REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DE TECHNOLOGIE

DEPARTEMENT D'ELECTRONIQUE

DOMAINE : SCIENCE ET TECHNOLOGIE

FILIERE : ELECTRONIQUE

OPTION : ELECTRONIQUES DES SYSTEMES EMBARQUES

Mémoire présenté pour l'obtention

Du diplôme de Master Académique

Par :

AMRIOU WALID

Intitulé

Conception d'une plateforme embarquée pour la surveillance de malade

Design of an embedded system platform for patient monitoring

Soutenu devant le jury composé de :

O . Guichi	Université de M'sila	Président
A . Benhamadouche	Université de M'sila	Rapporteur
Y . Brik	Université de M'sila	Examinateur

Année universitaire : 2018 /2019

Dedication

To everyone that supported, helped or motivated me in this life, for everyone that believed in me every time.

Amriou Walid

Acknowledgements

I would like to express my thanks to my supervisor Mr. BENHAMADOUCHE Abdelouahab who supported and followed up this work.

I thank the jury president and jury members for agreeing to judge this work.

I would like to thank the managers and all the staff of the Electronics Department of M'sila for the facilities they have granted me to complete this work.

Finally, I thank all those who have contributed directly or indirectly in this project, I would like to thank the engineer Olfa Karoui for helping and providing advice in the software part of this project.

Summary

In this project, we will design from scratch a platform similar to patient monitoring system, but in a way, that makes it lighter, less expensive, portable, more capable of interacting with the surrounding equipment (such as computers and smartphones), and working in difficult conditions, directed to the public, developers, students and medical professionals. In addition, we will build a prototype that is based on this design.

في هذا المشروع ، سنصمم من الصفر نظامًا مشابهًا لنظام مراقبة المريض، ولكن بطريقة تجعله أخف وزناً، أقل تكلفة، محمولًا، أكثر قدرة على التفاعل مع المعدات المحيطة (مثل أجهزة الكمبيوتر والهواتف المحمولة الذكية)، العمل في ظروف صعبة، موجه إلى العامة والمطورين والطلاب والمهنيين الطبيين. بالإضافة إلى ذلك ، سنقوم بإنشاء نموذج أولي يستند إلى هذا التصميم.

Dans ce projet, nous allons concevoir une platforme similaire au système de surveillance des patients, mais d'une manière qui la rende plus légère, moins chère, portable, plus capable d'interagir avec les équipements environnants (tels que les ordinateurs et les smartphones) et de supporter des conditions difficiles, destinées au public, aux développeurs, aux étudiants et aux professionnels de la santé. De plus, nous allons construire un prototype basé sur cette conception.

Nomenclature

°C	Celsius
°F	Fahrenheit
ADC	Analog to Digital Converter
BLE	Bluetooth Low Energy
CPAP	Continuous positive airway pressure
CT scans	Computerized tomography scans
ECG	Electrocardiogram
EHR	Electronic health record
EKG	Electrocardiogram
GATT	Generic Attribute Profile
GND	Ground
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GSR	Galvanic Skin Response
Hb	Hemoglobin without carries oxygen
HbO2	Hemoglobin carries oxygen
HR	heart rate
I2C	Inter-Integrated Circuit
ICT	Information and communications technology
ICU	Intensive care unit
IoT	Internet of things
IR	Infrared radiation
JSON	JavaScript Object Notation
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Microcontroller
MEMS	Micro-electromechanical system
MEMS	Micro Electro Mechanical Systems
MISO	Master In Slave Out
MOSI	Master Out Slave In
MRI	Magnetic resonance imaging
Node.js	Node JavaScript
OTA	Over The Air
PCB	Printed Circuit Board
PET scans	positron emission tomography scans
RAM	Random Access Memory
RGB	red, green, and blue
ROM	Read-Only Memory
RPM	Rotations per minute
RTC	Real time clock
RTD	Resistance temperature detectors
RX	Receive in series communication
SCK	Serial Clock
SD	secure digital
SoC	System on a chip

SPI	Serial Peripheral Interface
SPO2	pulse oximeter oxygen saturation
SS	Slave Select
SSD	Solid-state drive
TFT	Thin Film Transistor
TX	Transmit in series communication
USB	Universal Serial Bus
VCC	source supply
WAN	Wide Area Network
WI-FI	Wireless Fidelity
LAN	Local Area Network
WAN	Wide Area Network
RISC	Reduced Instruction Set Computer
DSP	Digital signal processor
ASIPs	Application Specific Instruction Set Processors

List of Figures

Chapter I Health field

Chapter II Embedded system design

Chapter III General Design

Chapter IV The detailed design

Chapter V Realization

List of Equations

Chapter I Health field

Chapter V Realization

List of Tables

Chapter III General Design

Chapter IV The detailed design

Chapter V Realization

Index

Chapter IV The detailed design

Chapter V Realization

Annexes

General introduction

The development of microelectronics (a branch of material physics) has helped the development of devices and platforms for the public and public services, from the telephone to various devices that represent a common object today. The electronic integration and development teams around the world have contributed to their evolution and scale to the size that can be carried anywhere. This is especially true in the medical field, where microelectronic has contributed to accelerate the integration of medical sensors to small and more efficient devices. In the field of embedded systems, the developer and engineer are interested in building devices and platforms that use the latest technologies for public and private application, either to develop what exists or to create new solutions that solve deeper problems.

In the modern medical field, the quality of medical instruments is the most important point that supports the medical diagnosis, which is the basic rule of treatment. Among these medical devices, patient monitoring systems are the most used in hospitals, medical institutions, and rescue vehicles. However, they are considered to be of high-cost and cumbersome.

In this project, we will design from scratch a platform similar to hospital medical monitor, but in a way that makes it lighter, less expensive, portable, more capable of interacting with the other equipment. Employment and utilization are directed to the public, developers, students and medical professionals. To support our design approach, we will build a real prototype based on simple components and readily available on internet stores.

This manuscript is organized as follows: The first chapter presents the main idea of e-health domain including some associated electronic devices and systems. The second chapter presents our design approach for e-health embedded system. In the third chapter, a general design plan is proposed that gives a global view of the system intends and architecture. The fourth chapter gives more detailed aspects of the system and its implementation. In the last chapter, system prototype is presented, tested and evaluated with several experiments to highlight its effective working and performance.

Finally, the manuscript is completed with overviewing conclusions and providing directions for future work.

CHAPTER I

HEALTH FIELD

Introduction

We dedicate this chapter to introduce some concepts of health domain, e-health, medical sensors, and other related fields. to better insight into e-health issues, some definitions are needed to identify the key points of system design in health fields. Furthermore, we have to present some medical sensors and other systems dedicated to patient monitoring.

1. healthcare domain

Health care means the act of taking preventative or necessary medical procedures to improve a person's well-being. This may be done with surgery, the administering of medicine, or other modifications in a person's lifestyle. These services are typically offered through a health care system made up of hospitals and physicians [1]. On the other hand, the health care industry is one of the biggest industries in the world, and it has a direct effect on the quality of life of society. The health care (some say “healthcare”) is the diagnosis, treatment, and prevention of disease, illness, injury, and other physical and mental impairments in humans. Health care is delivered by practitioners in medicine, chiropractic, dentistry, nursing, pharmacy, allied health, and other care providers [2].

2. eHealth and mHealth

eHealth is an umbrella term that covers a wide range of health and care services delivered through information and communication technologies (ICTs), such as electronic health records (EHRs), health information systems, remote monitoring and distant consultation services (e.g. telehealth, telemedicine, telecare), tools for self-management, and health data analytics. In addition, Mobile Health (mHealth) represents a subset of eHealth, namely the application of mobile technology and applications to provide or use health services, share clinical information and collect data [3].

3. Embedded Systems in Healthcare application

Embedded systems have many applications in healthcare. They are used for monitoring vital signs, for amplifying sounds in electronic stethoscopes, and in nearly every kind of imaging system, including PET (Position Emission Tomography) scans, CT (Computerized Tomography) scans, and MRI (Magnetic Resonance Imaging). Glucose monitors, pacemakers, CPAP (Continuous Positive Airway Pressure) machines, and a variety of biomedical sensors also rely on embedded systems. With biomedical applications, embedded systems allow doctors to remotely monitor patients’ health and make diagnoses and treatment decisions through telemedicine and other remote systems [4].

4. Examples of the association between medical, electronic and embedded system

The most important relation between the medical, electronics and embedded system is the medical sensors, because when the medical sensors evolve, the medical diagnosis evolve and for that, the medical evolve. We focused in this example on the medical sensors because it shows us the power of the electronics and embedded system to possibility to get the information's from the biological human body, just like what we do in this project. We will work to get the information from the existing sensors, however in this example we will look at how some sensors get the information from the body to an information usable to developers like us.

In the first we need to look to the sensors, the sensor is a device that measures the physical quantity (i.e. Heat, light, sound, etc.) into an easily readable signal (voltage, current etc.). It gives accurate readings after calibration [5] . Now, there are two types of sensor, the analog sensor and the digital sensor. Analog sensor senses the external parameters and gives analog voltage as an output. Digital sensor produces discrete, digital, binary values or signals [6] . Today generally the sensors connect with boards like microcontroller by two communication protocols, the I²C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface). These protocols help the developers to use multi sensors in one wire and simplify the communication programming between the components of the system.

Before we start to explaining some medical sensor, we need to tell you that it is important to understand how the sensors will function, and know its limitations because erroneous readings can lead to unnecessary testing, and to know the problem you need to know what happen in the sensor. Sometimes, the problem in the sensor but if you do not know how it works, it can mislead you into thinking that the problem is in your interface, bus, or you code in the microcontroller or SoC.

4.1. Pulse oximeters sensor

The pulse oximeters sensor used to measure a person's oxygen saturation (SO2) or how much of the hemoglobin in blood is carrying oxygen (Figure 1). To understand how the pulse oximeters work we need to know what the oxygen saturation is. The oxygen saturation refers to the extent to which hemoglobin is saturated with oxygen. Hemoglobin is an element in the blood that binds with oxygen to carry it through the bloodstream to the organs, tissues, and cells of the body. Normal oxygen saturation is usually between 96 percent and 98 percent. [7]

- **Operating principle**

To calculate the oxygen saturation, we need to know the number of all the hemoglobin and the number of the hemoglobin without oxygen (deoxygenated hemoglobin or deoxy Hb)

and the number of the hemoglobin with oxygen (oxygenated hemoglobin or oxy Hb). After that, the oxygen saturation simply refers to the percentage of the available hemoglobin that carries oxygen, for example, there are 16 hemoglobin units and none of the 16 have oxygen. The oxygen saturation is therefore 0 %, and if 8 of the 16 Hb have oxygen. The oxygen saturation is therefore 50 %, and when all the Hb have oxygen, the saturation is 100 %. So, oxygen saturation tells you the percentage of the total hemoglobin that is carrying oxygen.

Therefore, the mathematics equation that explain how calculate the oxygen saturation is:

Equation 1 Mathematics equation that explain how calculate the oxygen saturation

$$SqO2 = \frac{HbO2}{(Hb + HbO2)}$$

Where:

HbO2 = Hemoglobin carries oxygen.
Hb = Hemoglobin without carries oxygen.

Figure 1 Pulse oximeters sensor

- **Technical principal**

Technically, pulse oximetry is based on the principle that oxygenated hemoglobin and deoxygenated hemoglobin differentially absorb red and near infrared (IR) light. It is fortuitous that oxygenated hemoglobin and deoxygenated hemoglobin have significant differences in absorption at red and near-IR light because these two wavelengths penetrate tissues well whereas blue, green, yellow, and far-IR light are significantly absorbed by nonvascular tissues and water (Figure 2).

Oxygenated hemoglobin absorbs greater amounts of IR light and lower amounts of red light than does deoxygenated hemoglobin; this is consistent with experience well oxygenated blood with its higher concentrations of oxygenated hemoglobin appears bright red to the eye because it scatters more red light than does deoxygenated hemoglobin. On the other hand,

deoxygenated hemoglobin absorb more red light and appears less red. Exploiting this difference in light absorption properties between oxygenated hemoglobin and deoxygenated hemoglobin, pulse oximeters emit two wavelengths of light, red at 660 nm and near-IR at 940 nm from a pair of small light-emitting diodes located in one arm of the finger probe. The light that is transmitted through the finger is then detected by a photodiode on the opposite arm of the probe; i.e., the relative amount of red and IR light absorbed are used by the pulse oximeter to ultimately determine the proportion of Hb bound to oxygen.

