8266EX (alone or combined to another MCU) are numerous and include among others:

- Home Appliances
- Home Automation
- Smart Plug and lights
- Mesh Network
- Industrial Wireless Control
- Baby Monitors
- IP Cameras
- Sensor Networks
- Wearable Electronics
- WiFi Location-aware Devices
- Security ID Tags
- WiFi Position System Beacons

3.4 The Inputs

3.4.1 The Temperature Sensor

For an efficient operation, the system must be able to measure itself the temperature of the room where it is located. Taking into account the cost and the sizing specifications of the system, the choice fell on the LM335 whose pin-out and calibrated schematic are shown in Figure 3.6. The LM335 is a cheap, precise (1 ∘C initial accuracy available), easily-calibrated and integrated temperature sensor. It operates as a 2 terminal Zener over a 450uA to 5mA (which has a relatively low consumption) current range with virtually no change in performances. Its reverse breakdown voltage is proportional to absolute temperature at $10mV/\,^{\circ}\mathrm{K}$. When calibrated at 25C, the LM135 has typically less than 1C error over a 100C temperature range. Unlike other sensors, the LM135 has a linear output.

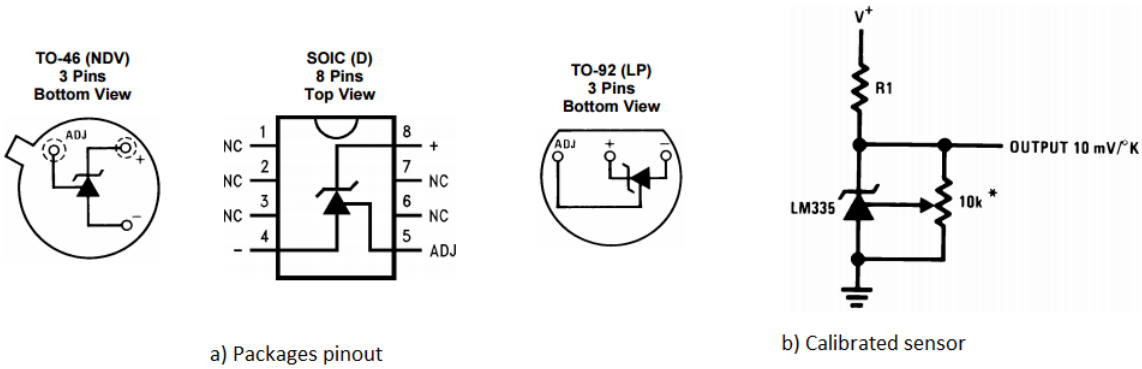


FIGURE 3.6: LM335 pin-out and calibrated schematic [13]

Its operating temperature range: T ∈[40,100] °C but in the project it can be assumes temperature range could be reduced to $[10, 40]\,^{\circ}\mathrm{C}$. The output of the sensor is directly connected to the MCU. Since this output is an analog voltage, one ADC channel of the MCU will be used to convert the value into a digital one. By looking into the datasheet there are little computations to do in order to get the correct value of the temperature. Since the ESP8266 is powered at 3.3V and the MCU can use the same voltage, the LM335 will be powered by the same supply. According to calibrated schematic in Figure 3.6, R_1 will be chosen to be nearly 1mA or less to reduce self-heating error.

At $25\,^{\circ}\mathrm{C}$, $V_{out} = 2.98V$. If it is assumed that the temperature range is $[10, 40]\,^{\circ}\mathrm{C}$, it corresponds to $[2.83, 3.13]V$.

For a fixed value $R_1 = 390\Omega$, imply I_{bias} ∈[1.2mA,435μA].

3.4.2 The Keyboard

One of the important elements of the HMI is the keyboard. It enables the user to give orders directly to the system without having to go through the application. Thus the user can change the temperature, define an operating program and more. The keyboard consists of push buttons. One problem encountered with push buttons is the phenomenon of rebound that bring the designers to associate a debounced circuit. Figure 3.7 shows a push button and its debounced circuit. To reduce costs and the number of components, the debounce will be done in the software. To reduce the energy, pull-down resistors are used (Figure **??**). To achieve a good interfacing with the user, the board needs several push-buttons:

- **UP** to increase the temperature value wanted
- **DOWN** to decrease the temperature value wanted
- **OK** to validate an order
- **ECO** to activate or disable the economic mode

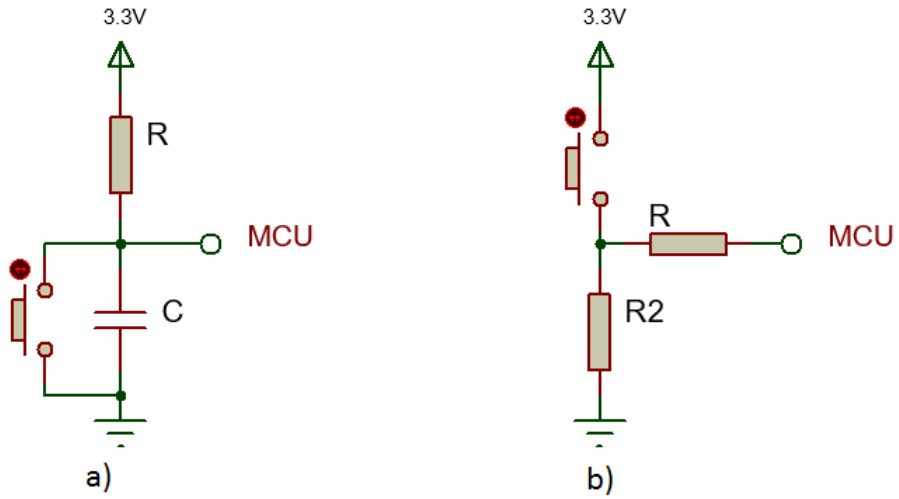


FIGURE 3.7: a) Debounced circuit b) push-button with pull-down resistor

3.4.3 The Battery Level Sensor

It is always important for battery powered devices to inform the user about the level of autonomy so that he replaces it at the right time. A simple estimation of the battery lifetime is given by the following equation:

$$BatteryLife = \frac{Battery\_Capacity\_in\_mAh}{Load\_Current\_in\_mAh}$$

Rare are the applications that use a constant current over time which means relying on the equation is not right and therefore the level of the battery must be regularly measure. Determine the exact remaining capacity of a battery is a complex exercise. Among the parameters that determine the remaining capacity of a battery are included the voltage, current, temperature, aging battery,...etc. To measure it, some critical applications typically use special ICs dedicated to battery management that will continuously measure the consumed current, voltage and temperature to allow these calculations. Figures 3.8 and 3.9 present the MN1203 typical discharge characteristics based on the voltage, the temperature, the discharges current and resistance. One important information that can be noted is that the battery has a 0.8V cut-off voltage. From this value there is an almost vertical slope and the battery can not power the unit. The Figure 3.10 shows various typical discharge curves of the ID1203 (a Duracell industrial 4.5V battery)

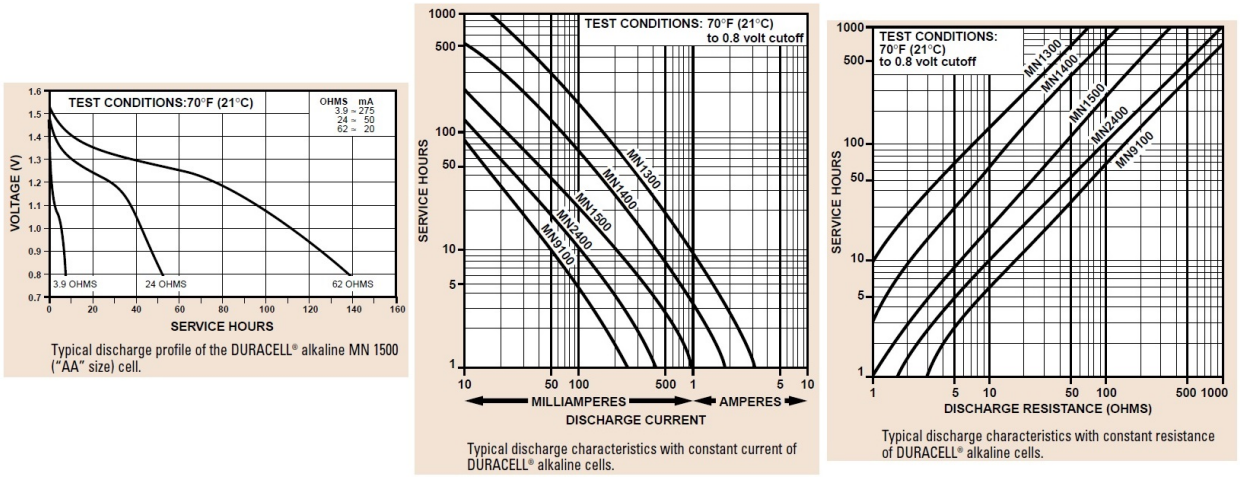


FIGURE 3.8: Duracell MN1203 typical discharge characteristics [6]

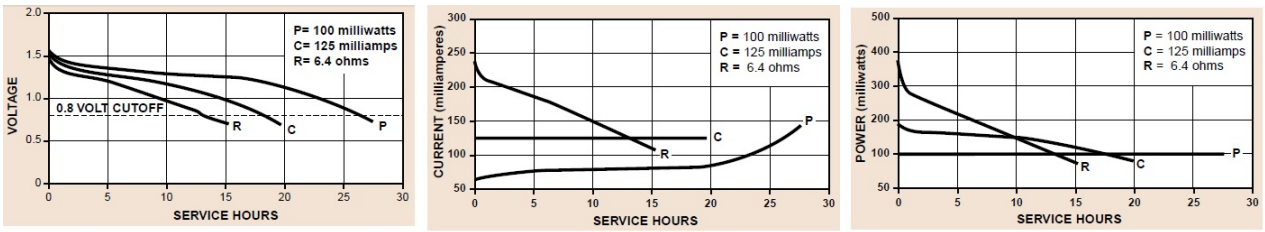


FIGURE 3.9: Duracell MN1203 typical discharge characteristics [6]

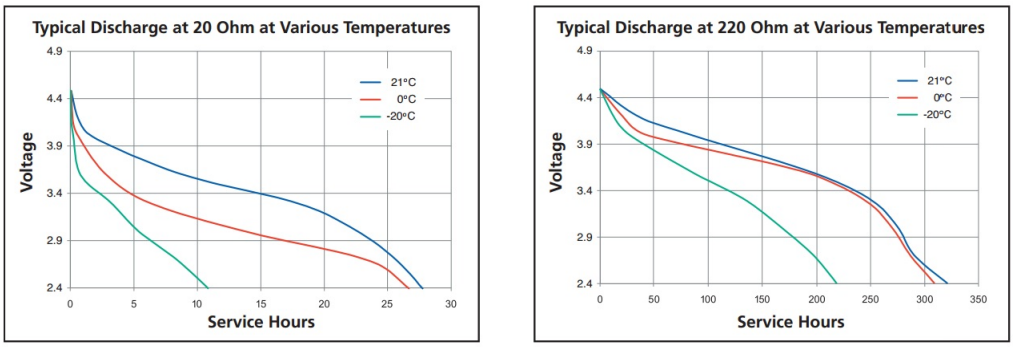


FIGURE 3.10: Duracell MN1203 typical discharge characteristics [7]

Some devices rely only on a voltage measurement to estimate the remaining capacity. As the battery of interest are from the alkaline type and the application is not critical, a simple estimation can be made using a voltmeter. It is known that the voltage goes down over time and usage. Most of the 4.5V alkaline batteries have their cut-off voltage between 2.4V and 2.8V. Although the MCU can be powered between 1.8V and 3.6V, the ESP8266-01 needs at least 3.0V. Therefore, the system will alert the user when the battery voltage drops to 3.3V.

The battery voltage is measured through an ADC channel of the MCU. Since the value of the battery voltage is higher than the MCU supply which serves as a reference, a voltage divider is used. We have :

$$V_{in} = BatteryVoltage * \frac{R_1}{R_1 + R_2}$$

$R_1 + R_2$ is chosen high enough (1 $M\Omega$ is good) to reduce the current leakage due to the voltage divider.

3.5 The Outputs

3.5.1 The UART

The Universal Asynchronous Receiver/Transmitter (UART) is a very popular serial communication interface which provides Half or Full Duplex communication between two devices. UART uses two data lines for sending (TX) and receiving (RX) data and share a common Ground/Reference. As the name indicates it is an asynchronous communication interface, which means that it does not need to send CLOCK along with data as in synchronous communications. The ESP8266 and the MCU will exchange information

through their UART modules. The Figure 3.11 shows how to connect two devices via the UART. Note that the RX of a device is connected to the TX of the device with which it exchanges information.