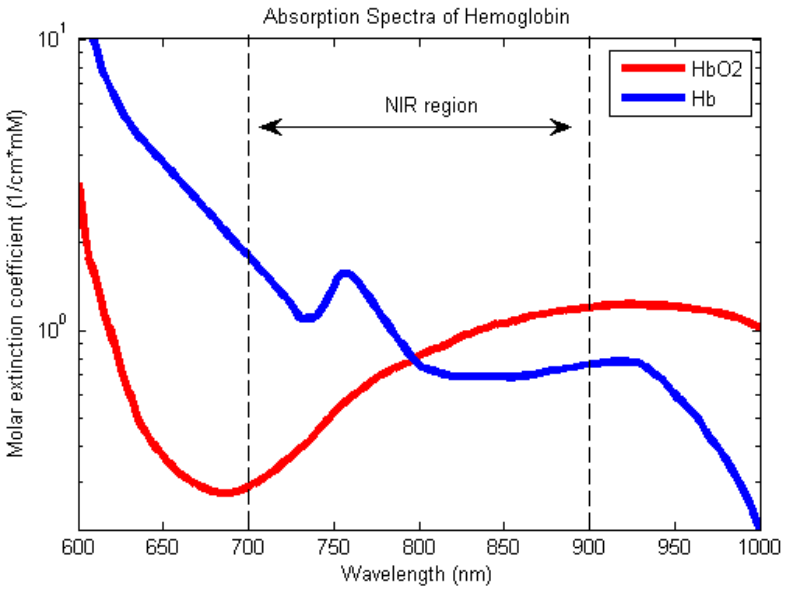

Figure 2 Absorption spectra of oxygenated hemoglobin (HbO2) and deoxygenated hemoglobin (Hb) for red and infrared wavelengths

4.2. Body Position sensor

The patient position sensor monitors five different patient positions (standing/sitting, supine, prone, left and right.). Body position sensor uses a triple axis gyroscope to obtain the patient's position. To understand how the body position sensor work we need to know how the gyroscope sensor work.

Figure 3 Body Position sensor

The gyroscopes are sensor used to measure angular velocity. It works more less the same with an accelerometer but it gives more accurate information for example to know how exactly an object is oriented. While accelerometers are affected by gravity, gyroscope are not and so they make a great complement to each other. They measure angular velocity in units of rotations per minute (RPM), or in degrees per second (°/s). The three axes of rotation are mostly referenced in many texts as roll, pitch, and yaw.

The figure 3 show an example of the body position sensor, it is like belt and has the sensor core in a small box and the sensor send the data by a cable.

In real world, the gyroscope sensor is a MEMS (microelectromechanical system) gyro, and the MEMS or Microelectromechanical systems is the technology of microscopic devices, particularly those with moving parts. In simple, to need know the body position, we just get it from angular.

4.3. Body temperature sensor

The temperature sensor allows you to measure the human body temperature. Normal body temperature varies by person, age, activity, and time of day. The average normal body temperature is generally 37°C (98.6°F), some studies have shown that the "normal" body temperature can have a wide range, from 36.1°C (97°F) to 37.2°C (99°F). [8]

To measure the temperature and because skin temperature cannot directly be correlated with interior body temperature, body temperature measurement is traditionally performed inside a body cavity: orally, rectally, or under the arm. These measured cavity temperatures may vary from the “true” core temperature depending on a number of physiological and environmental effects. An old and traditional device used for body temperature measurement is the mercury thermometer that does not contain sensors. Its drawbacks are slow operation and difficult reading and registration of the result; furthermore, it can easily be broken and poison the environment. However, its application is widespread because it is a low-cost device.

Figure 4 Body temperature sensor

In general, the choice a temperature sensor for a project can be an overwhelming task. The current range of sensors in the market is broader than ever, and it is easy to feel lost if you are not familiar with calibration. In practice, there are three core types are used:

- Thermocouples: A thermocouple uses two metal wires to produce a voltage relative to the temperature present in the junction between them. There are many specialized kinds of thermocouples – they can combine different metals to measure various characteristics and temperature ranges, and produce specialized calibrations.
- Resistance temperature detectors (RTDs): An RTD sensor measures temperature based on the resistance changes in a metal resistor inside. The most popular RTDs, called PT100 sensors, use platinum and have a resistance of 100 ohms at 0°C.
- Thermistors: A thermistor is like an RTD, but contains a ceramic or polymer resistor instead of metal. [10]

However, another technology is vastly used in electronic devices and IC technologies and their application is already widespread, low-cost electronic thermometers with digital displays are replacing most thermometers in a limited temperature range. They generally contain diodes as temperature-sensing elements with a special package design that can assure small thermal capacity and good thermal conductivity to the environment. They have relatively short response times and good visible display units. More complicated versions for clinical use may also contain intelligent functions, such as storage and readout facilities for computer data processing. For embedded electronic, sensors could have an electronic interface to transmit measured data from source to the processing device.

- **Diode temperature sensors (LM34)**

These sensors are made up by using regular PN junction diodes. These are the most inexpensive type of temperature sensors which are competent enough to provide very adequate results if constant and steady excitation current is supplied to them. Also, they need a two point calibration for satisfactory operation. An ordinary semiconductor diode provides a sensibly linear forward biased voltage whose temperature coefficient is around 2.3mV/°C. A typical diode temperature sensor is shown in the figure below.

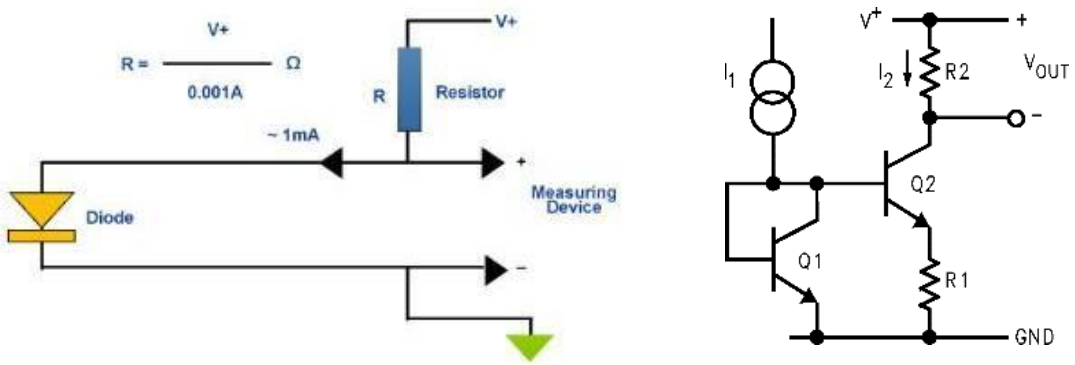

Figure 5 Diode temperature sensor circuit example

4.4. Electrocardiogram Sensor or ECG sensor or EKG sensor

ECG or electrocardiogram is a measure of how the electrical activity of the heart changes over time as action potentials propagate throughout the heart during each cardiac cycle. [11]

The electrical currents that are generated spread not only within the heart, but also throughout the body. This electrical activity generated by the heart can be measured by an array of electrodes placed on the body surface. A "typical" ECG tracing associated with heart activity is shown in Figure 6 . The different waves that comprise the ECG represent the sequence of depolarization and repolarization of the atria and ventricles.

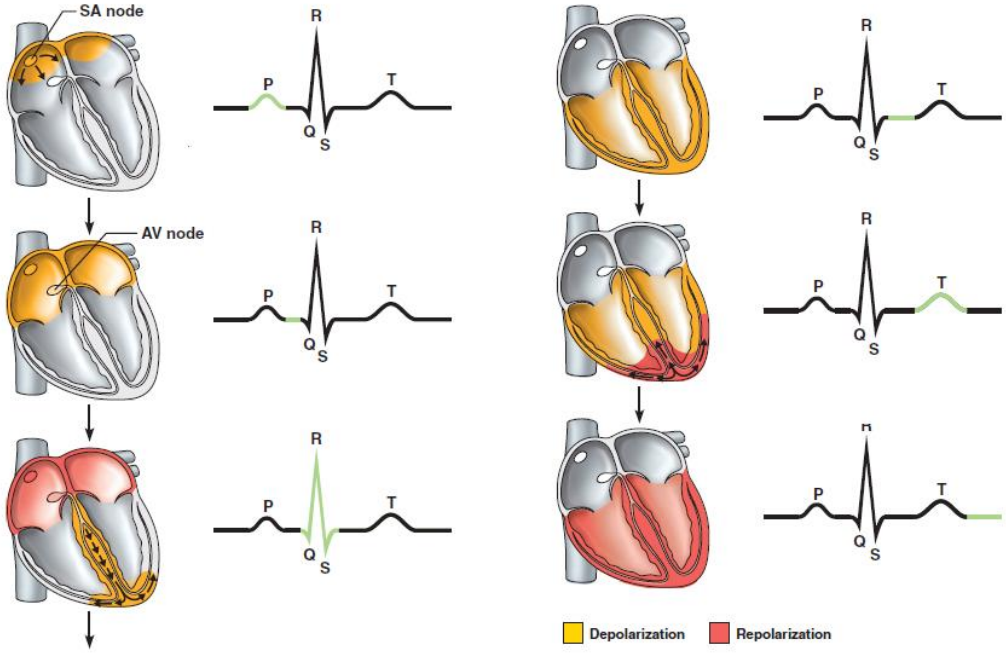

Figure 6. The sequence of depolarization and repolarization of the heart related to the deflection waves of an ECG tracing.

The ECG is recorded at a speed of 25 mm/sec (5 large squares/sec), and the voltages are calibrated so that 1 mV = 10 mm (2 large squares) in the vertical direction. Therefore, each small 1-mm square represents 0.04 sec (40 msec) in time and 0.10 mV in voltage. Because the recording speed is standardized, one can calculate the heart rate from the intervals between different waves.

ECG records the electrical activity generated by heart muscle depolarizations, the electricity amount is in fact very small, it can be picked up reliably with ECG electrodes attached to the skin (in microvolts, or uV). Therefore, in ECG graph each small 1-mm square represents 0.04 sec (40 msec) in time and 0.10 mV in voltage. Because the recording speed is standardized, one can calculate the heart rate from the intervals between different waves.

The full ECG setup comprises at least four electrodes which are placed on the chest or at the four extremities according to standard nomenclature (RA = right arm; LA = left arm; RL = right leg; LL = left leg). Of course, variations of this setup exist in order to allow more flexible and less intrusive recordings, for example, by attaching the electrodes to the forearms and legs. ECG electrodes are typically wet sensors, requiring the use of a conductive gel to increase conductivity between skin and electrodes. [12]

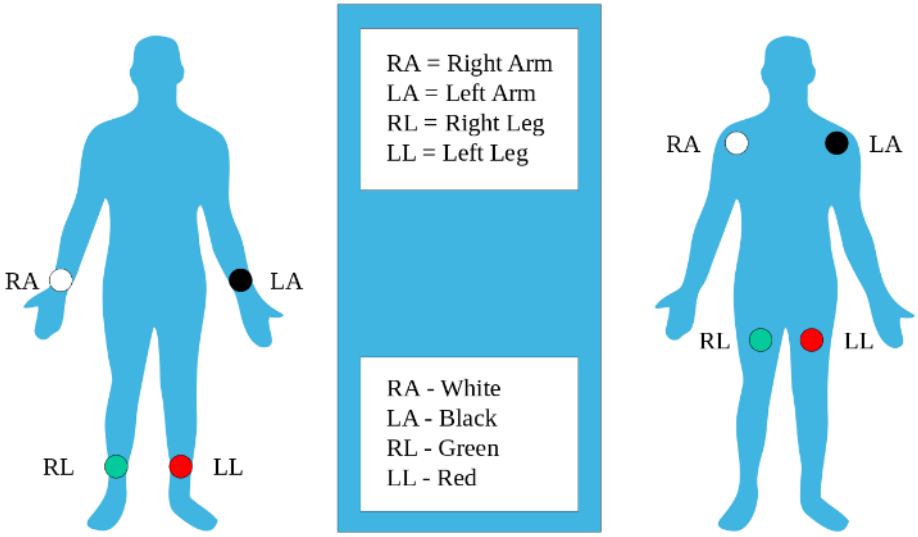

Figure 7 where placed the ECG electrodes

Different types of ECG sensor exist, in the following figure we resume some types that are widely used in medical embedded systems.

Figure 8. ECG sensing devices

5. Patient monitoring system

Patient monitoring system is the system in which different body parameters of patient are monitored and observed. An existing example of such system is an ICU (intensive care unit), where sensors are attached to the patient body. The results are displayed on respective patient monitoring screen and are observed by the doctors [5].

Patient monitoring system involve repeated or continuous observations or measurements of patients, their physiological functions, and the function of the life support equipment. for the purpose of guiding management decisions, including when to make therapeutic interventions, and assessment of those interventions [13].

Figure 9 MULTI-PARAMETER ECG MONITOR / ETCO2 / SPO2 / NIBP by medical Expo

We need to look inside an example of the patient monitoring system, this a block diagram of health care monitoring system (Is the same to the patient monitoring system) [14]:

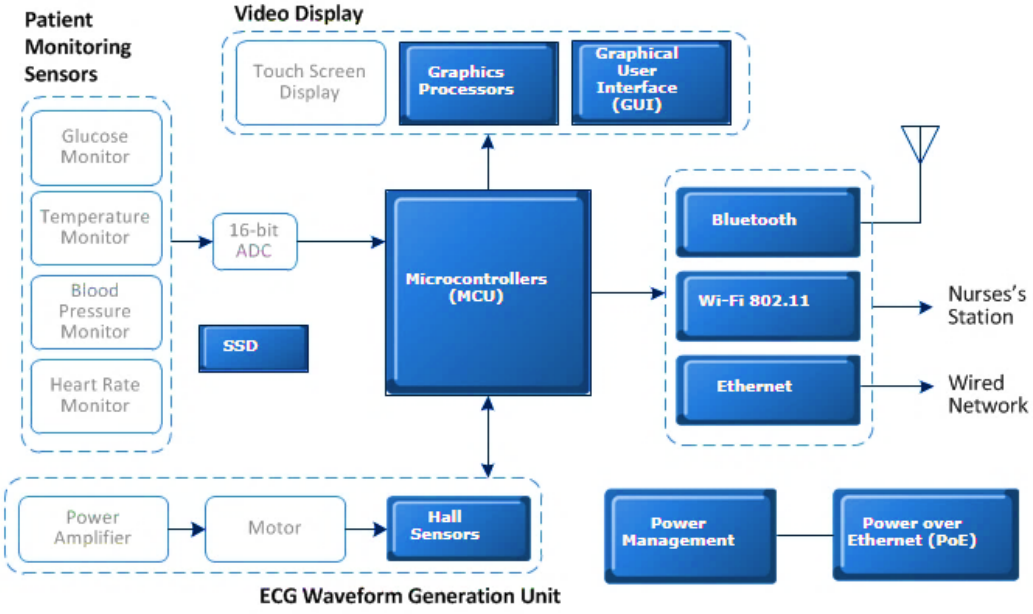

Figure 10 Example of Patient Monitoring System Block Diagram

In this diagram of patient monitoring system, we see that this system is a combination of patient monitoring sensors, video display, ECG waveform generation unit, microcontrollers, SSD storage, power management, Bluetooth, Wi-Fi, Ethernet.

The aim our project is to conceive a similar system, but with taking attention to simplify the system utilization, to reduce the cost of production and to reduce the size of the package.

6. Similar examples of patient monitoring system

Manny similar systems to patient monitoring systems are available around the world. Some of these systems are intended to few medical sensors or intended to a specific disease. Other systems are more complicated and this technical complexity leads to additional costs. Therefore, those systems are not destined for developing countries, such as African countries. The emerging markets customers need high quality and low-cost equipment with the capability to operate in hostile environments and particular situations.

- **MySignals**

MySignals is one of the best similar examples of our project; it is a development platform for medical devices and eHealth applications. You can use MySignals to develop your eHealth web applications or even to add your own sensors to build new medical devices. In 2019, MySignals allows you to measure more than 20 biometric parameters such as pulse, breath rate, oxygen in blood, electrocardiogram signals, blood pressure, muscle electromyography signals, glucose levels, galvanic skin response, lung capacity, snore waves, patient position, airflow and body scale parameters (weight, bone mass, body fat, muscle mass, body water, visceral fat, Basal Metabolic Rate and Body Mass Index).

These broad sensing portfolios makes MySignals the most complete eHealth platform in the market. [14]

Figure 11 MySignals platform

- **Nellcor PM10N Portable SPO_2 Patient Monitoring System**

The Nellcor PM10N Portable SPO2 Patient Monitoring System is handheld patient monitoring system that has SPO2 sensor, screen and battery. The Nellcor portable SPO2 patient monitoring system, PM10N, is a convenient, handheld monitor that is ideal for spot-checks or continuous monitoring in various healthcare and home use settings. Its ergonomic shape and simple design make it intuitive to use and simple to operate.

Figure 12 Nellcor PM10N Portable SpO_2

Conclusion

In this chapter, we presented some concepts involving essentially health and e-health domains. Also, we give a general view of how health needs to evolve with electronic systems, especially embedded system.

We gave some examples of systems that involve electronic sensors and embedded electronics and informatics, these examples can help as to build our own patients monitoring systems to observe and supervise patient health indicators. The deal is how to design a health platform with valuable benefits at low budgets and using a specific system design methodology.

CHAPTER II

EMBEDDED SYSTEM DESIGN

Introduction

This chapter introduces some concepts in the embedded system and system design. The coverage of this chapter begins with giving some definitions and characteristics of an embedded system. Then, the basic concepts of design methodology are introduced, including an overview of the most common steps in embedded system design flow, particularly for medical purpose.

1. What is an embedded system?

There are many definitions of the embedded system, such as: an embedded system is a system with an electronic hardware part that has a software part, it is programmable or non-programmable depending on the application, it is defined as a way of working, organizing, performing single or multiple tasks according to a set of rules, and all the units assemble and work together according to the program [1].

On others definitions, an embedded system is "a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function" [2]. On the other hand, an embedded system can be defined as an electronic device that has a central component that performs computational tasks, it is designed for specific and limited functionality, and it is implemented as a component of an electrical or mechanical system [3].

In this work, an embedded system is considered as a system combination of a hardware electronic part and a firmware part, and eventually a software application is needed for computer and smartphone supervising and control.

This definition can be explained through the figure 1, where a possible organization of the embedded system is illustrated. The embedded system is constructed around a CPU, it has a variety of interfaces that enable to measure, manipulate, test, diagnosis, and interact with human, object and external environment.

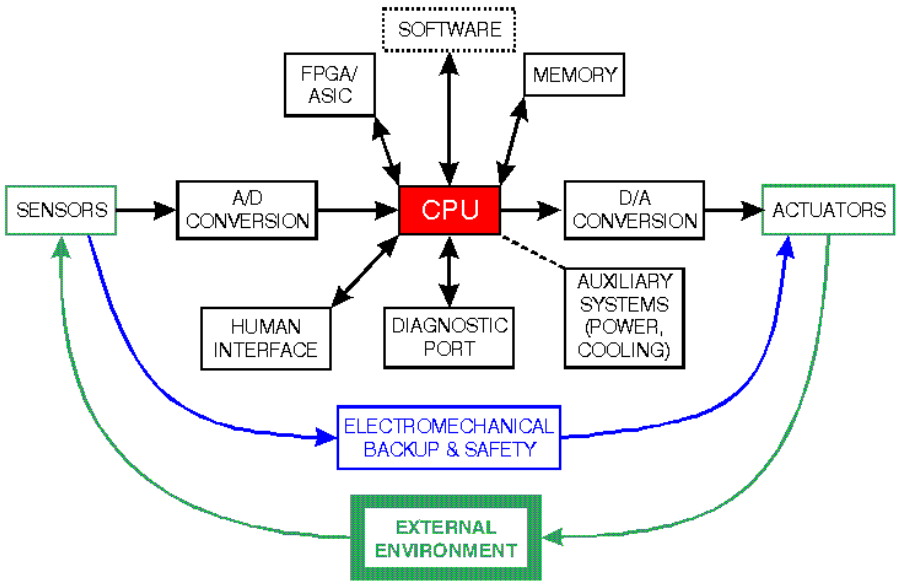

Figure 1 Possible organization of an embedded system.

2. Characteristics of an Embedded System

The characteristic is a distinguishing feature or attribute of an item, person, phenomenon, etc. so, when we want to point something out to its features, we will say he has blue eyes or black hair or, etc. The characteristics of an embedded system will give us the points in a system to say this is an embedded system.

- Clear function: An embedded system usually executes only one program, repeatedly. Even if we talk about use an operating system in the system, the operating system in an embedded system need to know every input and the output of the system, and for that when we see outside the system, the operation system is like one program repeat.

- Tightly constrained: All computing systems have constraints on design metrics, but those on embedded systems can be especially tight. A design metric is a measure of an implementation's features, such as cost, size, performance, and power. Embedded systems often must cost just a few dollars, must be sized to fit on a single chip, must perform fast enough to process data in real-time, and must consume minimum power to extend battery life or prevent the necessity of a cooling fan.

- Reactive and real-time: Many embedded systems must continually react to changes in the system's environment, and must compute certain results in real time immediately. [4]

3. Embedded System Classification

There are many classifications of embedded systems and it can be classified into different type based on performance, functional requirements and performance of the microcontroller and more (Figure 2).

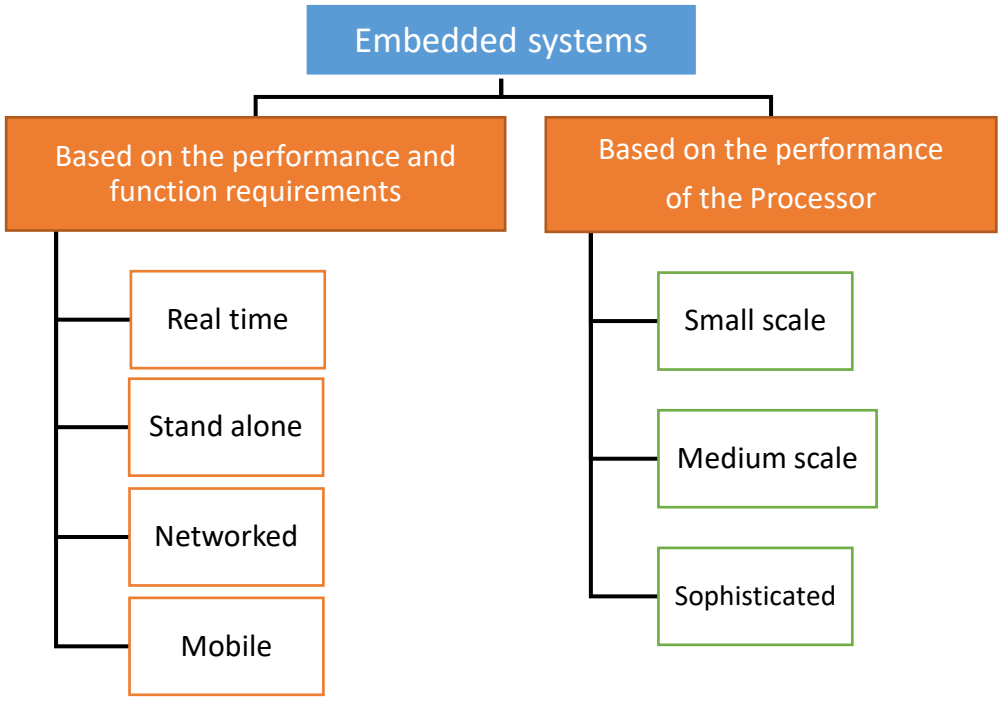

Figure 2 Diagram of classifications of embedded systems

- **Real Time Embedded System:** It is a system that works within strict time constraints and provides a worst-case time estimate for critical situations [5]. It follows the time deadlines for completion of a task. In addition, it classified into two types such as soft and hard real time systems.

- **Stand Alone Embedded Systems:** Standalone embedded systems do not require a host system like a computer, it works by itself. It takes the input from the input ports either analog or digital and processes, calculates and converts the data and gives the resulting data through the connected device-Which either controls, drives or displays of the connected devices. Examples for the stand alone embedded systems are MP3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

- **Networked Embedded Systems:** These types of embedded systems are related to a network to access the resources. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless.

- **Mobile Embedded Systems:** Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and limitation of memory.

- **Small Scale Embedded Systems:** These types of embedded systems are designed with a single 8 or 16-bit microcontroller, which may even be activated by a battery. For developing embedded software for small-scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE).