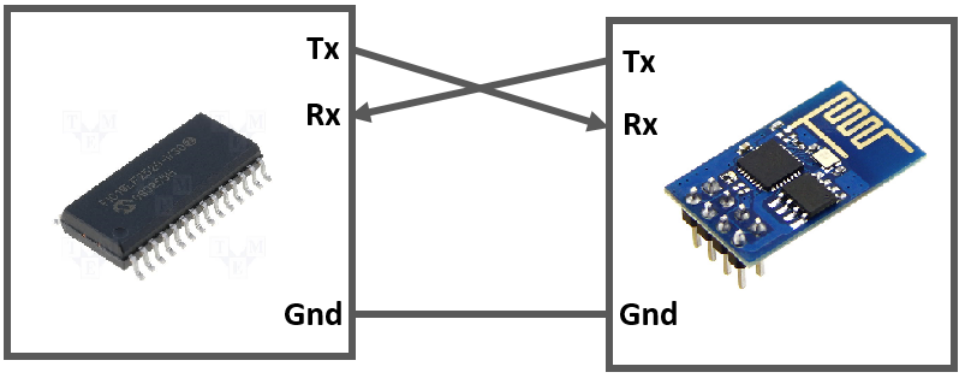


FIGURE 3.11: UART connection between the MCU and the ESP8266

By default, the firmware version of new ESP8266-01 ensure the communication speed is set to 115200 baud. Later or modified versions set the baud rate to 9600 bauds. These communication speeds are supported by PIC MCU UARTs.

3.5.2 The LCD Display

The LCD is the most important element of the HMI. It is through it that the system transmits to the user the status information and acknowledges orders from the user. For the designer it is very useful for debugging purposes. As other system components, its electrical characteristics are part of the most relevant criteria since the system power sypply is limited.

One of the most suitable LCD diplays for the system is the DOGS102W-6 (Figure 3.12) from the DOGS GRAPHIC SERIES of ELECTRONIC ASSEMBLY. It is a 102*64 LCD Graphic Display designed for compact hand-held devices and particularly it is suitable for low-power application. It is powered by a voltage between 2.5V and 3.3V and consumes a current of 250 μA typically. Its SPI and I^2C interfaces reduces the number of pins of the MCU used for interfacing.

Instead of the DOGS102, the system will be connected to the MC21605H6W-SPR3 which is a 2*16 Alphanumeric LCD display with a typical supply voltage of $3V$ and supply current of $2mA$. It can operate in two different modes:

- an eight bits mode which means every line of the data bus (DB0-DB7) of the LCD are used to send data to the LCD

FIGURE 3.12: DOGS102 LCD Graphic Display [3]

- a four bits mode meaning that just four lines of the bus are used to display a character.

Of course the latter is chosen to reduce the number of MCU pins and the consumption.

3.5.3 The DC Motor Driver

The control of the thermostat adjustment valve is made with a DC motor associated with a gearbox. The DC motor has to be a low power one. DC-motors need currents more important than the maximum output a MCU can provide. To control the motor with a MCU, a four transistors H-bridge configuration is then used. Its main functions are:

- to supply the motor with higher current than what the MCU can afford
- to enable the control of the motor in the two directions (forward and reverse) by inverting the voltage

The H-bridge (pin assignment, truth table and application circuit are shown in Figure 3.13) takes its name from its structure similar to a "H". The diodes of the bridge are freewheeling diodes whose goal is to disconnect the motor power safely for transistors. If the power is brutally cut while the motor is running, an overshoot will appear (Lenz's law $V = |L\frac{di}{dt}|$) across the switches and could destroy them. Therefore the diode allows current to keep flowing .

The closure of two transistors diagonally while the two others are OFF runs the motor in one direction. To change the direction of the motor, simply reverse i.e. close those which were open and open the others.Two transistors of the same arm should never be closed simultaneously, otherwise there will be a short circuit of the power supply. The resistor values are chosen to make transistors operate in the saturation mode.

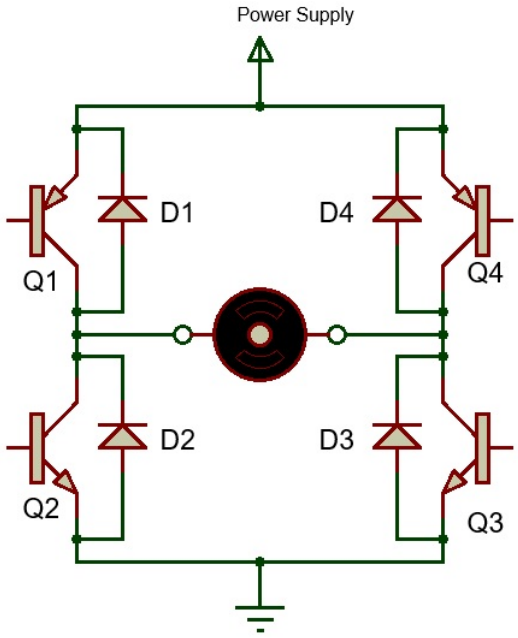


FIGURE 3.13: The DC-motor driver [21]

TABLE 3.4: H-bridge truth table

Operation	Q1	Q2	Q3	Q4
Forward	ON	OFF	ON	OFF
Reverse	OFF	ON	OFF	ON
Coast	OFF	OFF	OFF	OFF
Break	OFF	ON	ON	OFF

The following Table 3.4 specifies the motor operating type according to the state of the transistors.

In the project we will use the IC LB1938FA (whose pin assignment shown on Figure 3.14 which is an H-bridge motor driver that supports low-voltage drive and features low-saturation outputs in an ultra-miniature slim package.

3.6 The Power Supply

One of the most important features to take in consideration while designing battery-operated systems is reducing power consumption, which increases the battery discharge time and extends the operational life of a system. Taking into account the reduction of power consumption is even more important for IoT devices as they are equipped with

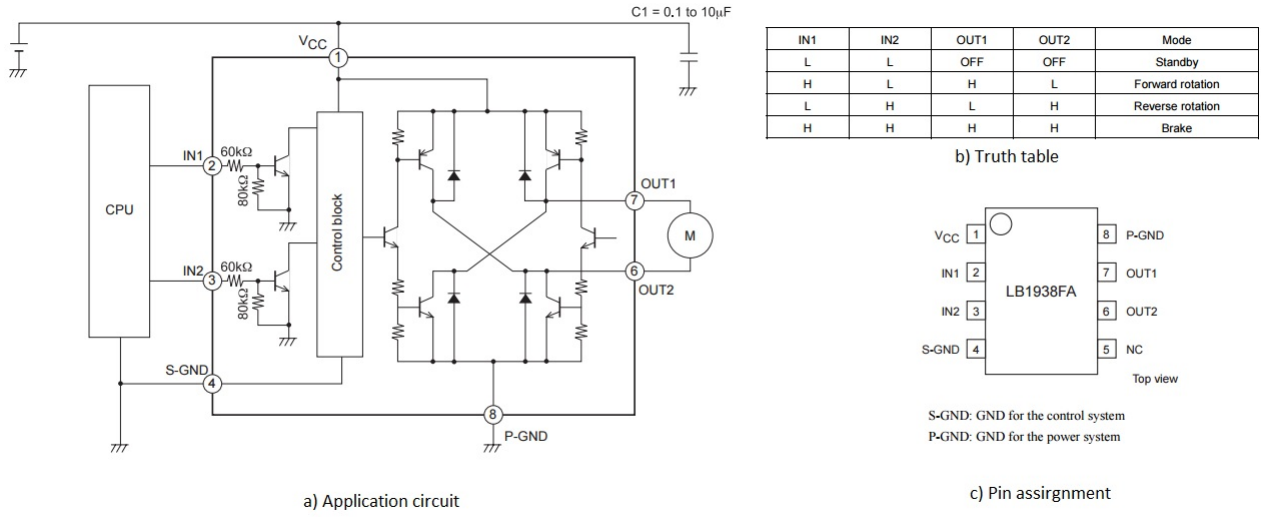


FIGURE 3.14: The H-bridge IC LB1938FA [21]

RF transceiver that require a significant amount of power to receive or transmit data packets. As the operating voltage of the ESP8266 is 3.3V and since it exchanges data with the MCU via their UART modules, it seems good to supply the entire system with 3.3V even if the MCU can drop to 1.8V. The Figure 3.15 represents the block diagram of the power supply module.

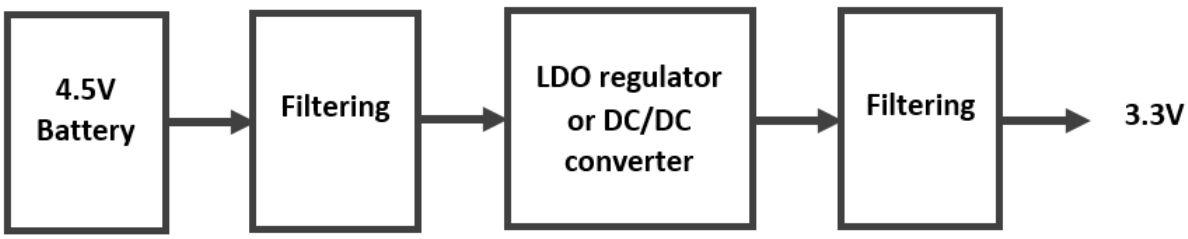


FIGURE 3.15: Power supply block diagram

In the design stage of a battery powered device, it is always preferable where possible to oversize the capacity of the battery to extend the operating time. Since the operating voltage is 3.3V, a 3.7V Lithium Polymer (LiPo) battery would do the job but cost constraints guide the choice to be a single MN1203 4.5V ($5400mA$) Alkaline battery or 3 AA cells in series. These batteries are cheap, safe to use, available everywhere and their capacity for AA size go from 1700 mA to more than 2800 mA. Some models are rechargeable.

A low dropout regulator is a linear voltage regulator DC which can regulate the output voltage even when the supply voltage is very close to the output voltage . The LDO

regulator main advantages are:

- absence of the switching noises (because no switching takes place)
- low ripple
- good noise rejection
- smaller size of the device
- And a greater simplicity of design (available in slim IC packages).

The linear voltage regulators unlike switching DC-DC converters dissipate a large part of the power limiting their efficiency around 50%. The efficiency is increase when LDO are used with an input voltage close to the output expected. The switching regulators with better yields (up to 85%) reduce the total current sourced from the battery and Extend battery life .

3.7 The PCB Design

The following figures represent respectively the global schematic (Figure 3.16), a view of the printed circuit design (Figure 3.17), a view of the printed circuit with and without the ground plane (Figure 3.18), a "3D view" pictures of the component side (Figure 3.19) and the solder side (Figure 3.20) of the board. The schematic design, simulations and the printed card was done using Proteus©. The card measures 5cm*5cm which is the order of magnitude of most thermostat cards.

The PTC (Positive Temperature Coefficient) added can be seen as an over-current protection device like fuses. Unlike a fuse, a PTC can self reset itself. When the threshold current is reached, the device begins to heat up, that increases its resistance, effectively choking the current back down to a safety level that can not damage components on the board. When the current is removed, the device cools off and is ready to go again!

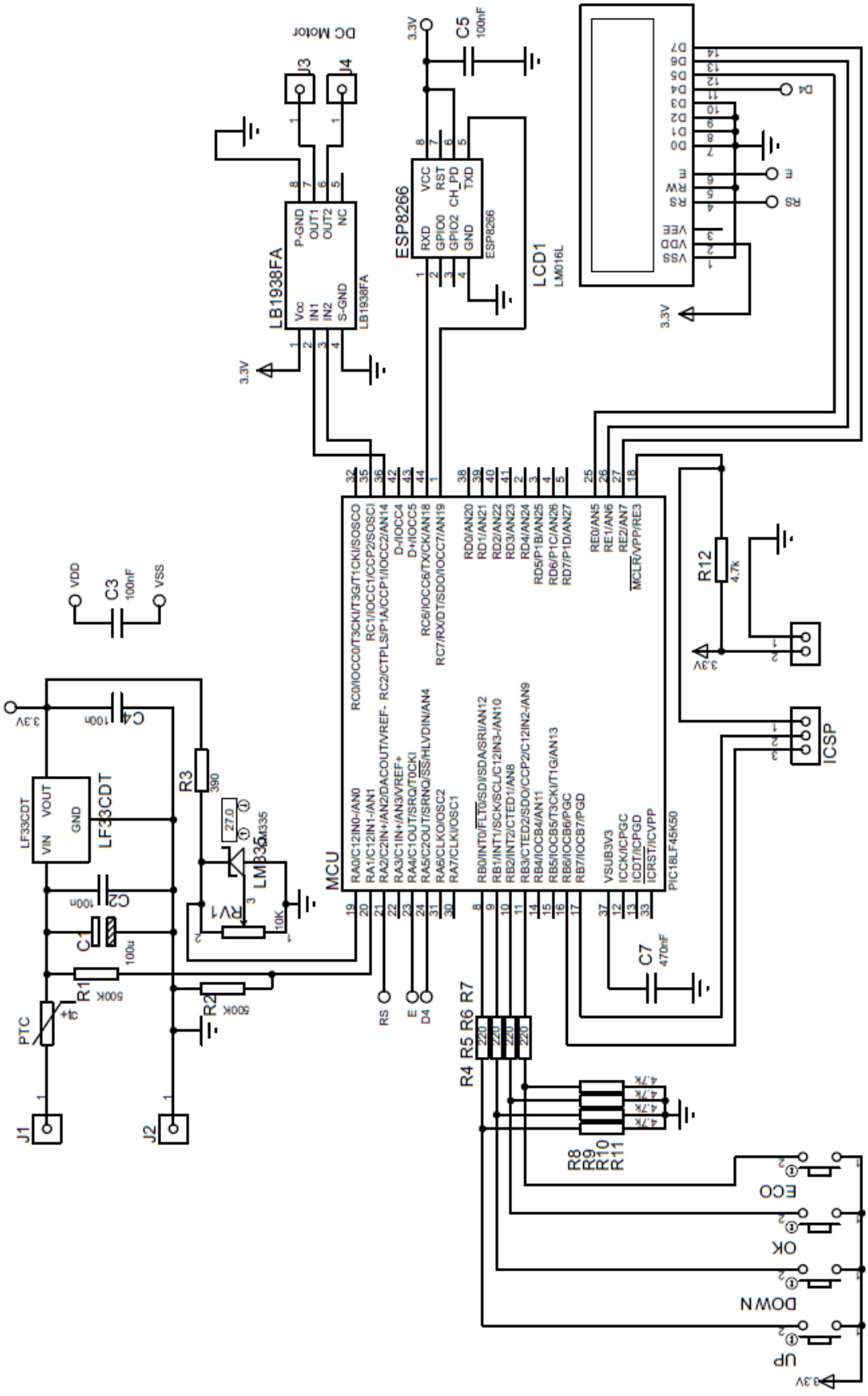


FIGURE 3.16: Electronic schematic

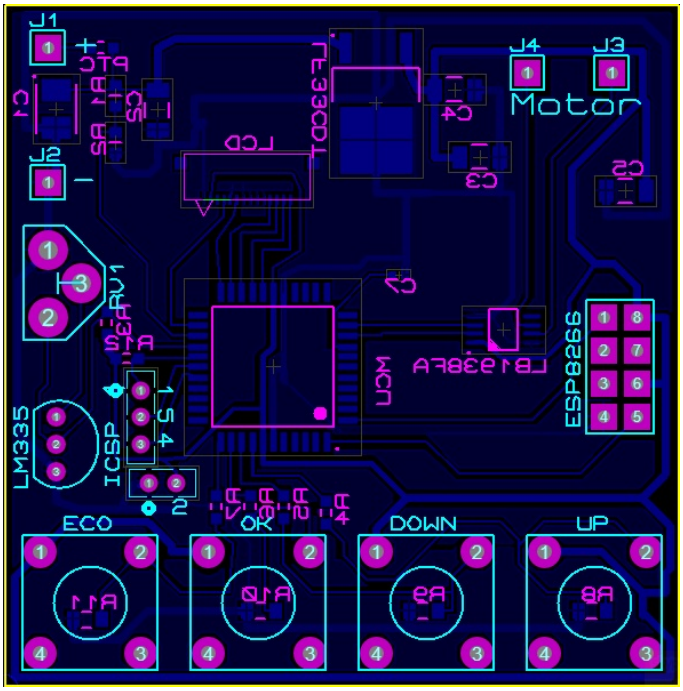


FIGURE 3.17: The printed circuit design

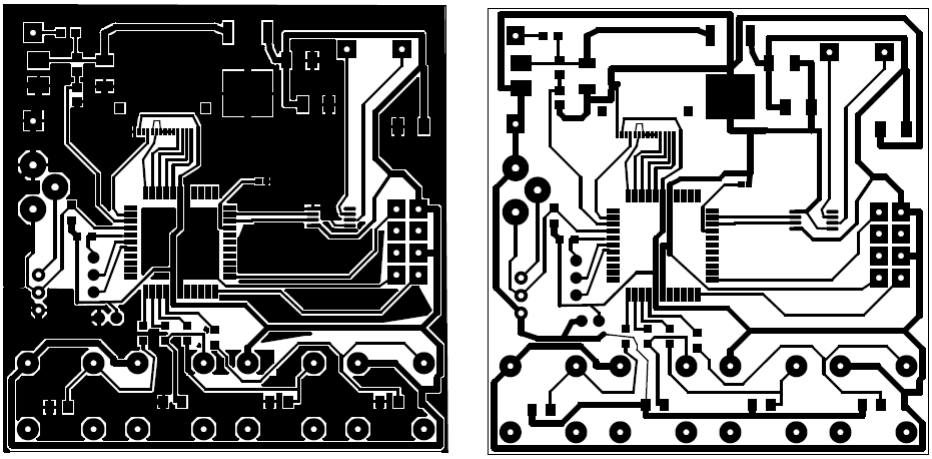


FIGURE 3.18: The printed circuit with and without the ground plane

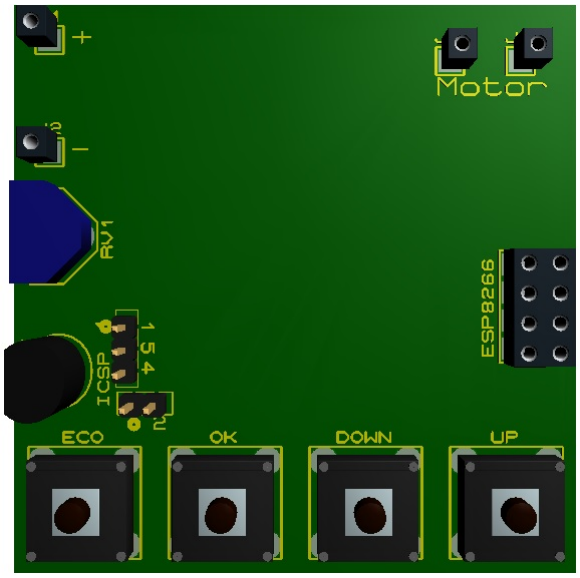


FIGURE 3.19: Component side

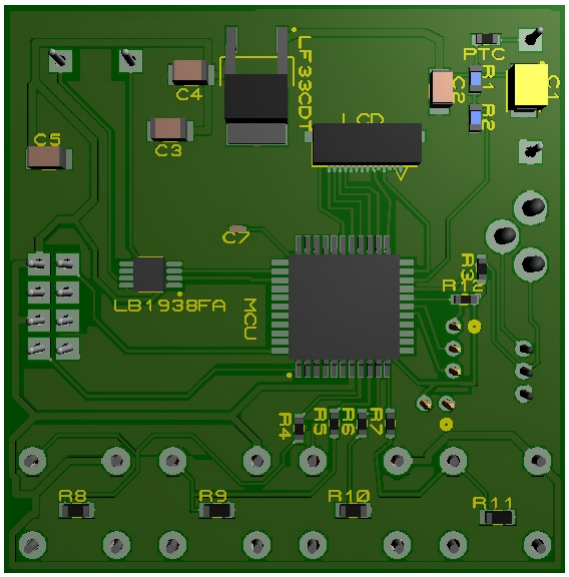


FIGURE 3.20: Solder side

Chapter 4

Software Design

In the previous chapter, the scope was to design a battery-powered electronic board for a radiator thermostat with a particular emphasis on improving the use of energy. This new chapter primarily deals with the software part of the project. To stay in the logic of the project, the software must be written in a style that can help to improve the lifetime of the battery. The structure of the hardware of the card directs the software. To achieve the scope, the chapter is divided in three parts:

The first part of this chapter describes the overall approach of the electronic board software. The second part presents the interface through which the user exchanges with the board through Internet . The last two parts respectively describe the software implemented in the MCU and the transceiver.

4.1 The Flow Chart

The electronic card is strutured around two chips which constitute the core of the card. Both chips are actually two MCU (one is associated with a transceiver) that operate independently and only exchange data through the UART port. Then each one will have its own code. The MCU is responsible for sending the current temperature to the ESP8266 module. The ESP8266 updates the value of temperature on the web page, reads commands and transmits them to the MCU that will control the valve position.

4.1.1 The MCU Flow Chart

The main functions of the MCU are:

- measure temperature and transmit it to the radio module
- monitor the battery level

- display useful information on the screen to users
- adjust the position of the thermostatic valve in order to issue the desired temperature
- manage both user's manual orders and orders received from the transceiver

The Figure 4.1 below represents the flowchart of the PIC18LF45K50. This diagram describes in a comprehensive manner the main stages of operation of the MCU.

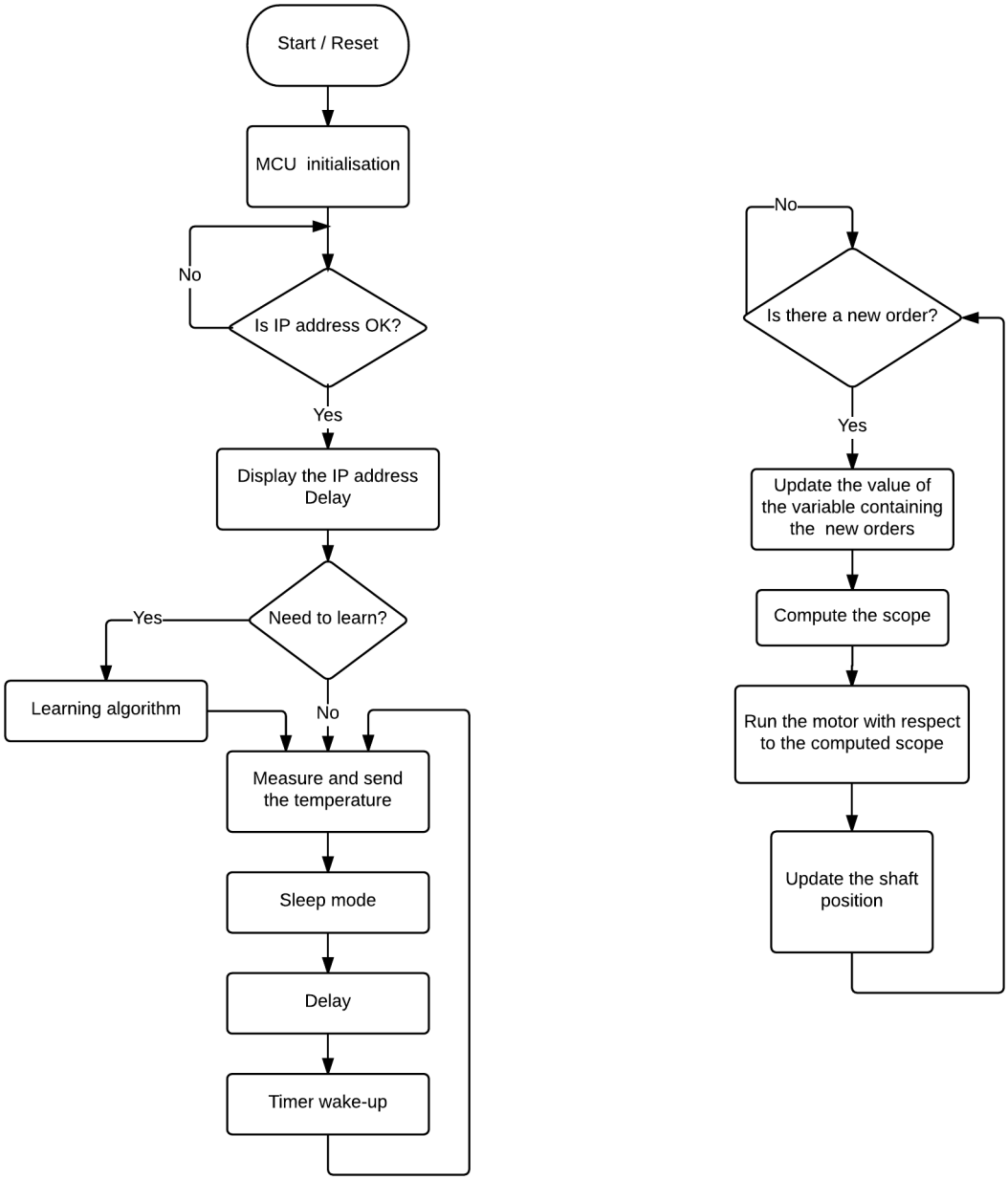


FIGURE 4.1: The MCU flow chart

When the electronic card is powered or the MCU is reset, the MCU activates and configures the necessary modules for operation of the thermostat. Since the thermostat is designed to be connected to any home WIFI network, it is not possible to program the connection credentials. It would be ideal for the thermostat to tell the user how to connect to the WIFI. In this project, the MCU must receive and display the IP address of the ESP8266. From this address, it can connect to a webpage and submit ESP8266's login data to connect the home WIFI. Once the module is connected to the WiFi network, the MCU offers the choice to the user to enable the implementation of a learning algorithm. This algorithm allows the system to apprehend the environment and then improve its functioning. In the next stage, the system enters an infinite loop. It performs measurements (temperature and battery level), updates the display, enters a sleep mode to reduce the energy it uses, after a delay wakes up to perform the next measurements.