- **Medium Scale Embedded Systems:** These types of embedded systems design with a single or 16 or 32 bit microcontroller, RISCs or DSPs. These types of embedded systems have both hardware and software complexities. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, JAVA, Visual C++, and RTOS, Debugger, source code engineering tool, simulator and IDE.

- **Sophisticated Embedded Systems:** These types of embedded systems have enormous hardware and software complexities, which may need ASIPs, IPs, PLAs, scalable or configurable processors. They are used for cutting-edge applications that need hardware and software Co-design and components, which have to assemble in the final system. [6]

These classifications can guide us at the analyzes phase of the purposes and the principle idea of the system.

when we know the classification of the system we know more than the global features, because any classification has specification features, the specification features are constraints in the design of the system. When the system is a real time embedded system, time is the specification feature of the system, and when we develop on an example the firmware of the microcontroller, we need to focus in the time.

4. Design flow of an Embedded system

In many design projects of mixed electronic and information devices, managing person, plan or solution is a complicated task. In several projects we have done before, we used instinctively steps to realize promptly and correctly a new design. We assume in this project to follow a methodological strategy to enhance our productivity and to achieve a reliable product, and this last point is very important in embedded systems that are dedicated to healthcare applications.

We note that, this method works in parallel, so when we see the diagram without reading this sentence we would think that the method is in serial, but in reality you can change and develop the parts of the method, for example sometimes some requirements cannot develop as you put it in the specified design so for that you can make a feedback and change the requirements where the problem and develop the design where can the problem affect.

In this flow we start by an idea, after that the requirement analysis will make the specification design, the specification design will definite the different functional tasks and design constraints, after that we will make the papers of the specifications requirements, we can simply say that we define system specifications based on the requirement analysis.

The next section is the architecture and function design. In this section, we will transfer the specifications requirements to technical methods like architectures, algorithms, etc. we will definite the functions intern, the protocols, the operations, data flow, etc.

The next section is the detailed design and in this section, we will transfer those solutions of the architecture and function design to hardware's or circuits electronics, firmware's or codes intern in the circuits, and software's if the hardware need software in computers or smart device like mobile, we do this in the first transfer the architectures and the technical methods to deep technical solution like definite the communication norms, the parts electronics, etc.

Finally, we will realize the system. In the first, we divide the system to small systems, this method can help to develop the system carefully and this method called partial realization, after we confirm that all the parts or subsystems work great we integrate it in on system, we test this new system, and if all the system work fin we will make the PCB, 3D case, etc.

In best practise, you should design prototype before making the full version of the system, we will talk about that in next chapters.

The diagram of the flow of design :

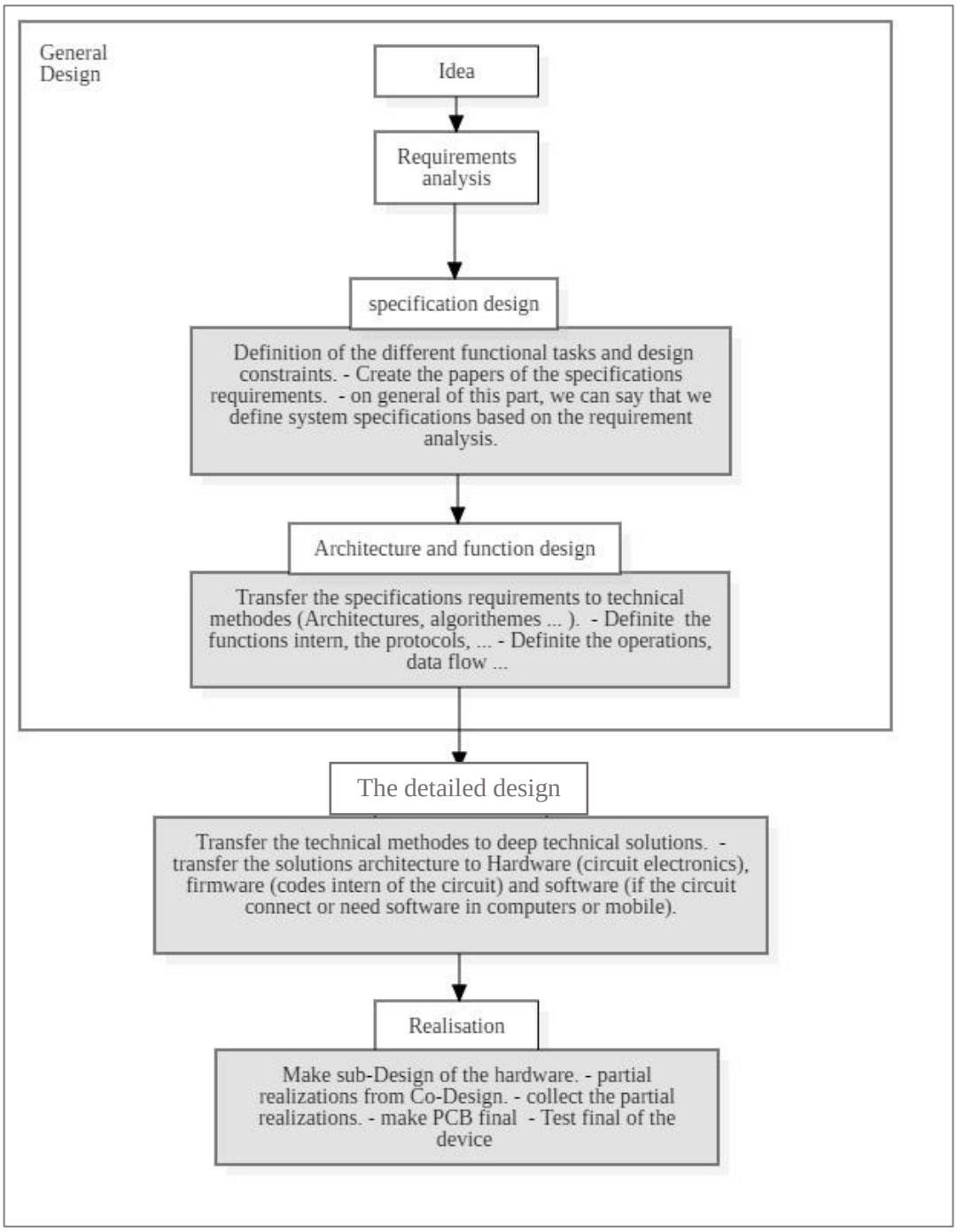

Figure 3 The diagram of the flow of design an embedded system

Conclusion

In this chapter, we give a rapid overview of embedded system that include some definitions and features. Then, and assuming that a good design methodology is necessary for a specific system design, we introduce a design flow to be a mainstay of all development steps to carry out a good product. and the components of the diagram of the flow of design is the same the names of the next chapters. In next chapter, we will start directly to the design of the system.

CHAPTER III

GENERAL DESIGN

Introduction

In this chapter, the general design is presented. As declared in the last chapter the general design is about the specification, architectural design, functional design, it is the first stage in the project design flow which represent a system-level design.

To overcome all design issues, it is necessary to have a comprehensive overview, starting with a preliminary study including a problem identification and system analysis, and then make a global architecture witch permit to identify future functions and components of the system.

1. System analysis

The aim of this work is to conceive an embedded system for medical purpose. This device provides continuous monitoring at any time to patients and the injured. The project aims to provide a device easy to use, portable, interacts with mobile devices, computers, and cloud. The device can become the first layer of e-Health infrastructure (or telemedicine).

The user must place the sensors in the device, and then he can take the recordings, after that he can display, send or store the data. The device must be able to work in difficult conditions like relief operations or wars. We decided to call this project **Medical Signals Box**.

Another objective of this project is to provide a modular platform for developers and students.

2. The general concept

This is a sample mock-up of the operations that will be available to the end-user of the system.

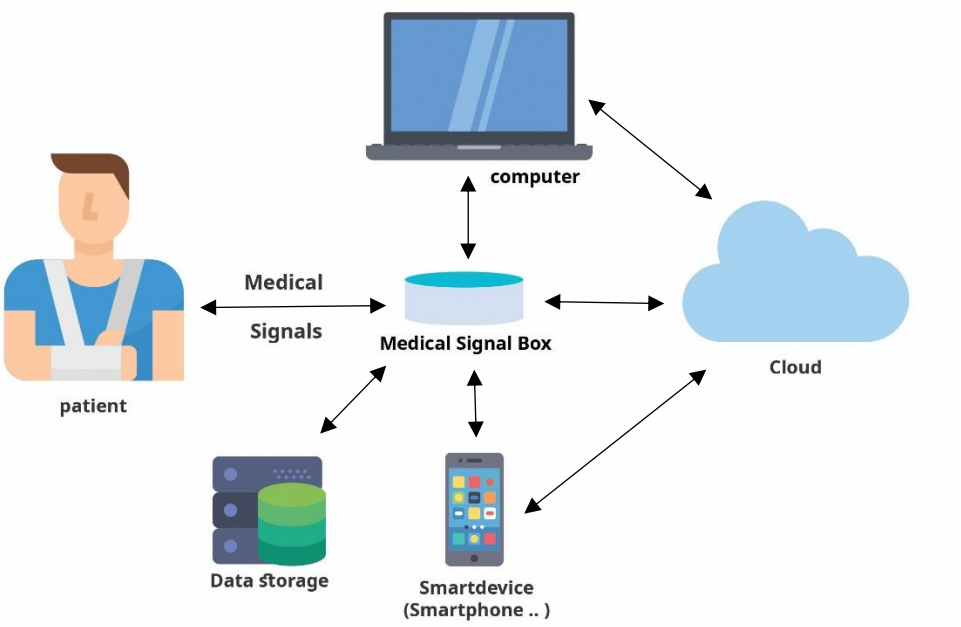

Figure 1 Mock-up of the operations that will be available to the end-user of the system

From this mock-up, we can see that the principle function of the system is to collect from the patient the necessary medical signals. The patient or any medical personnel can use the system to get these medical signs. Then, signal can be displayed in embedded LED display, saved in local memory disk, or transmitted to computer, smart devices and cloud service.

We connect the device directly to the cloud services via internet but we do not focused in this part, because when we do that we start talk about internet of things (IoT), and IoT is the interconnection via the internet of embedded system devices, enabling them to send or receive data.

3. Specifications design

This section defines the different functions, tasks, design constraint, and determine the specification requirements of the main project. In our project, the system is composed of a hardware part that embed a firmware part, and a software application for computer and smartphone.

3.1. Functions Requirements

The functional requirements describe what user can do with the system, whether the hardware or the software of the system.

A- Hardware function requirements

The hardware of the system should allow users to:

- Measure their biomedical signals from biomedical sensors.
- Save the data in a portable storage.
- Visualize the recorded data in a small screen.
- Send the recorded data to computer or smartphone.

B- Software function requirements

The software application which is installed in a computer or a smartphone should allow users to:

- Receive the data from the hardware by wire or wireless communication.
- Visualize the received data.
- Save the data received in an internal database.
- Send the data received to cloud services.
- Visualize the historic of the data received.
- Search for the data by the name of the patient.
- Set up connection parameters for communication with the hardware.

3.2.Performance requirements

This section defined how well the system performs certain functions under specific conditions:

A- Hardware performance requirements

The hardware of the system should

- Start-up in few milliseconds.
- Allow users to use a battery to power it.
- Consume less energy as possible.
- Warning the user when the battery is low.
- Processing the medical signals as fast as possible.

B- Software performance requirements

The software of the system should

- Be able to work in low performance computers and smartphone.
- Start-up as fast as possible.

3.3. Customer requirements

This section defined in detail customer requirements for each component of the system:

- **Language:** All documents, text in the screen of the device, text in the software application should be in English and in the local language where the device is used.
- **Color:** The color of the device should flow the colors of the Logo of the project.
- **Size:** The device should be placed in small emergency bag.
- **The use:** The system should have simple user interface.
- **Documents:** Help documents are needed to describe how customer can use the device and how the device work and an information about the biomedical sensors.

3.4. Packaging requirements (Assembly)

This section defined requirements on how the system should assembled:

- **Battery:** the battery should be removable, packaged separately and must be easily replaced by the user.
- **Sensors:** Sensors should be removable, some sensors should just the wire removable and the core of the sensor will be integrated in the main package of the device.

3.5. Safety requirements

- **Electricity:** The input power of the system should be low and safe.
- **Water:** Moderately water-resistant.

3.6.Maintenance and support requirements

- User manual provided.
- User removable batteries will used.
- Put a physical port for update firmware or automatic update from the network (OTA).