The Pic18LF45K50 EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer independent of device program execution. This explains why the left and right parts of the flowchart are not connected. The MCU can receive data from the ESP8266 at any time. When it receives a new temperature target, it updates the related variables, computes the distance which the piston must be moved, controls the motor, update the motor position and waits for the next reception.

4.1.2 The ESP8266 Flow Chart

The ESP8266 is the interface between the thermostat and the Internet network. It is through it that the system informs the user about the current temperature and receives remote commands. The Figure 4.2 gives the flowchart of the ESP8266.

The first action of the ESP82866 after initialization is to send its IP address through the UART. This address will allow the user to enter a local web page through which he enters his WIFI credentials (WIFI name and password). After the chip connects to the WIFI, it opens the UART connection and then enters an infinite loop :

- waits for the reception of the current temperature
- updates the website
- reads the future temperature
- sends the desired temperature value to the MCU if necessary
- enters into the sleep mode to reduce energy consumption

- wakes-up after a delay
- waits for the next reception

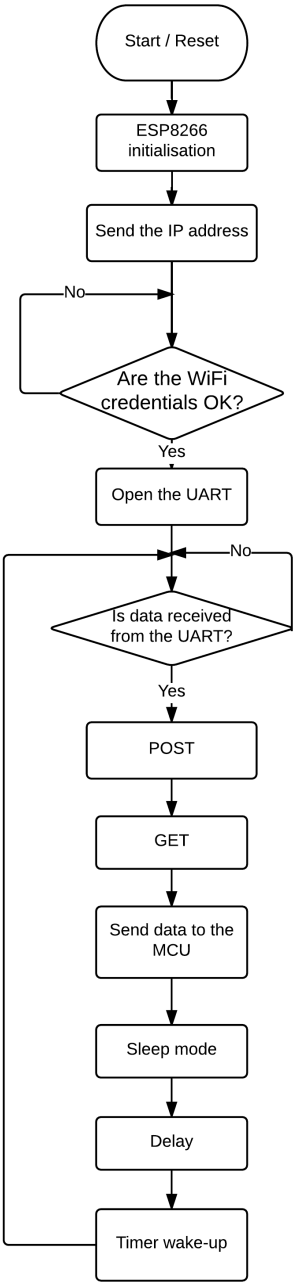


FIGURE 4.2: The ESP8266 flow chart

4.2 The Database and Website Interface

Many ESP8266 projects exploit its ability to implement a mini web server. This approach is very simple and interesting when trying to access the ESP8266 from the same local network. In fact from a smartphone application for example, requests are sent to the router through an url. The router forwards the information to the ESP8266 thanks to its IP address for processing. To access to the chip from the Internet requires more manipulation such as:

- search for the IP address assigned by the ISP (Internet Service Provider) and update it in the application
- Update the router to do port forwarding

These actions are not simple for most users.

Another important problem with the previous approach is that the server running the chip must run continuously. It is not possible to determine when the server is likely to receive requests so it is always available. This approach implies a waste of resources and is not good for battery-powered systems.

For this project, it will be simpler and adequate to build the bulk of the application on a host provider server. Now it is possible to "shut down" the WIFI module and enable it only when needed. A small database is built to allow the thermostat to share data with the user. By using http requests, the ESP8266 will be able to read or update information on the database. The user will access the thermostat data and send orders via a web interface.

4.2.1 The DataBase

A database can be seen as a system that records information and organize it hierarchically . This is something on which the website construction is based. The web hoster (http://www.freewebhostingarea.com) chosen for the project uses MySQL which is a free and well kwown DBMS (DataBase Management System). Figure 4.3 shows a screen-shot of phpMyAdmin, the web application used to manage the database.

The table named "*thermostat*" contains a line with two fields : *current* and *asked* which represent respectively the current temperature of the room and the temperature desired by the user. This line corresponds to one thermostat module. If the number of modules is increased, the number of rows in the table is simply increased too. The thermostat

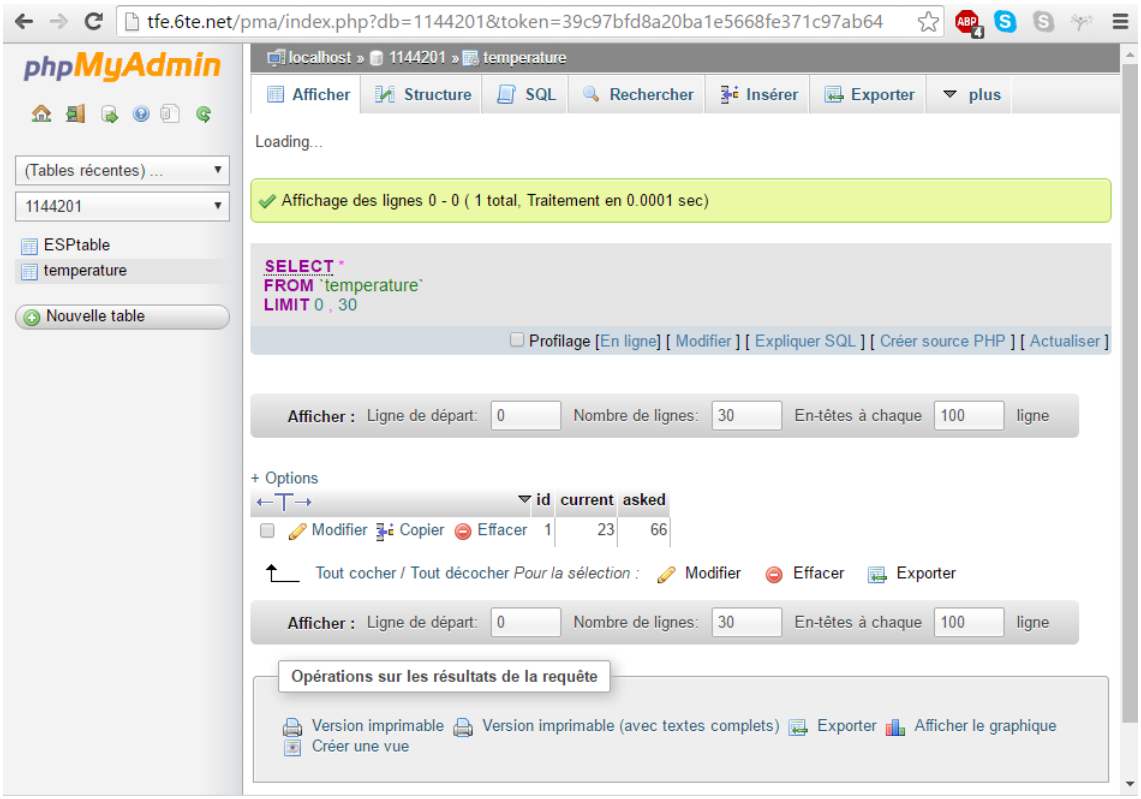


FIGURE 4.3: Design of the database with phpMyAdmin

changes the variable *current* and reads the value of *asked* while the user changes *asked* and reads *current*.

4.2.2 The Website Interface

For the software application, it would be interesting to develop a smart-phone application but this requires to do so for all available operating systems (Symbian OS, Android OS, Apple iOS, Windows OS, Blackberry OS...). The choice to host the application on a simple website is more appropriate since it would be accessible from any smart-phone or PC.

The database is the adequate tool to establish remote communication between the thermostat and the user. But it is not possible for the user to manage directly this communication from phpMyAdmin. He needs a more intuitive GUI. The language PHP is used to make the interface between the user and MySQL. PHP (Hypertext Preprocessor) is a server side scripting language that was designed specifically to produce dynamic web pages via an HTTP server. The PHP code is included in an HTML page and will be executed every time that a visitor displays the page [16]. It is performed at at the server and

generates HTML code or other data viewable in the user's browser . Figure 4.3 shows a screen-shot of the final interface from which the user will interact with the database.

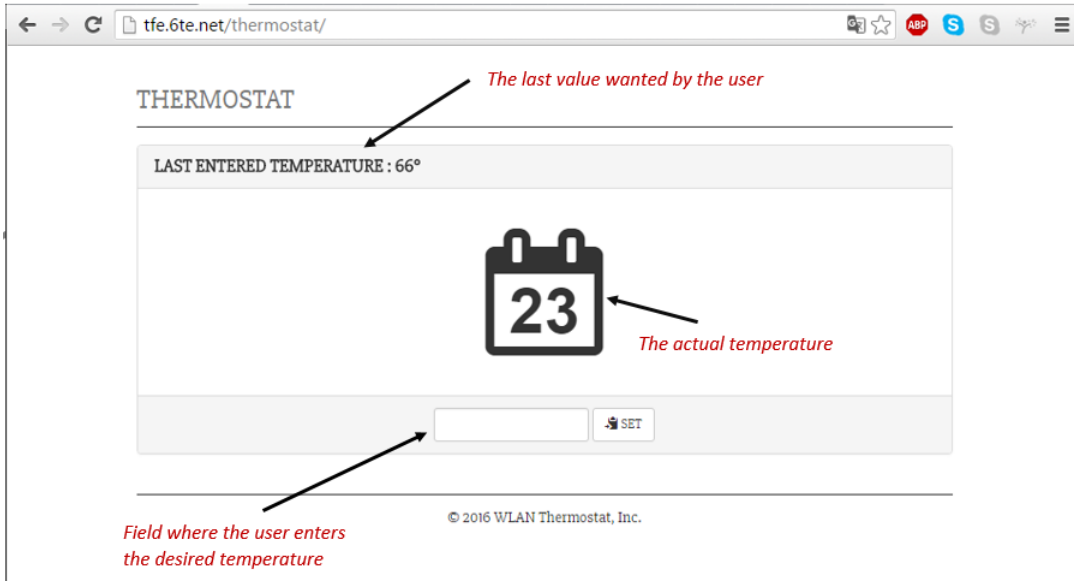


FIGURE 4.4: A view of the web page interface

In AppendixC are shown the scripts from the PHP files used to generate and manage this interface. The structure of the website interface is made from templates available on https://github.com/naousse.

- **index.php** is an access point to the webpage. It verifies if the variable "asked" stored in the database has changed and update the webpage
- **temperature.php** verifies that an action was requested. Depending on the request, it executes a piece of code and echo a key value pair
- **TemperatureController.ph** inherited from the super class Controller, manages the access to the application
- **TemperatureRepository.ph** manages the database
- **Connection.ph** connects to the database

4.3 The MCU Code

4.3.1 The Learning Algorithm

Ideally the learning algorithm could be implemented on the application side to limit the use of the energy resources of the board. Given the energy limitation and the fact that emission/reception of the WiFi module consumes a large part of the energy available, a simplified version of the algorithm is implemented in the MCU. Figure 4.5 presents the flow chart of the learning algorithm. The main idea is to completely close the radiator valve and then open it by regular small steps. At each one waits a delay to balance the temperature. If the value of the temperature is relevant, the number of steps and the value of the temperature are stored in the MCU EEPROM.

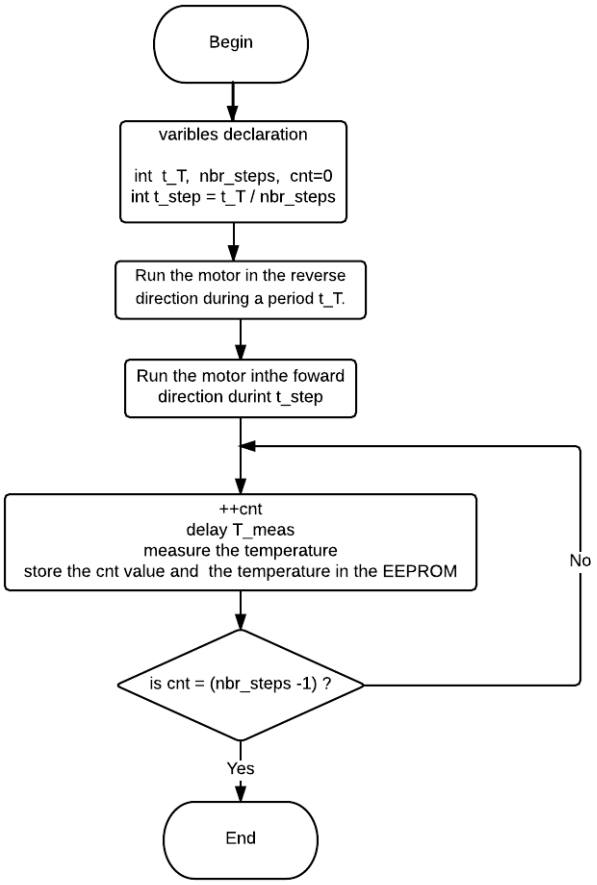


FIGURE 4.5: Learning algorithm flow chart

4.3.2 The Main Code

The code of the PIC18LF45K20is located in AppendixD

4.4 The ESP8266

The ESP8266 is a MCU that supports programs added by users like any other MCU. Most of the available modules on the market are equipped with "AT" firmwares that supports AT commands via the serial port. This firmware is easy to use but is poorly suited when it comes to make big programs. To overcome this difficulty, the firmware can be replaced by one of the many top performing firmware which include NodeMCU that can be programmed in Python or Lua language.

4.4.1 The Lua Language

The Lua is a programming language developed in 1993 by a team of the Pontifical Catholic University of Rio de Janeiro. It is a powerful language , efficient, lightweight supporting different type of programming like procedural programming, object-oriented programming, functional programming,...etc The Lua presents several advantages:

- it is one of the fastest language
- it is portable
- it is embeddable
- it is powerful but simple
- it is a freeware
- its occupies a small amount of memory

The availability of many libraries and development tools for the ESP8266 and the simplicity of their use make the Lua language one of the most used.

4.4.2 NodeMCU

NodeMCU is a firmware specially developed for programming the ESP8266 more intuitively with Lua or python language. One of the main advantages of NodeMCU is that it gives the opportunity to build specific firmwares. The user, depending on the project he is working on, has the possibility to choose only the tools he needs and thus build a lighter firmware (http://nodemcu-build.com).

The NodeMCU firmware built for this project includes the following modules :

- **CJSON** to encode/decode data to/from JSON which is a key value pair representation to share data

- **Enduser setup** which provides a simple way to the user to enter credentials for the WIFI connection through an url (Figure 4.6)
- **File** to provides access to the file system and its individual files
- **GPIO** to access the GPIO (General Purpose Input/Output) subsystem
- **HTTP** that provides an interface to do http requests (GET, POST, PUT, DELETE)
- **Node** to access to system-level features such as sleep and restart
- **RTC time** provides advanced timekeeping support
- **Timer** to allow access to simple timers
- **UART** for the configuration of the UART serial port
- **WiFi** for interfacing with WiFi

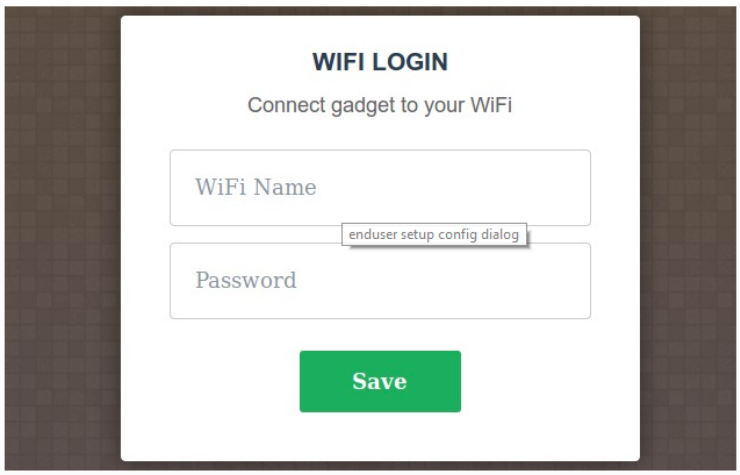


FIGURE 4.6: The end-user connection portal [9]

4.4.3 The ESP8266 Code

The ESP8266 program is distributed in 4 files: init.lua, openUart.lua, connect.lua and thermo.lua. The content of these files is presented below. The init.lua file is always the first to be read. In this file a Boolean variable and a timer are used to stop the execution of the program in case of a bug. If all goes well, init.lua hands over to openUart.lua which opens the serial link. Connect.lua gets the IP address, sends it to the port to allow the user to enter the URL of the connection portal. Thermo.lua monitors the UART. When the module gets a temperature on the serial port, it updates the website, reads the temperature asked and sends it to the MCU if necessary. The next step is the entry into a deep-sleep mode to wait for the next reception.

init.lua

```
function startup()
    if abort == true then
        print('startup aborted')
        return
        end
    --print("In startup") -- just for debugging
    dofile('openUart.lua')
    end
function initializeUart()
    uart.setup(0, 9600, 8, uart.PARITY_NONE, uart.STOPBITS_1, 0)
    end
    abort = false
tmr.alarm(0,10000,0,startup) --delay to exit in case of malfunctioning
```

openUart.lua

```
--print("in file test.lua")-- for debugging
uart.setup(0, 9600, 8, 0, 1, 0)
dofile('connect.lua')
```

connect.lua

```
wifi.setmode(wifi.STATIONAP)
wifi.ap.config({ssid="MyPersonalSSID",auth=wifi.AUTH_OPEN})
enduser_setup.manual(true)
enduser_setup.start(
  function()
    print("To:"..wifi.sta.getip())
  end,
  function(err, str)
    print("enduser_setup: Err #" .. err .. ": " .. str)
  end
);

uart.write(0,wifi.sta.getip().."\r\n");
print("Connected")
dofile('thermo.lua'))
```

thermo.lua

```
currentTemp=20

uart.on("data", "\r",
  function(data) --print("receive from uart:", data)
    currentTemp=tonumber(data)
    --print("A POST method");
    http.get("http://tfe.6te.net/thermostat/temperature.php?action=setCurrent&current=".. 
    currentTemp, nil, function(code, data)
        if (code < 0) then
            print("HTTP request failed")
        else
            --print(code, data);
            print("update the room temperature is: "..currentTemp);
        end
    end)
    tmr.delay(10000)
    --print("A GET method")
    http.get("http://tfe.6te.net/thermostat/temperature.php?action=getFuture",
    nil, function(code, data)
        if (code < 0) then
            print("HTTP request failed")
        else
            result = cjson.decode(data);
            temperature = result["future"];
            --print(code, data);
            if (temperature ~= futureTemp) then
                --print("A new value");
                uart.write(0,"NT"..temperature.."\r\n"); -- the future te

            elseif (temperature == futureTemp) then
                --print("The same value");
                --uart.write(0,"ActualT"..temperature.."\r\n");
            end
       end
    end)
    --tmr.delay(10000)
    node.dsleep(840 * 1000000) -- 840s = 14min
    if data=="quit\r\n" then
      uart.on("data") -- unregister callback function
```

```
    end
end, 0)
```

Deep Sleep Mode

In the ESP8266 deep-sleep mode, only RTC is powered on. The rest of the chip is powered off and the current consumption is typically $10\mu A$. To get access to this mode, a hardware configuration has to be set: a connection between the XPD_DCDC (GPIO16 of the MCU) and the external RST pins (AppendixB). Unfortunately for the ESP8266-01 version to connect those pins is not easy. The connection has to be done with a thin wire like shown on Figure 4.7 below.

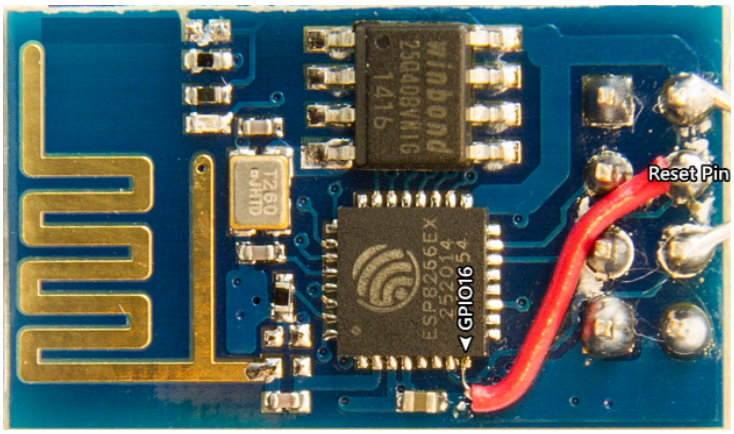


FIGURE 4.7: ESP8266-01 hardware connection to enable deep-sleep [24]

Chapter 5

Experimental Results and Discussion

The previous two chapter dealt with the design of the hardware and software parts of the thermostat centered around the PIC18LF45K50 and the esp8266-01. The main characteristics that the prototype of the connected thermostat should have are its battery power, its ability to reduce the energy consumption and its control (both manual and remote).

This chapter discusses the results of the operating tests of the thermostat. A stripboard is used to perform tests on the thermostat. The first part of this chapter focuses on its operation. The second part deals with the thermostat consumption to evaluate the lifetime of the battery. The last part is a simulation of the energy that could be saved through the use of a connected thermostat.

5.1 The Thermostat

The connected thermostat designed fulfilled all requirements defined at the beginning of the project:

- it is a low power device and battery powered
- it connects directly to a home WiFi router without going through the intermediary of a central gateway .
- it can post information on a website and retrieve commands from the user.
- it is remotely controllable
- it obtains IP address automatically,this making it convenient for the user
- it is similar in size than most electronic thermostats on the market
- adding connectivity does not affect significantly the cost since the added component is very cheap. This allows obtain at the end a cheap thermostat with good features.

5.2 The Power Consumption

The card is powered by a battery. So it is important to evaluate the battery lifetime. Although the datasheet of the various components indicate the ranges of values, measurements on board helps to improve the battery lifetime prediction. Figures 5.1, 5.2 and 5.3 respectively represent the measurement device (Rigol DS1052E DSO), the measurement of the current consumed during the transmission/reception processes and finally the measurement of the maximum current consumption duration.