3.7. Design requirements (packaging)

- **Case of the device:** The basic design of the case of the device is small cylindrical:

Figure 2 An abstracted form for the device packaging

- **Connectors:** The connectors of the sensors to the device should be similar to Jack connectors, because the majority of people know this type of connectors and it is easy to use.

4. The architecture and functional design

After the first stage of design which helps the designer to define the problem and to outline the desired functionalities, and to identify the global design constraints. However, the present section defines the different functions of the system, the interaction between the user and the system, and the structure of the system. For those purposes, we use UML modeling language to define three different system models:

- Use case diagram: which describe system functionality from the point of view of the user.
- Structure diagram: which describe the structure of the system in terms of objects, attributes, associations, and operations.
- Sequence diagram: which describe the internal behavior and internal activity of the system.

4.1. Use cases diagram

This section establishes the workflows associated with each of the different relevant use cases of the system. Every one of the use cases answer the question “the user can ...” This use case is for the full version. The specific purpose of the use case diagram is to gather system requirements and actors. It specifies the events of a system and their flows. However, use case diagram never describes how they are implemented.

A- Hardware

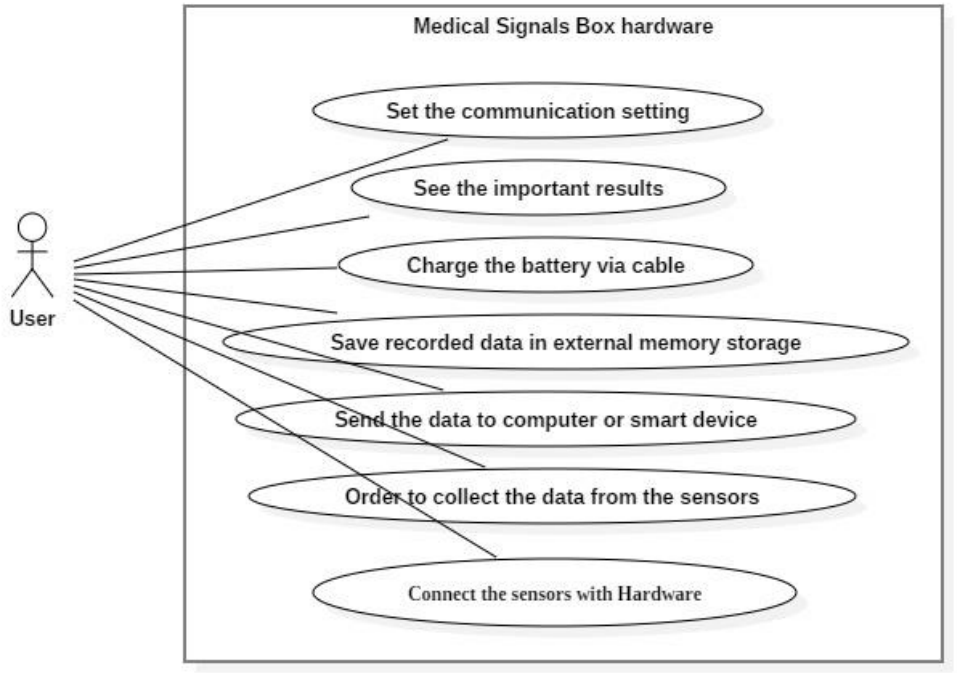

Figure 3 Use cases hardware

B- Software

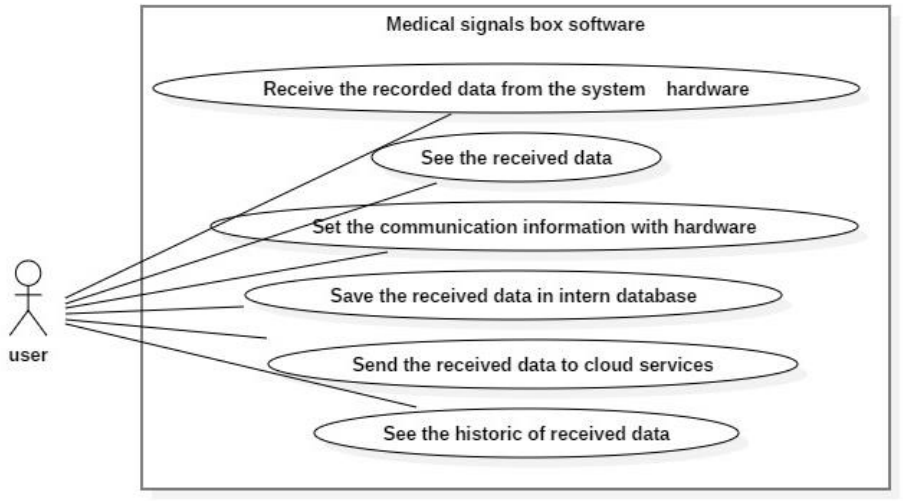

Figure 4 Use cases software

4.2.Sequence diagram

This section shows object interactions arranged in time sequence. We use it because it shows the interaction logic between the objects in the system in the time order that the interactions take place.

A. Hardware

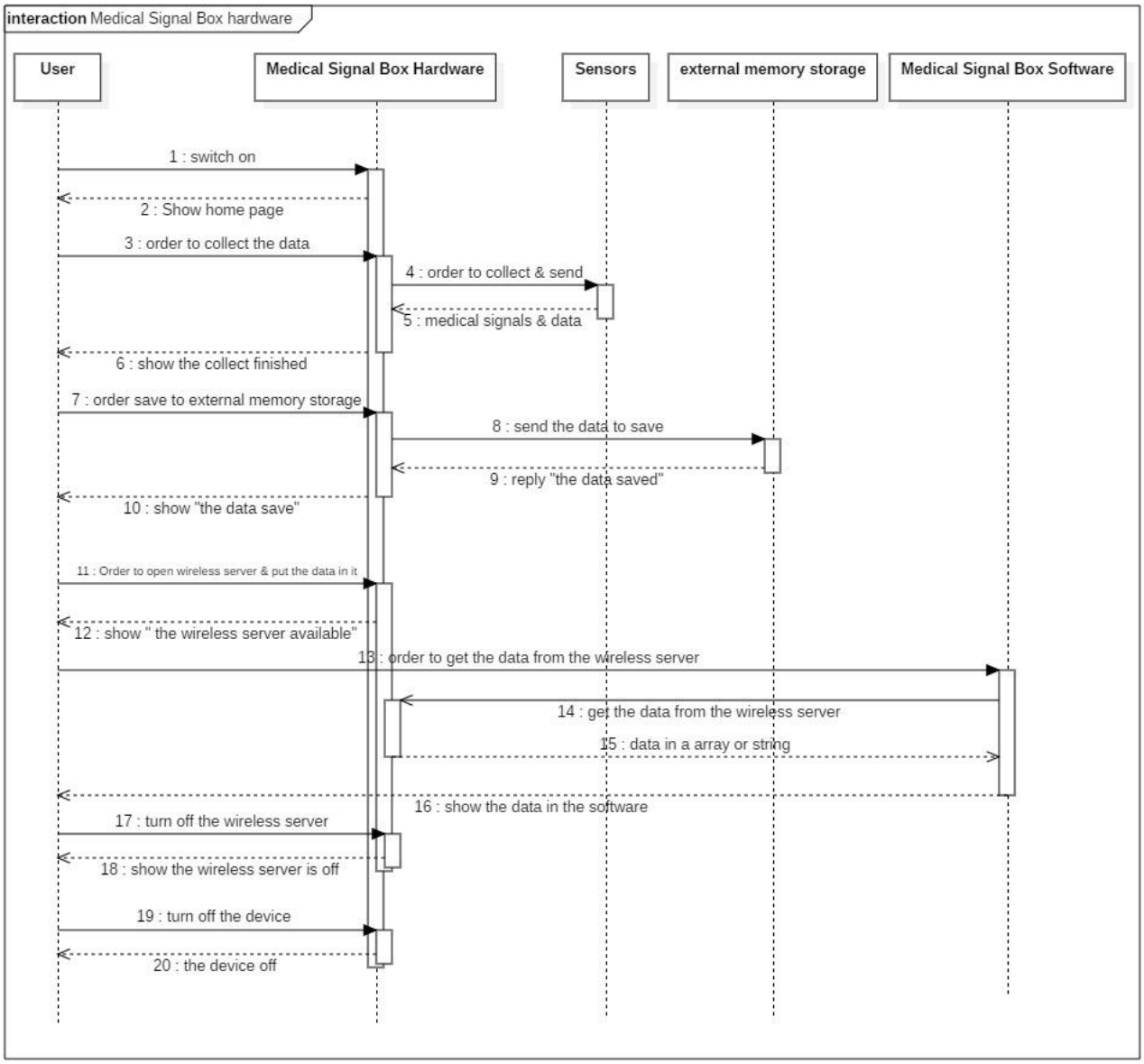

Figure 5 Sequence diagram hardware

B. software

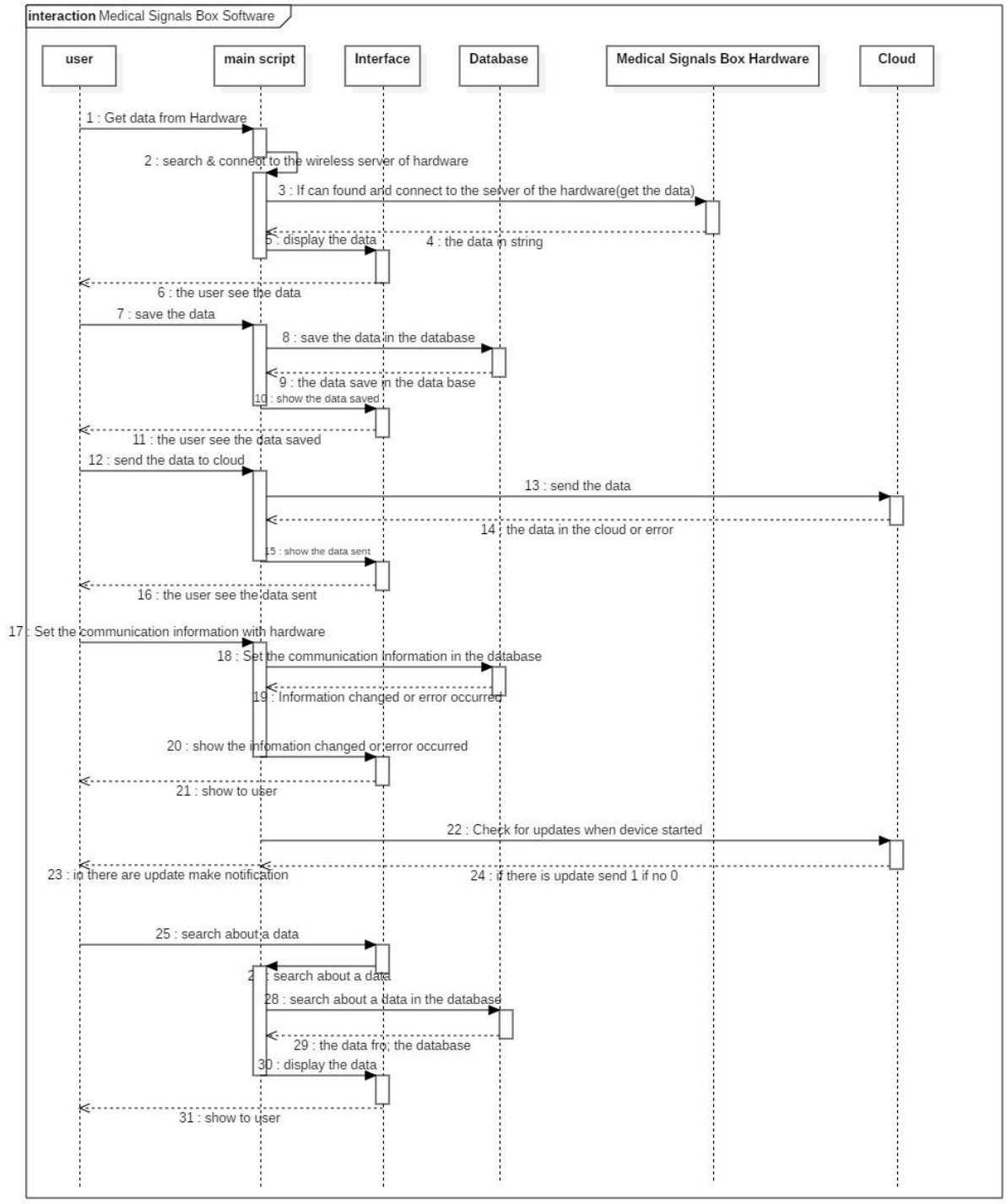

Figure 6 Sequence diagram software

4.3. Block diagram of the system

This section shows the diagram of the system in which the principal parts or functions represented by blocks connected by lines that show the relationships of the blocks.