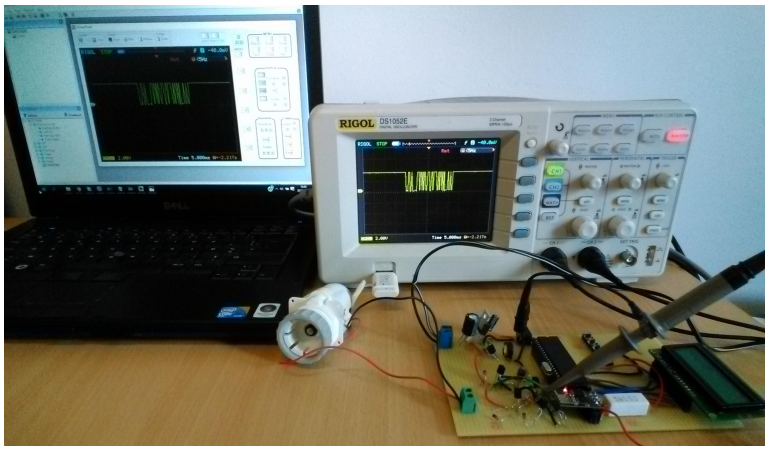


FIGURE 5.1: Measurements on the stripboard

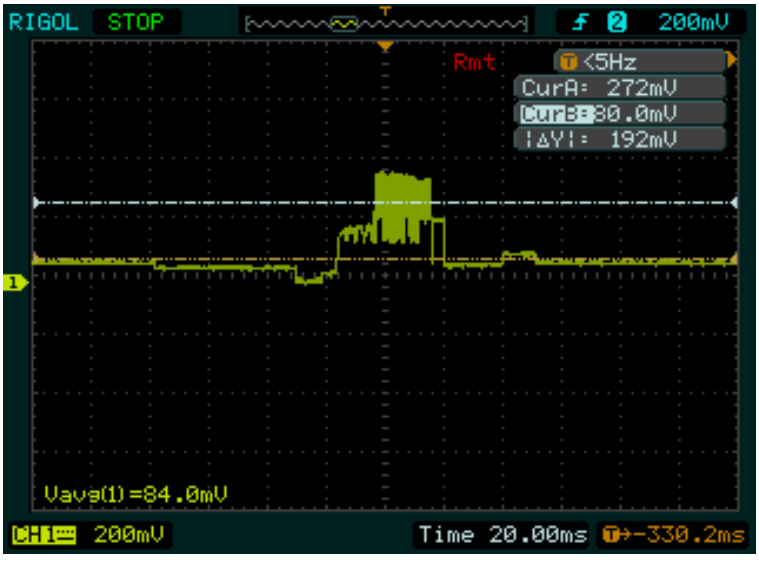


FIGURE 5.2: Measurement of the Tx/Rx current consumption

The Rigol DS1052E ($50MHz/1GSa/s$) gives an idea of the measurements. It is not fast enough to make accurate measurements. The current in the Tx/Rx phases is measured through a $1\Omega/1W$ resistor and is it about 180mA. To measure the duration, GPIO2

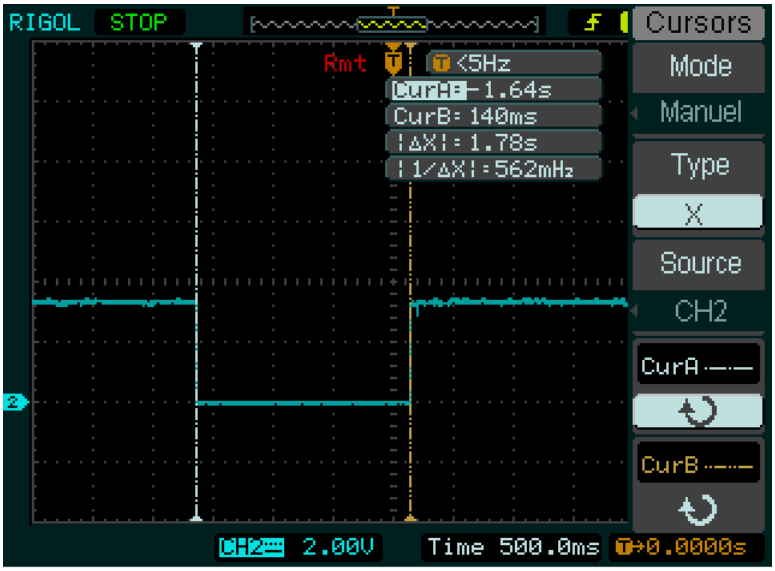


FIGURE 5.3: Measurement of the Tx/Rx duration

signal is set to LOW just before the "POST" method and set to HIGH just after the receipt of the response of the "GET" method. This gives a time $T_{TX/RX} = 1.78s$. This time takes into account the transmissions, the receptions but also the data processing.

The following assumptions are made to estimate the lifetime of the battery:

- the code execution lasts 3s
- 4 update of temperature per hour so a sleep time of 897s
- a consumption of $200mA$ during code execution
- a consumption of $5mA$ during the sleep mode
- a $5400mAh$ capacity battery
- a safety discharge 20%

The capacity consumed in one hour is given by:

$$C_1 = 4 * \frac{3 * 200 + 897 * 5}{3600} = 5.65mA$$

Then, the battery lifetime is:

$$lifetime = \frac{0.8 * 5400}{5.65} = 764.6hours$$

$$lifetime = \frac{0.8 * 5400}{5.65} = 31days20hours$$

One way to improve the power consumption of the modules is to remove both the power and status light emitting diodes from the board. These operations are very delicate and allow to recover around $5mA$.

5.3 Energy Saving Estimation

It is always important to pay attention to the rational use of energy. One of the main advantages of using the connected thermostat compared to a mechanical thermostat is that it helps to reduce the energy consumption. To assess the quality of the system, a simple case was studied. Consider a room of $4 * 4 * 2.5m^3$. The total area (walls, floor and ceiling) is $72m^2$. The temperature of the room is supposed to be at 20 °C all the time even if the outside temperature varies. The following simplifying assumptions are used:

- the temperature is assumed to be 15 °C during the day and 10 °C during the night.
- day and night are assumed to have equal durations
- the static model is adopted for all the walls (stationary operation)
- all internal walls are in the internal temperature
- all the outside walls are in the external temperature
- the convection exchange resistance between the inside and the internal faces of the walls is zero
- the convection exchange resistance between the outside and the outside walls is zero
- the radiation exchange resistance is equal to zero.

In steady state, the thermal power is given by

$$\dot{Q} = A * U * (T_{in} - T_{out})$$

where A represents the total surface and AU the global exchange coefficient.

The energy consumed is:

$$Q = \dot{Q} * time$$

In the case studied, and knowing that $U = 0.22W.m^{-2}K * -1$, the total energy used in one day by a mechanical valve preset in the night is:

$$Q_{mec/day} = 24 * 72 * 0.22 * (20 - 10)$$

$$Q_{mec/day} = 3801.6Wh$$

If instead of the mechanical valve the connected thermostat is used,

$$Q_{connected/day} = 12 * 72 * 0.22 * (20 - 15) + 12 * 72 * 0.22 * (20 - 10)$$

$$Q_{connected/day} = 2851.2Wh$$

The amount of energy that can potentially be save is given by:

$$Q_{saved/day} = Q_{mec/day} - Q_{connected/day} = 950.4Wh$$

that represents in this case 25% of the total energy.

In the case of comparing the thermostat connected to a basic electronic thermostat, a parameter to highlight is the occupancy. The electronic thermostat adapts to external weather conditions but can not turn off the heater when the room is not occupy. Assuming that the room is occupied all the night and one third of the day (16 hours), the amount of energy that can potentially be save is :

$$Q_{connected/day} = (2/3) * 12 * 72 * 0.22 * (20 - 15)$$

$$Q_{connected/day} = 633.6Wh$$

that represents in this case 22.22% of the total energy.

Although this is a very simple case, it shows the potential of economic gain that the connected thermostat represents.

Chapter 6

Conclusion

The goal of the thesis was to investigate whether a standard programmable radiator thermostat can be converted into a WLAN connected device that can directly be connected to a standard home WiFi. A software application was supposed to be developed to control the thermostat remotely. the main interest addressed by this study was then to build a smart thermostat cheaper and less bulkier than those available on the market.

A literature review helped to understand that one important factor in the development of the Internet of Things is the democratization of some technologies across the development the telecommunications sector and in particular of smart-phone devices. The exploitation of the advancement in the field of microelectronics has allowed researchers to develop low-power IoT devices adapted to be powered by batteries .

6.1 Contribution

A study of the features of classical electronic thermostats on the market allowed to design an electronic thermostat board for home radiator. In addition to its small sizes, it includes a WiFi transceiver that has good performances and low cost compared to its competitors.

The designed thermostat has several good features compared to the classic electronic thermostat or mechanical valve. The first thing that has been shown is that we can very simply added connectivity to any WLAN module such objects the ESP8266 so as to make a smart-thing. The resulting module can connect directly to a WiFi home without going through a central gateway that involves extra costs. The reduced cost of wireless module and its simple connection to an MCU card with a serial link has allowed to produce a connected thermostat with very good features but a much cheaper than those available on the market.

A web application has been developed to allow users to remotely control the thermostat via the Internet. They can read or change the temperature of the room where the thermostat is located from any computer or phone with an Internet connection.d. One of the main advantages of this application is that it is possible to add easily new algorithms in the server side to manage the room temperature. A simple algorithm enables the MCU to learn how to modulate the displacement of the transmission shaft to get a certain temperature.

The design of the connected thermostat allows cost saving through the significant reduction of the energy losses by adapting to external weather conditions or by activating the heater with respect to the the occupancy of the room.

6.2 Prospect

Although the main objectives set in the beginning of the project have been achieved, this work represents only one approach of the problem to solve. It would be very interesting to further investigate how to improve the proposed solution. Among further researches, one could :

- study the possibility to use the heat source that represents the radiator and improve the battery lifetime
- consider replacing the MCU/ESP8266-01 pair by an ESP8266-12 that has more GPIOs, and modules such as ADC, PWM,...
- develop smart-phone applications and examine how the use of MQTT could help to reduce power consumption

Appendix A

PIC18LF25K50 main features

MICROCHIP

PIC18(L)F2X/45K50

28/40/44-Pin, Low-Power, High-Performance Microcontrollers with XLP Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Crystal-less Full Speed (12 Mb/s) and Low-Speed Operation (1.5 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 Bidirectional)
- 1 Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver

Flexible Oscillator Structure:

- 3x and 4xPLL Clock Multipliers
- Two External Clock modes, Up to 48 MHz (12 MIPS)
- Internal 31 kHz Oscillator
- Internal Oscillator, 31 kHz to 16 MHz
  - Factory calibrated to ± 1%
  - Self-tune to ± 0.20% max. from USB or secondary oscillator
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- Up to 33 I/O pins plus 3 Input-Only Pins:
  - High-current Sink/Source 25 mA/25 mA
  - Three programmable external interrupts
  - 11 programmable interrupts-on-change
  - Nine programmable weak pull-ups
  - Programmable slew rate
- SR Latch
- Enhanced Capture/Compare/PWM (ECCP) module:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
  - Pulse steering control
- Capture/Compare/PWM (CCP) module
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all four modes) and I^2C™ Master and Slave modes
- Two Analog Comparators with Input Multiplexing
- 10-Bit Analog-to-Digital (A/D) Converter module:
  - Up to 25 input channels
  - Auto-acquisition capability
  - Conversion available during Sleep
- Digital-to-Analog Converter (DAC) module:
  - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
  - 5-bit rail-to-rail resistive DAC with positive and negative reference selection
- High/Low-Voltage Detect module
- Charge Time Measurement Unit (CTMU):
  - Supports capacitive touch sensing for touch screens and capacitive switches
- Enhanced USART module:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-wake-up on Start bit
  - Auto-Baud Detect

Extreme Low-Power Management with XLP:

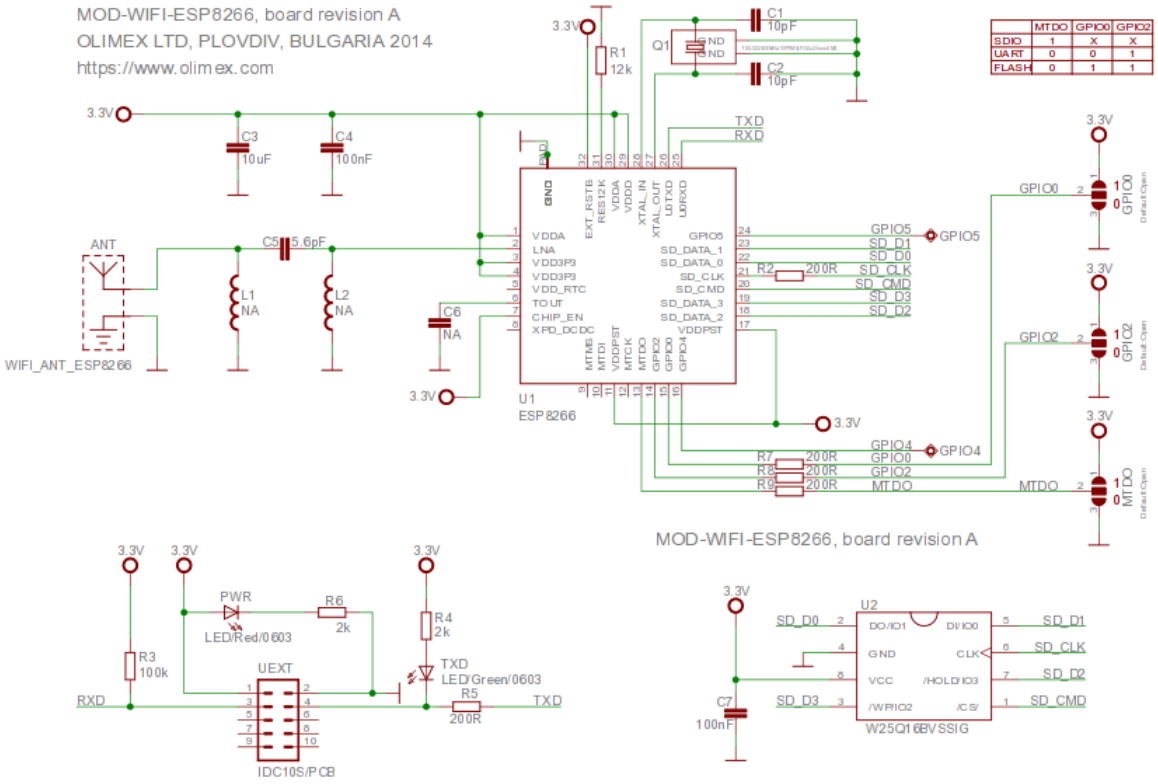
- Sleep mode: 20 nA, typical
- Watchdog Timer: 300 nA, typical
- Timer1 Oscillator: 800 nA @ 32 kHz
- Peripheral Module Disable

Special Microcontroller Features:

- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Power Management Features:
  - Run: CPU on, peripherals on, SRAM on
  - Idle: CPU off, peripherals on, SRAM on
  - Sleep: CPU off, peripherals off, SRAM on
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with Three Breakpoints via Two Pins
- Optional dedicated ICD/ICSP Port (44-pin TQFP Package Only)
- Wide Operating Voltage Range:
  - F devices: 2.3V to 5.5V
  - LF devices: 1.8V to 3.6V
- Flash Program Memory of 10,000 Erase/Write Cycles Minimum and 20-year Data Retention

Appendix B

The ESP8266 Electronic Schematic



Appendix C

The Code of the Web Interface

C.1 index.php

```
<?php

require_once './vendor/autoload.php';
use Thermostat\Controller\TemperatureController;

/**
 * indes.php temperature.php are the access point to the webpage. It verifies
 * if the variable "asked", store it in the database update the webpage
 **/

$temperatureController = new TemperatureController();

if (isset($_POST['asked']) && is_numeric($_POST['asked'])){
    $asked = intval($_POST['asked']);
    if ($asked>=0){
        $temperatureController->setFutureTemperatureAction(1,$asked);
        header('Location: http://'.$_SERVER['HTTP_HOST'].$_SERVER['REQUEST_URI']);
    }
}
$temperatureController->indexAction(1);

?>
```

C.2 temperature.php

```
<?php

require_once './vendor/autoload.php';

use Thermostat\Controller\TemperatureController;

/**
 * It verifies that an action was requested. Depending on the action executes a
 * piece of code and echo a key value pair
 */

$temperatureController = new TemperatureController();

if (isset($_GET['action'])) {
    if ($_GET['action'] == "setCurrent") {
        if (isset($_GET['current']) && is_numeric($_GET['current'])) {
            $temperature = intval($_GET['current']);
            if ($temperature >= 0) {
                $temperatureController->setCurrentTemperatureAction(1, $temperature);
                echo json_encode(array("status"=>201));
                return;
            }
        }
        echo json_encode(array("status"=>500, "error"=>"The value you provided
        is not correct!"));
        return;
    }
    if ($_GET['action'] =="setFuture") {
        if (isset($_GET['future']) && is_numeric($_GET['future'])) {
            $temperature = intval($_GET['future']);
            if ($temperature >= 0) {
                $temperatureController->setFutureTemperatureAction(1, $temperature);
                echo json_encode(array("status"=>201));
                return;
            }
        }
        echo json_encode(array("status"=>500, "error"=>"The value you provided
        is not correct!"));
        return;
    }
    if ($_GET['action'] == "getFuture") {

        $temperature = $temperatureController->getFutureTemperatureAction(1);
        echo json_encode(array("status"=>200, "future"=>$temperature));
```

```
        return;
    }
    if ($_GET['action'] == "getCurrent") {
        $temperature = $temperatureController->getCurrentTemperatureAction(1);
        echo json_encode(array("status"=>200, "current"=>$temperature));
        return;
    }
}

?>
```

C.3 TemperatureController.php

```
<?php

namespace Thermostat\Controller;

use Thermostat\Entity\Temperature;
use Thermostat\Repository\TemperatureRepository;

/**
 * Inherited from the super class Controller.
 * It manages the access to the application
 */

class TemperatureController extends Controller {

    public function indexAction($id) {
        $temperatureRepository = new TemperatureRepository();

        $this->render(array(/*affiche */
            "view" => "Temperature/index",/*vue*/
            "data" => array( /*les valeurs a afficher sur la vue*/
                "temperature" => $temperatureRepository->getCurrent($id),
                "future" => $temperatureRepository->getFuture($id)
            )
                )
        );
    }
    public function setCurrentTemperatureAction($id, $current) {
        $temperatureRepository = new TemperatureRepository();
        $temperatureRepository->setCurrent($id, $current);
    }
    public function getCurrentTemperatureAction($id) {
```

```
        $temperatureRepository = new TemperatureRepository();
        return $temperatureRepository->getCurrent($id);
    }
    public function setFutureTemperatureAction($id, $current) {
        $temperatureRepository = new TemperatureRepository();
        $temperatureRepository->setFuture($id, $current);
    }
    public function getFutureTemperatureAction($id) {
        $temperatureRepository = new TemperatureRepository();
        return $temperatureRepository->getFuture($id);
    }
}
```

C.4 TemperatureRepository.php

```
<?php

namespace Thermostat\Repository;

use Thermostat\Entity\Temperature;
/**
* Manages the database
*/

class TemperatureRepository extends Repository {
    public function save(Temperature $temperature) {

        $this->connection->exec(
                "INSERT INTO "
                . "temperature(current, asked)"
                . " VALUES (" . $temperature->getCurrent() . "," .
                $temperature->getFuture() . ")"
        );
    }
    public function update(Temperature $temperature) {
        $this->connection->exec(
                "UPDATE TABLE temperature "
                . "SET "
                . "current = '" . $temperature->getCurrent() . "' "
                . "future = '" . $temperature->getFuture() . "' "
                . "WHERE id = " . $temperature->getId()
        );
    }
```

```
     public function setFuture($id, $temperature) {
        $this->connection->exec(
                "UPDATE temperature "
                . "SET "
                . "asked = " . $temperature . " "
                . "WHERE id = " . $id
        );
    }
    public function setCurrent($id, $temperature) {
        $this->connection->exec(
                "UPDATE temperature "
                . "SET "
                . "current = " . $temperature . " "
                . "WHERE id = " . $id
        );
    }
    public function getCurrent($id) {
        $resultats = $this->connection->query("SELECT current FROM
        temperature WHERE id=".$id);
        $resultats->setFetchMode(\PDO::FETCH_OBJ);
        $current = null;
        while ($resultat = $resultats->fetch()) {
            $current = $resultat->current;
        }
        $resultats->closeCursor();
        return $current;
    }
    public function getFuture($id) {
        $resultats = $this->connection->query("SELECT asked FROM
        temperature WHERE id=".$id);
        $resultats->setFetchMode(\PDO::FETCH_OBJ);
        $future = null;
        while ($resultat = $resultats->fetch()) {
            $future = $resultat->asked;
        }
        $resultats->closeCursor();
        return $future;
    }
}
```

C.5 Connection.php

```
<?php

namespace Thermostat\Repository;

/**
 * Connects to the database
 */

class Connection {
    protected static $dbname = "DB_name";
    protected static $db = "mysql";
    protected static $username = "user_name";
    protected static $password = "mypassword";
    protected static $host = "localhost";
    protected static $port = 3306;
    protected static  $connection = null;

    public static function getConnection() {
        $dsn = self::$db.':host='.self::$host.';port='.self::$port.';dbname='.self::$dbname;
        $options = array(
            \PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8',
            \PDO::ATTR_ERRMODE => \PDO::ERRMODE_EXCEPTION
        );
        if (self::$connection == null){
            self::$connection = new \PDO($dsn, self::$username, self::$password, $options);
        }
        return self::$connection;
    }
}
```

Appendix D

The PIC18LF45K20 Code

D.1 Includes.h

```
#ifndef __INCLUDES_H
#define __INCLUDES_H

// Define CPU Frequency
// This must be defined, if __delay_ms() or
// __delay_us() functions are used in the code
#define _XTAL_FREQ   20000000
    #include <htc.h>
    #include "UART.h"
    #include "ISR.h"
#endif
```

D.2 ISR.c

```
#include "Includes.h"
void interrupt ISR(void)
{
if(RCIF)  // If UART Rx Interrupt
{
if(OERR) // If over run error, then reset the receiver
{
CREN = 0;
CREN = 1;
}
//SendByteSerially(RCREG); // Echo back received char
}
}
```

D.3 lcd.h

```
//LCD Functions Developed by electroSome

void Lcd_Port(char a)
{
if(a & 1)
D4 = 1;
else
D4 = 0;

if(a & 2)
D5 = 1;
else
D5 = 0;

if(a & 4)
D6 = 1;
else
D6 = 0;

if(a & 8)
D7 = 1;
else
D7 = 0;
}
void Lcd_Cmd(char a)
{
RS = 0;             // => RS = 0
Lcd_Port(a);
EN  = 1;             // => E = 1
        __delay_ms(4);
        EN  = 0;             // => E = 0
}

Lcd_Clear()
{
Lcd_Cmd(0);
Lcd_Cmd(1);
}

void Lcd_Set_Cursor(char a, char b)
{
char temp,z,y;
if(a == 1)
{
```

```
  temp = 0x80 + b - 1;
z = temp>>4;
y = temp & 0x0F;
Lcd_Cmd(z);
Lcd_Cmd(y);
}
else if(a == 2)
{
temp = 0xC0 + b - 1;
z = temp>>4;
y = temp & 0x0F;
Lcd_Cmd(z);
Lcd_Cmd(y);
}
}

void Lcd_Init()
{
  Lcd_Port(0x00);
   __delay_ms(20);
  Lcd_Cmd(0x03);
__delay_ms(5);
  Lcd_Cmd(0x03);
__delay_ms(11);
  Lcd_Cmd(0x03);
  ////////////////////////////////////////////////////
  Lcd_Cmd(0x02);
  Lcd_Cmd(0x02);
  Lcd_Cmd(0x08);
  Lcd_Cmd(0x00);
  Lcd_Cmd(0x0C);
  Lcd_Cmd(0x00);
  Lcd_Cmd(0x06);
}

void Lcd_Write_Char(char a)
{
   char temp,y;
   temp = a&0x0F;
   y = a&0xF0;
   RS = 1;             // => RS = 1
   Lcd_Port(y>>4);             //Data transfer
   EN = 1;
   __delay_us(40);
   EN = 0;
   Lcd_Port(temp);
```

```
   EN = 1;
   __delay_us(40);
   EN = 0;
}

void Lcd_Write_String(char *a)
{
int i;
for(i=0;a[i]!='\0';i++)
   Lcd_Write_Char(a[i]);
}

void Lcd_Shift_Right()
{
Lcd_Cmd(0x01);
Lcd_Cmd(0x0C);
}

void Lcd_Shift_Left()
{
Lcd_Cmd(0x01);
Lcd_Cmd(0x08);
}
```

D.4 UART.c

```
/*  File:     UART.c
    Purpose:  To provide UART functionality for PIC uC
*/

#include "Includes.h"

void InitUART(void)
{
//TRISC6 = 1;    // TX Pin
    TRISC6 = 0;    // TX Pin
TRISC7 = 1;    // RX Pin

SPBRG = ((_XTAL_FREQ/16)/BAUDRATE) - 1;
BRGH  = 1;                    // Fast baudrate
SYNC  = 0; // Asynchronous
SPEN  = 1; // Enable serial port pins
CREN  = 1; // Enable reception
```

```
SREN  = 0; // No effect
TXIE  = 1; // Disable tx interrupts
RCIE  = 1; // Enable rx interrupts
TX9   = 0; // 8-bit transmission
RX9   = 0; // 8-bit reception
TXEN  = 0; // Reset transmitter
TXEN  = 1; // Enable the transmitter
}


void SendByteSerially(unsigned char Byte)  // Writes a character to the serial port
{
while(!TXIF);  // wait for previous transmission to finish
TXREG = Byte;
}

unsigned char ReceiveByteSerially(void)   // Reads a character from the serial port
{
if(OERR) // If over run error, then reset the receiver
{
CREN = 0;
CREN = 1;
}

while(!RCIF);  // Wait for transmission to receive

return RCREG;
}

void SendStringSerially(const unsigned char* st)
{
while(*st)
SendByteSerially(*st++);
}
```