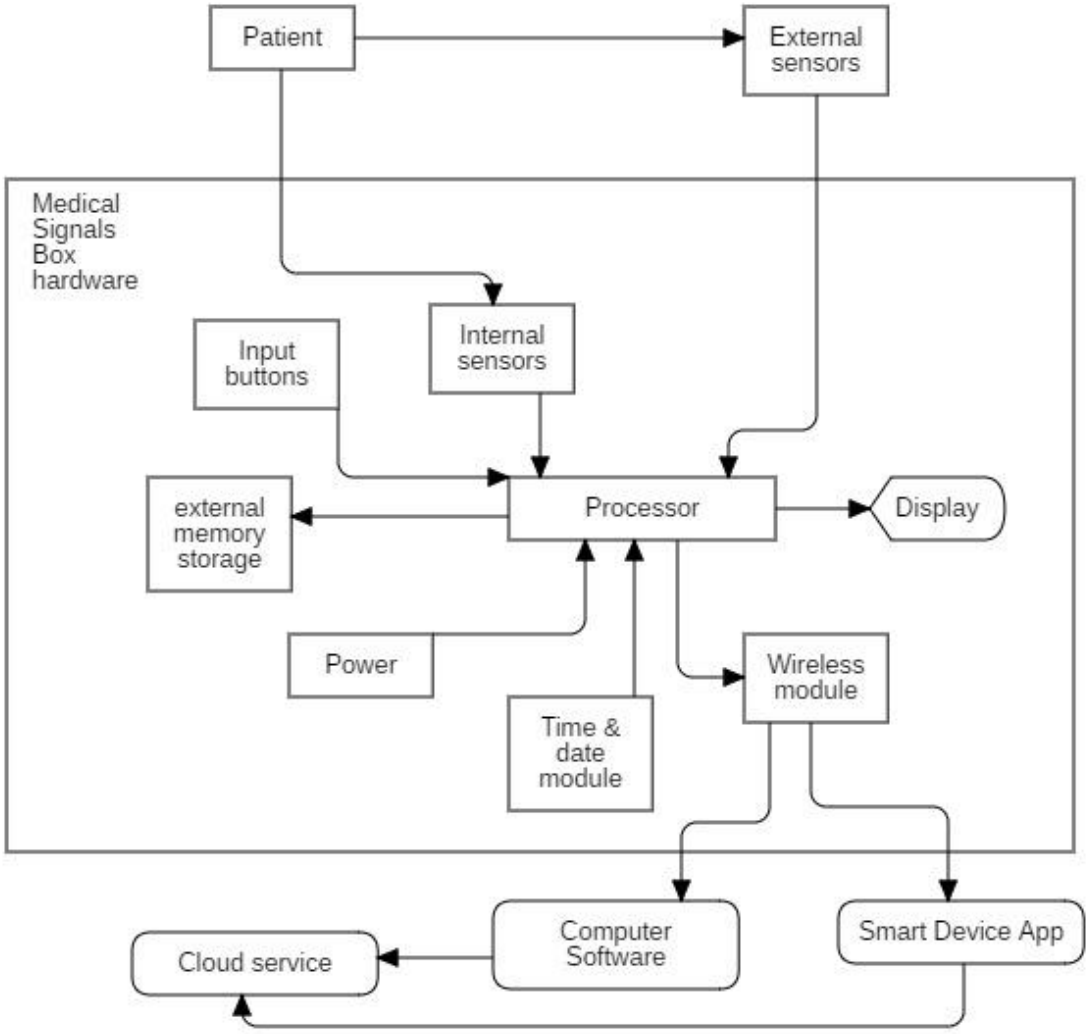

Figure 7 Block diagram of the system

In this block diagram, the processor can be a microcontroller (like Arduino, ESP8266 …) or a processor (based ARM e.g.) or FPGA … This choice comes from what the system needs in term of performances and flexibility.

As illustrated in figure 7, the propose architecture allows the use of two kinds of medical sensors, external and internal sensors, some sensors can be integrated on the device hardware, and other sensors such as the patient position sensor or body position sensor should be in the patient, and for that, the core of electronic circuit should be external to the main circuit of the device.

The device gets the orders from the user by Input buttons. The user shows the data and the information in the display; he can save the data in external memory storage. The time & date module is for getting the current time and date and it is has a small battery to remember the time and date when there is no power in the circuit. In addition, the processor sends the data to computers and smart devices in wireless-by-wireless module.

4.4. Architecture diagram of the system

This section shows the diagram of the system in which the principal parts or functions are represented by blocks connected by lines, these lines show the relationships between the internal blocks, it is like block diagram but architecture diagram tells us " how blocks are connected" like it shows data lines ,address lines which connect different parts, so it tells what is happening inside the structure. e.g., how data is flowing etc.

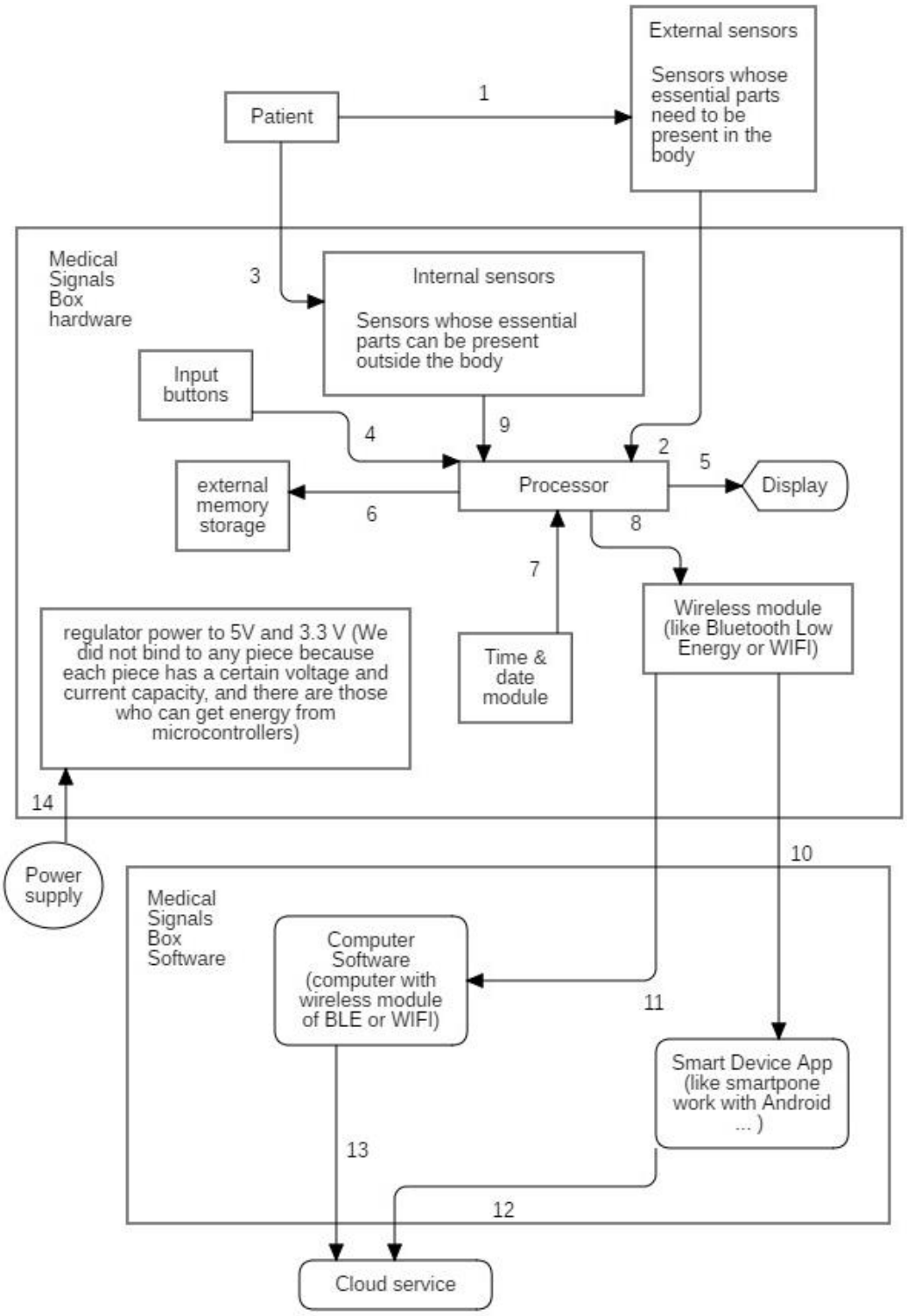

Figure 8 Architecture diagram of the system

4.5. Data Flow of the Architecture diagram of the system

In this section we will detail the dataflows between the components of the system, this description is based on the architecture diagram.Table 1 summarizes the dataflows between the different block of the architecture, with each connector number is given a brief description.

Table 1 table of Data Flow of the Architecture diagram of the system

Line	Data element	Descriptions
1,3	Physical signals	An X number of variable of signals coming in from the body
2,9	Medical signals	An X number of variable of medical signals readings coming in from the sensors
4	Voltage signals	A logic voltage signal, if on is 1 or off is 0, and 1 is 5 v or 3.3 v and 0 is 0 v
5	Display data	A packet of information indicating what screens to display, and what data to populate the fields with.
6	Data information packet	A packet of information identifying the location and the data where save it in the external data storage.
14	Power	A voltage or current which conveys power, the voltage or current should be compatible to the next component.
7	Time data	A packet of information identifying the current date and time.
8	Communication information packet	A packet of information containing the information of the local wireless server or the data that will send to the devise via the wireless local server.
10, 11	Wireless data	A packet of information containing the data that send by the hardware of the system to devices have the software of the system.
12, 13	Cloud information packet	A pocket of information containing the data and information of the cloud, this pocket travail via the internet.

4.6. General Algorithm of the firmware

The firmware is the program that manage the hardware device, this section shows the basic algorithm of firmware.

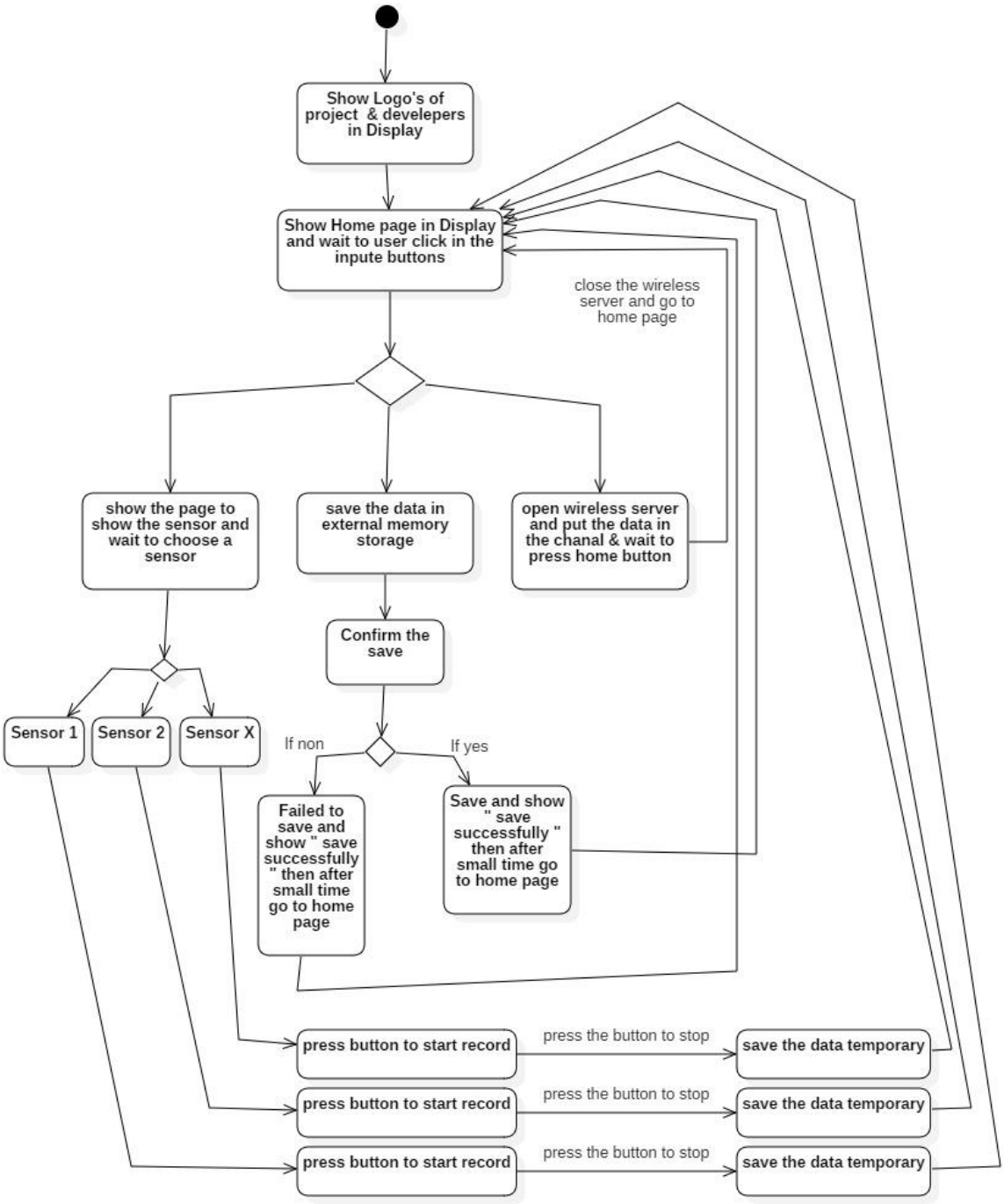

Figure 9 General Algorithm of the firmware

4.7. General Algorithm of the Software

This section shows the basic algorithm for software application to be run on computers or mobile devices. All operations of the algorithm finish by a final circle, this mean that the software operation is not stopped but there are further operations to be executed next.