D.5 main.c

```
#define _XTAL_FREQ 20000000

#define RS LATCbits.LATC5
#define EN LATCbits.LATC4
#define D4 LATDbits.LATD4
#define D5 LATDbits.LATD5
```

```
#define D6 LATDbits.LATD6
#define D7 LATDbits.LATD7
#define BUTTON_ECO LATBbits.LATB3
#define BUTTON_UP LATBbits.LATB4
#define BUTTON_DOWN LATBbits.LATB5
#define BUTTON_OK LATBbits.LATB6

#include <xc.h>
#include<string.h>
#include<p18f45k20.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "lcd.h"
#include "Includes.h"

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

// CONFIG1H
#pragma config FOSC = HS
#pragma config FCMEN = OFF
#pragma config IESO = OFF

// CONFIG2L
#pragma config PWRT = OFF
#pragma config BOREN = ON
#pragma config BORV = 18

// CONFIG2H
#pragma config WDTEN = OFF
#pragma config WDTPS = 32768

// CONFIG3H
#pragma config CCP2MX = PORTC
#pragma config PBADEN = ON
#pragma config LPT1OSC = OFF
#pragma config HFOFST = ON
#pragma config MCLRE = ON

// CONFIG4L
#pragma config STVREN = ON
#pragma config LVP = OFF
#pragma config XINST = OFF

// CONFIG5L
```

```
#pragma config CP0 = OFF
#pragma config CP1 = OFF
#pragma config CP2 = OFF
#pragma config CP3 = OFF

// CONFIG5H
#pragma config CPB = OFF
#pragma config CPD = OFF

// CONFIG6L
#pragma config WRT0 = OFF
#pragma config WRT1 = OFF
#pragma config WRT2 = OFF
#pragma config WRT3 = OFF

// CONFIG6H
#pragma config WRTC = OFF
#pragma config WRTB = OFF
#pragma config WRTD = OFF

// CONFIG7L
#pragma config EBTR0 = OFF
#pragma config EBTR1 = OFF
#pragma config EBTR2 = OFF
#pragma config EBTR3 = OFF

// CONFIG7H
#pragma config EBTRB = OFF


char *currentTemp;
char futureTemp[2];
char addressToConnect[20];


void myDelay_n10ms(int n){

    while(n > 0){
        __delay_ms(10);
        --n;
    }
}

void initPic(void){

    TRISA = 0xFF;
```

```
    TRISB = 0xFF;
    TRISCbits.RC4 = 0;
    TRISCbits.RC5 = 0;
    TRISD = 0x00;
}

void ADCinit()
{
 ADCON1 =0b00001101;
 //ADCON1 =0b00000000;
 ADCON2 = 0b10001010;//ADCON2 setup: rigth justified, Tacq =2Tad, Tad=32*Tosc (or Fosc/32)
}

// reverses a string 'str' of length 'len'
void reverse(char *str, int len)
{
    int i=0, j=len-1, temp;
    while (i<j)
    {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
        i++; j--;
    }
}

 // Converts a given integer x to string str[].  d is the number
 // of digits required in output. If d is more than the number
 // of digits in x, then 0s are added at the beginning.
int intToStr(int x, char str[], int d)
{
    int i = 0;
    while (x)
    {
        str[i++] = (x%10) + '0';
        x = x/10;
    }
    // If number of digits required is more, then
    // add 0s at the beginning
    while (i < d)
        str[i++] = '0';
    reverse(str, i);
    str[i] = '\0';
    return i;
}
```

```
// Converts a floating point number to string.
void myftoa(float n, char *res, int afterpoint)
{
int temp_int[2] = {0};
    // Extract integer part
    int ipart = (int)n;
    // Extract floating part
    float fpart = n - (float)ipart;
    // convert integer part to string
    int i = intToStr(ipart, res, 0);

    // check for display option after point
    if (afterpoint != 0)
    {
        res[i] = '.';  // add dot
        // Get the value of fraction part upto given no.
        // of points after dot. The third parameter is needed
        // to handle cases like 233.007
        fpart = fpart * pow(10, afterpoint);

        intToStr((int)fpart, res + i + 1, afterpoint);
    }
}

unsigned int ADCRead(unsigned char ch)
{
   if (ch>13) return 0;  //Invalid Channel

   ADCON0=0x00;
   ADCON0=(ch<<2);   //Select ADC Channel
   ADCON0bits.ADON=1;  //switch on the adc module
   ADCON0bits.GO_DONE=1;//Start conversion
   while(ADCON0bits.GO_DONE); //wait for the conversion to finish
   ADCON0bits.ADON=0;  //switch off adc
   return ADRES;
}

void temp(){
float T;
int i, result;
float conv;
char res[4];
char temp[3];
  int temp_int;
ADCinit();
    result = ADCRead(0);//Set the delay to the 8 MSB
```

```
    conv = (result*3.3)/1024.0;
    T = 25.0 +(conv -2.98)*100; //float
//FLOAT TO CHAR
temp_int = T;

if (T < 0.0) {
myftoa(-T, temp, 0);

        currentTemp = temp;

Lcd_Set_Cursor(1, 6);
     Lcd_Write_Char('-');
Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("Temp:");
        Lcd_Set_Cursor(1, 7);
    Lcd_Write_String(temp);
        SendStringSerially(currentTemp); // Send string on UART
      SendStringSerially("\r\n");
}
else if(T < 100.0 && T >= 0.0){
     myftoa(T, res, 0);
        currentTemp = res;
Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("Temp:");
        Lcd_Set_Cursor(1, 6);
   Lcd_Write_String(res);
        SendStringSerially(currentTemp); // Send string on UART
      SendStringSerially("\r\n");
  }
else if(T >= 100.0){
    myftoa(T, res, 0);
        currentTemp = res;
Lcd_Set_Cursor(1, 1);
    Lcd_Write_String("Temp:");
        Lcd_Set_Cursor(1, 6);
    Lcd_Write_String(res);
        SendStringSerially(currentTemp); // Send string on UART
      SendStringSerially("\r\n");
}

}


void linkToConnect(void){
    PEIE = 0;   // disable Peripheral Interrupts
```

```
    Lcd_Clear();
    Lcd_Set_Cursor(1,1);
    Lcd_Write_String(addressToConnect);
myDelay_n10ms(1000);

    PEIE = 1;   // Enable Peripheral Interrupts
}

void interrupt ISR(void)
{

    if(RCIF)  // If UART Rx Interrupt
{
if(OERR) // If over run error, then reset the receiver
{
CREN = 0;
CREN = 1;
}

        char string[20];

        char buffer[20];
        int i = 0;

        while((RCREG != '\r')){
            buffer[i] = ReceiveByteSerially();
            ++i;
        }
        if((buffer[0]=='1')&&(buffer[1]=='9')&&(buffer[2]=='2')&&(buffer[3]=='.')){
            memcpy(addressToConnect, buffer, i);
            addressToConnect[i] = '\0';
            linkToConnect();

        }
        else if((buffer[0]=='N')&&(buffer[1]=='T')){
            memcpy(string, buffer, i);
            string[i] = '\0';
            futureTemp[0] = buffer[2];
            futureTemp[1] = buffer[3];
            futureTemp[2] = '\0';
            Lcd_Clear();
            Lcd_Set_Cursor(2,1);
            Lcd_Write_String("Wanted: ");
            Lcd_Set_Cursor(2,9);
            Lcd_Write_String(futureTemp);
        }
```

```
        else if((buffer[1]=='N')&&(buffer[2]=='T')){
            memcpy(string, buffer, i);
            string[i] = '\0';
            futureTemp[0] = buffer[3];
            futureTemp[1] = buffer[4];
            futureTemp[2] = '\0';
            Lcd_Clear();
            Lcd_Set_Cursor(2,1);
            Lcd_Write_String("Wanted: ");
            Lcd_Set_Cursor(2,9);
            Lcd_Write_String(futureTemp);
        }
}

int main()
{
  initPic();
  InitUART(); // Intialize UART
  Lcd_Init();

  myDelay_n10ms(1000);
  SendStringSerially("dofile(init.lua)\r\n"); // starts the ESP

  GIE  = 1;   // Enable global interrupts
  PEIE = 1;   // Enable Peripheral Interrupts

    Lcd_Clear();
    Lcd_Set_Cursor(1,1);
    Lcd_Write_String("WLAN Thermostat");
    myDelay_n10ms(100);
    Lcd_Set_Cursor(2,1);
    Lcd_Write_String("ULg 2016");
    myDelay_n10ms(1000);
    Lcd_Clear();

  while(1)
  {
      Lcd_Clear();
      temp();
      myDelay_n10ms(87000);

  }
    return 0;
}
```

Bibliography

[1] eq 3. *MAX! Cube LAN Gateway.* URL: http://www.eq-3.de/produkte/max-heizungssteuerung/max-hausloesung/bc-lgw-o-tw.html.

[2] Perrinjaquet Roger Amphoux Pascal Jaccoud Christophe. *Dictionnaire critique de la domotique.* EPFL, Lausanne, 1989, pp. 44, 76.

[3] Electronic Assembly. *DOGS GRAPHIC SERIES.* URL: http://www.lcd-module.com/eng/pdf/grafik/dogs102-6e.pdf.

[4] Sue Beckingham. *Curious about the Internet of Things? Read on – a short intro. IoT.* URL: https://socialmediaforlearning.com/2015/12/04/curious-about-the-internet-of-things-read-on-a-short-intro-iot/.

[5] Maylab Information Technology Sdn Bhd. *What is IoT?* URL: http://maylab.my/?page_id=9.

[6] Duracell. *Alkaline-Manganese dioxyde, Performance characteristics.* URL: http://docs-europe.electrocomponents.com/webdocs/0215/0900766b802155d9.pdf.

[7] Duracell. *INDUSTRIAL by Duracell.* URL: http://www.all-batteries.fr/media/pdf/PRA2411_UK.pdf.

[8] energie+. *Les vannes thermostatiques.* URL: http://www.energieplus-lesite.be/index.php?id=10965.

[9] Robert Foss. *enduser setup Module.* URL: http://nodemcu.readthedocs.io/en/master/en/modules/enduser-setup/.

[10] John Greenough. *The 'Internet of Things' Will Be The World's Most Massive Device Market And Save Companies Billions Of Dollars.* URL: http://uk.businessinsider.com/how-the-internet-of-things-market-will-grow-2014-10?r=US&IR=T.

[11] Grant Hernandez et al. "Smart nest thermostat: A smart spy in your home". In: *Black Hat USA* (2014).

[12] Honeywell. *Honeywell Thermostat Evohome.* URL: https://www.amazon.fr/Honeywell-THR992GRT-Thermostat-Evohome/dp/B00MPJF6DU?dpID=41WgEIUxmxL&dpSrc=sims&ie=UTF8&preST=_AC_UL160_SR160%2C160_.

[13] Texas Instruments. *LMx35, LMx35A Precision Temperature Sensors.* URL: http://www.ti.com/lit/ds/symlink/lm235.pdf.

[14] The Garage Lab. *ESP8266 - The Easiest Way to Automate Your Home*. URL: http://thegaragelab.com/esp8266-the-easiest-way-to-automate-your-home/.

[15] Jiakang Lu and Al. "The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes". In: *SenSys '10 Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (2010), pp. 211–224. URL: http://dl.acm.org/citation.cfm?doid=1869983.1870005.

[16] Laura Thomson Luke Welling. *PHP MySQL*. Pearson, 2009, pp. 2,3.

[17] Toni McConnel. *DESIGN East - Microchip USB 8-bit PIC MCUs need no external crystal*. URL: http://www.embedded.com/electronics-products/electronic-product-reviews/mcus-processors-and-socs/4396293/Microchip-USB-8-bit-PIC-MCUs-need-no-external-crystal.

[18] Navigant Research. *Navigant Research Leaderboard Report: Smart Thermostats*. URL: https://www.navigantresearch.com/wp-assets/brochures/LB-STHERM-15-Executive-Summary.pdf.

[19] Margaret Rouse. *Internet of Things (IoT)*. URL: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT.

[20] Rui Santos. *Getting Started with ESP8266 WiFi Transceiver (Review)*. URL: http://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/.

[21] ON Semiconductor. "LB1938FA, Monolithic Digital IC 1ch, Low-saturation Forward/Reverse Motor Driver". In: ().

[22] Espressif Systems. *ESP8266EX Datasheet*. URL: http://download.arduino.org/products/UNOWIFI/0A-ESP8266-Datasheet-EN-v4.3.pdf.

[23] Janani Gopalakrishnan Vikram. *What is the IoT ?* URL: http://electronicsofthings.com/fundamentals-basics-start-101/what-is-iot/.

[24] www.esp8266.com. *ESP8266 enable deep-sleep mode*. URL: http://www.esp8266.com/viewtopic.php?f=32&t=5701&start=30.