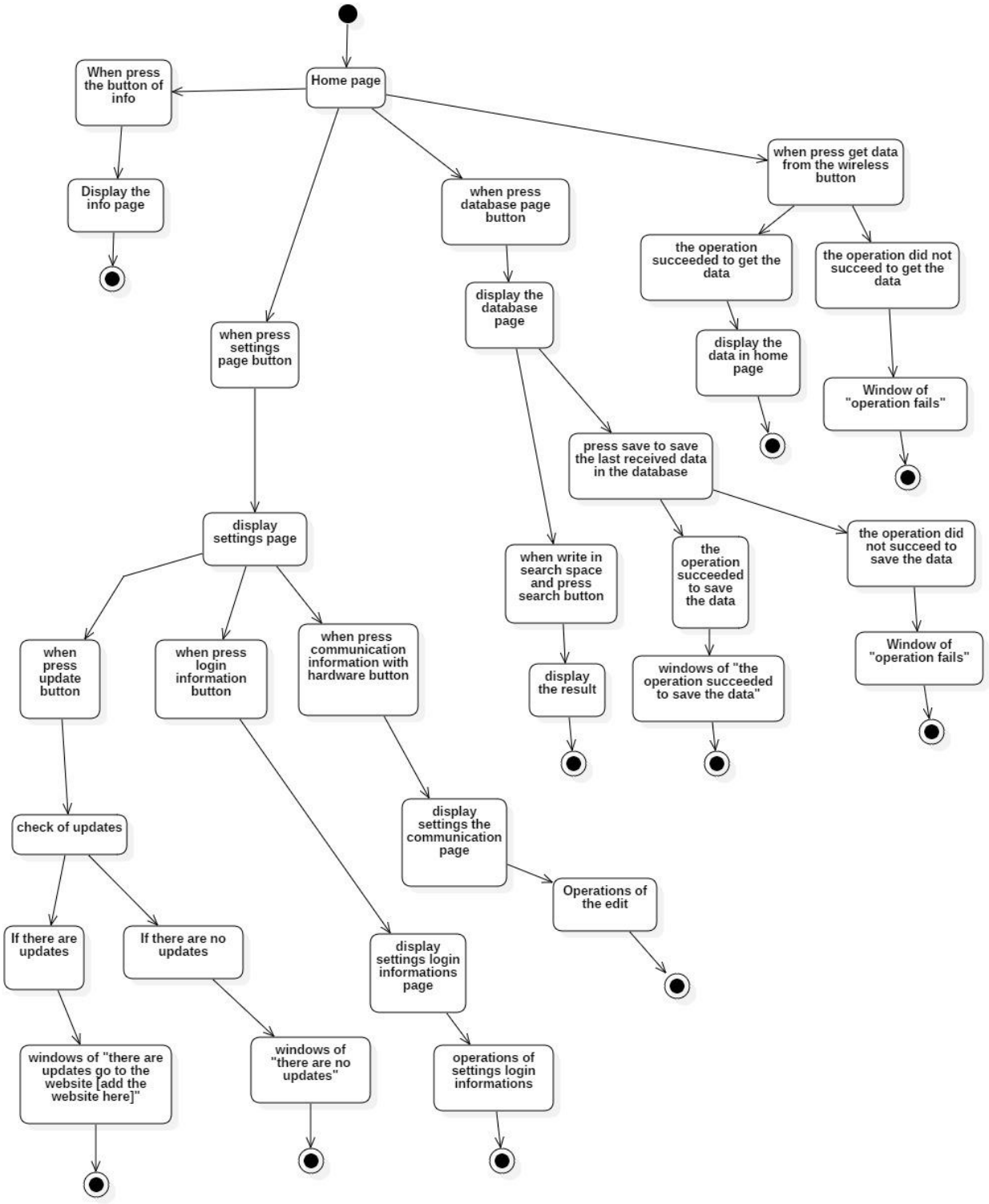

Figure 10 General Algorithm of Software

Conclusion

In this chapter, we get the general ideas about the system such as the main architecture of the project, the main algorithms of the firmware and the software of the project.

Moreover, many UML diagrams are presented which helps the designer to understand, analyze, and master the whole system operation. Also, this helps us to maintain or to make easily modifications in future versions of the system.

In the next chapter, more details are given on how the hardware implementation is considered and how components are chosen.

CHAPTER IV

THE DETAILED DESIGN

Introduction

This chapter focuses on the presentation of a detailed design of the system. In this second part of the design process, it is important to go deeper in the definition of the architecture, the hardware and software functionalities on one side, and on the other side to describe exhaustively the architecture and the involved electronic components.

1. Technical hardware architecture diagram

This section shows technical hardware diagram of the architecture that is mentioned in chapter III, or the transfer of the architecture diagram to a technical hardware architecture diagram, that mean the last architecture diagram is just a general look, and the present diagram is more detailed and involve technical hardware specification.

A. Medical signals measurement

The architecture is design to support 10 kinds of sensors as listed in table 1: body position, body temperature, electromyography (EMG), electrocardiography (ECG), airflow, galvanic skin response (GSR), blood pressure, pulse oximeter, glucometer and spirometer. When we talk technically or by a knowledge combined between medicine and hardware electronics, these sensors are divided into two parts: External sensors and internal sensors.

The external sensors are sensors whose essential parts needs or the core electronics circuit need to be present on the body like body position sensor. However, the internal sensors whose essential parts can be present outside the body like ECG sensor. The division of the sensors that we made will help us to design the electronic circuit (Printed circuit board or PCB) and the hardware box.

Table 1 The external and internal medical sensors in the system

External sensors	Internal sensors
Body position	Electromyography (EMG)
Body temperature	Electrocardiography (ECG)
Airflow	Galvanic skin response (GSR)
blood pressure	
Pulse oximeter	
Glucometer	
Spirometer	

B. Communication

In the part of the wireless module, the best choice between BLE or WIFI is not independent to other components, and for that the best choice is BLE, because when we talk about the energy consumption between WIFI and BLE, BLE is developed to be more low energy consuming.

For a connected device (or IoT device) or project like medical signals box, the voltage and current varies depending on its operating state. For that, if it has transmitted or received, power consumption increases. The BLE is low energy because transmits small packets as compared to the WIFI or the Bluetooth classic and when the BLE device is in sleep mode most of the time and wake up when there is an event.

C. Time

We will get the current time & date by RTC module (Real time clock module), it is a computer clock that keeps track of the current time, and the RTCs are present in almost any electronic device, which needs to keep accurate time, is personal computer electronic board we found the RTC where we found the small button battery, because in the first use we put the correct date and time in it and it computes it to keeps track of the current time.

D. Displaying

In the display we use TFT LCD (thin-film-transistor liquid-crystal display) it is low cost solution to get the best color screen, it is great and colorful, and the fast-moving graphics could help upgrade the user experience, but the options could be limited if the system is hosted on a low-cost or low-power microcontroller (MCU).

E. Storage

On the other hand, in the part of external memory storage we will use SD Card (secure digital card) because it ultra-small flash memory card, and the users know it because the SD cards used in many small portable devices, this solution can cover the lack of BLE in the devices that need to connect to get the data, so if the user doesn’t have a BLE or Bluetooth in the computer or the smart device, he can save the data in the SD card and transfer it to his device.

After this study, we can look to the technical hardware architecture diagram:

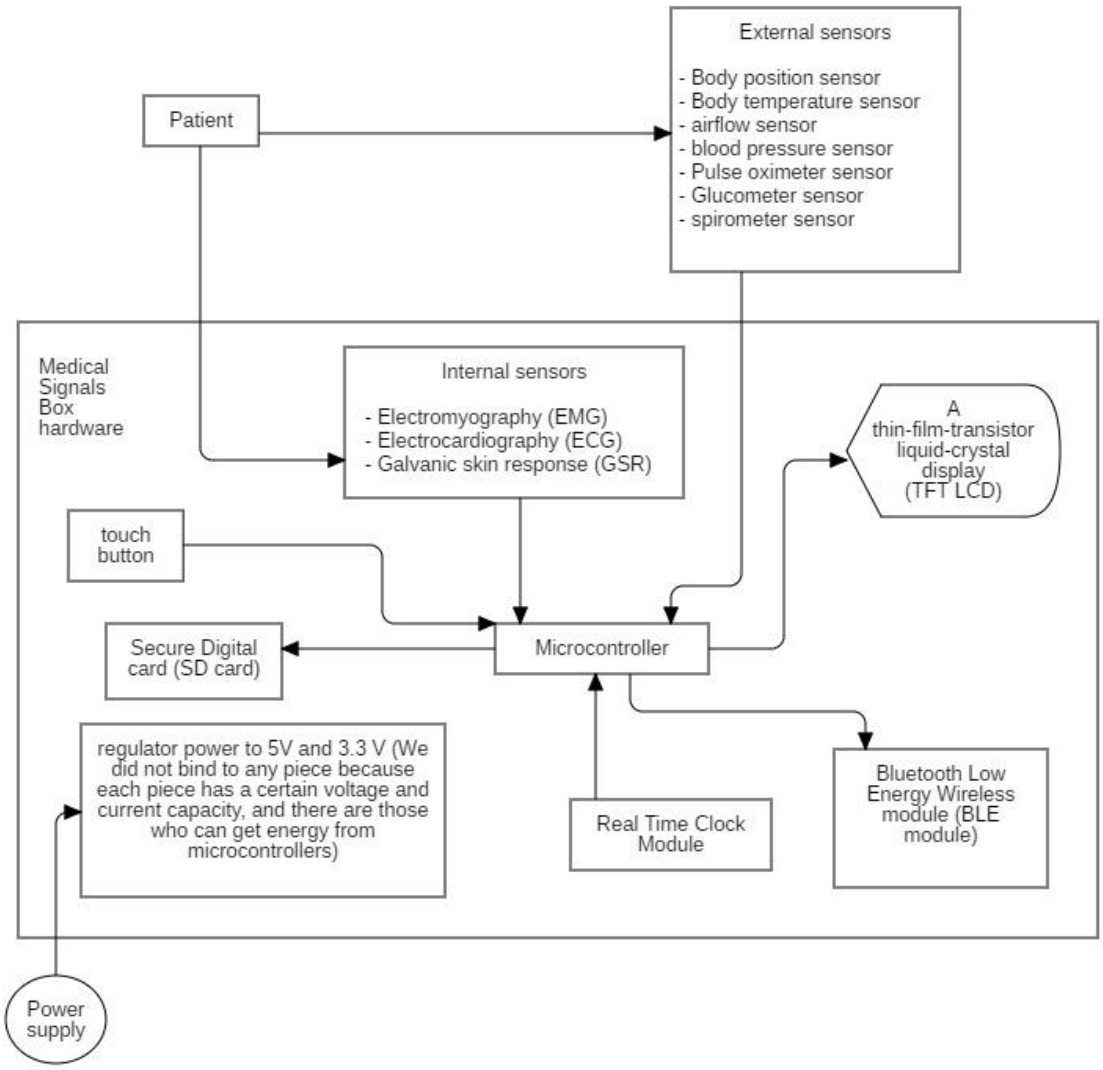

Figure 1 The technical hardware architecture diagram

2. Simplified the architectures to make the prototypes

In Cambridge dictionary, the prototype means the first example of something, such as a machine or other industrial product, from which all later forms are developed: a prototype for/of a new car. The prototyping is great method to create a new market or to understand customers. The prototype makes an idea a reality and makes investors confident by showing them that the idea could actually work. [1]

2.1. Hardware prototype

2.1.1. Hardware prototype structure

In the hardware prototype will focus to minimize the part of the sensors, to reduce the cost and focus on the basics of the project. If we just work with two sensors, we can add more sensors, it is like to build a structure to put books in, after you build the structure you can put a book or two to test how it looks like, after that if the structure is great with the example, you can put other books at time.For that, we will try as much as possible to use: body position sensor, pulse oximeter sensor and electrocardiography (ECG) sensor. In practice, we can get the heart rate from the pulse oximeter sensor or ECG sensor. Our goal in the prototype, to use at least two sensors from the list above, but we will try as much as possible to use them all.

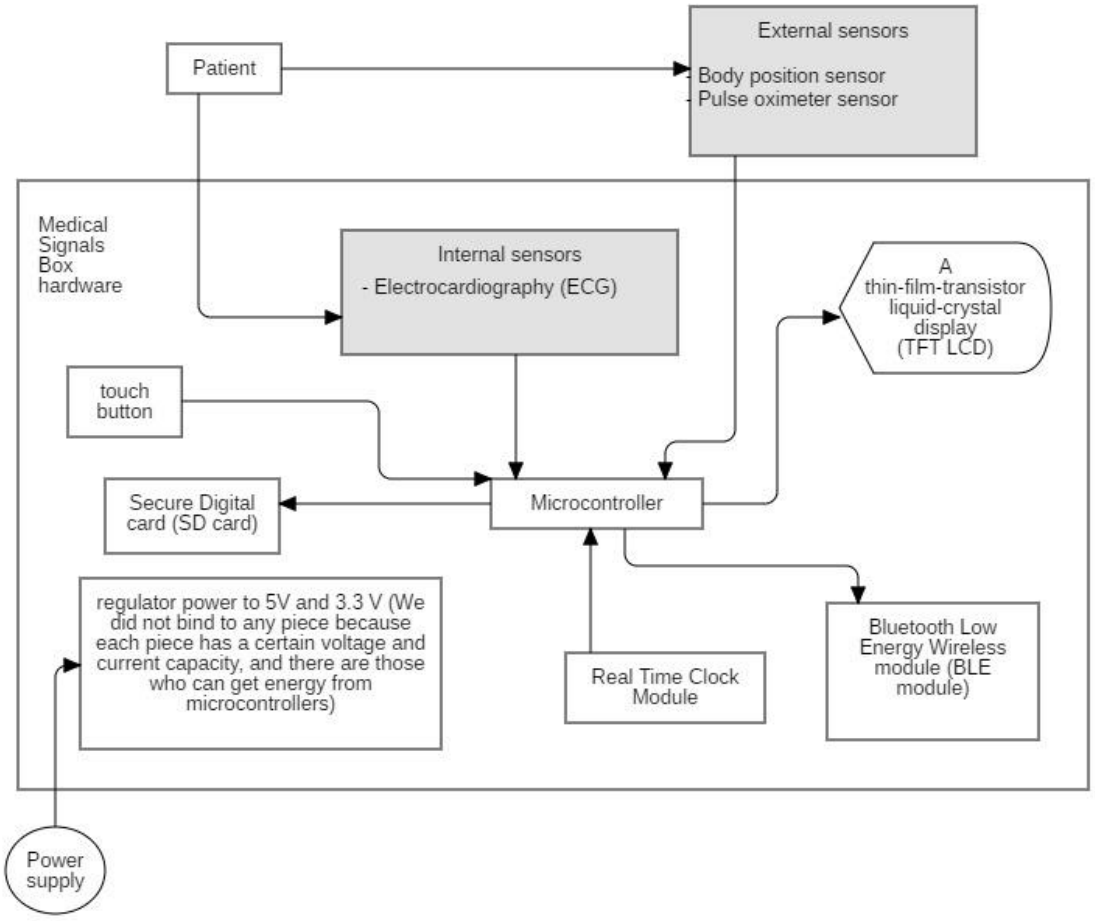

Figure 2 The prototype hardware architecture

2.1.2. Sensors and modules specification

"A module" this term collects all the electronics parts like sensors, power modules, big system like radio FM emitter etc. Moreover, if there is a hardware project, it needs modules, this part will be the most difficult one, because the choice of the modules does not happen just by thinking, but it is composited by knowledge, experiences, research and the deep thinking.

In this prototype, we will need to realize it: body position sensor, body temperature sensor, pulse oximeter sensor, ECG sensor, Microcontroller, TFT LCD screen, SD card reader module, touch button module, BLE module, RTC module, and power module.

After we do a deep analysis, we choose:

A. Microcontroller and BLE module in same module

The ESP32 is the best solution for our project, because it contains all what we need from the performance of analyses the medical signals to the BLE (Bluetooth) that integrate in the module.

Figure 3 ESP 32 module

The information is summarized in this table [2] :

Name	Description
Processors	Dual core Tensilica Xtensa 32-bit LX6 with clock frequency up to 240 MHz and performance up to 600 DMIPS, Ultra low power co-processor and allow to use ADC conversions, computation, and level thresholds while in deep sleep.
Wireless connectivity	- WIFI: 802.11 b/g/n/e/i. - Bluetooth: v4.2 BR/EDR and Bluetooth low energy (BLE).
Memory	- Internal memory:

	ROM: 448 KiB _ for booting and core functions SRAM: 520 KiB _ for data and instruction RTC fast SRAM: 8 Kib _ For data storage and main CPU during RTC boot from the deep sleep mode. RTC slow SRAM: 8KiB _ For co-precessor accessing during deep-sleep mode. eFuse: 1Kibit _ 356 bits for Mac address and chip configuration and 768 bits for customer application, including flash-Encryption and chip-ID. Embedded flash: Flash connected internally via IO16, IO17, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1 on ESP32-D2WD and ESP32-PICO-D4. - External flash & SRAM: ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches. Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported. Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.
Input / output	Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.

In this prototype, we use ESP LOLIN32; it is ESP 32 with a devKit hardware core.

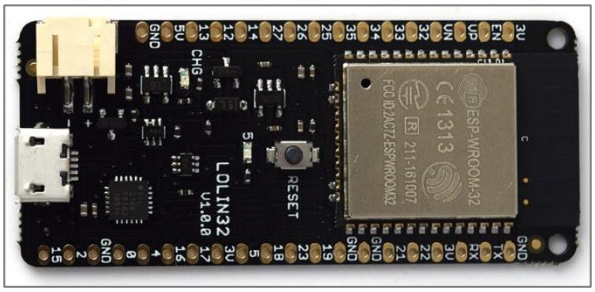

Figure 4 ESP LOLIN 32

B. ILI9225 TFT LCD

In the displaying, we chose ILI9225 TFT LCD, it is a RGB TFT liquid crystal display with resolution of 176x220 dots (dots = 1/3 of pixel, one pixel = 3 dots), 528-channel source driver, 87120 bytes RAM for graphic data, a 220-channel gate driver and power supply circuit.

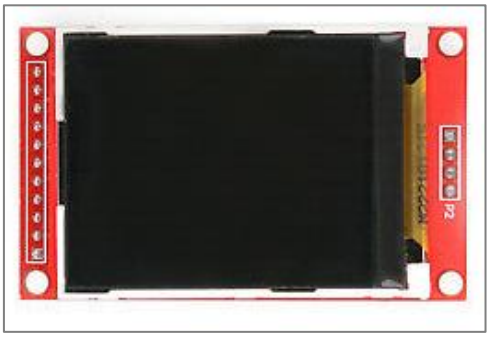

Figure 5 TFT LCD ILI9225 in front

ILI9225 has four kinds of system interfaces which are i80/M68-system MPU interface (8-/9-/16-/18-bit bus width), serial data transfer interface (SPI) and RGB 6-/16-/18-bit interface (DOTCLK, VSYNC, HSYNC, ENABLE, DB[17:0]). In addition, in this prototype we will use the serial data transfer interface (SPI). [3]

C. MAX30102 Pulse Oximeter and Heart-Rate Sensor

The MAX30102 is an integrated pulse oximetry and heart-rate monitor biosensor module. It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection.

Figure 6 MAX30102

The MAX30102 operates on a single 1.8V power supply and a separate 3.3V power supply for the internal LEDs. Communication is through a standard I2C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times. [4]

D. TTP224 Switch Touch Sensor Digital 4 channel Touch Capacitive Module

TTP224 is switch digital touch capacitive touch sensor module, with 4 key board capacitive touch sensor IC. The module can set the output mode, the key output mode, the longest time and fast output / low power options. It operating in 2.4V-5.5V.

Figure 7 TTP224 switch digital touch

E. AD8232 SparkFun Single Lead Heart Rate Monitor

The AD8232 SparkFun Single Lead Heart Rate Monitor is a cost-effective board used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram and output as an analog reading.

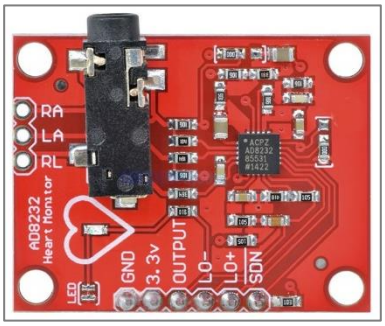

Figure 8 AD8232 heart rate monitor and ECG sensor

AD8232 ECG electrodes are typically wet sensors, requiring the use of a conductive gel to increase conductivity between skin and electrodes:

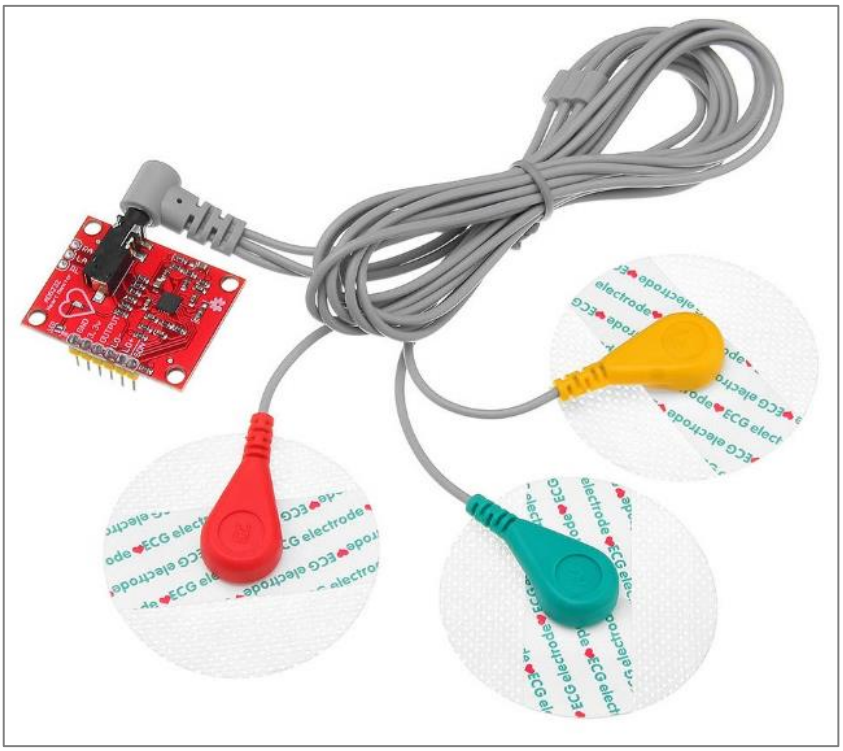

Figure 9 AD8232 with the ECG electrodes

F. MPU-6050 Accelerometer Gyroscope module sensors

The MPU-6050 sensor is accelerometer and gyro sensors in single chip based in MEMS technology. it contains 16-bits analog to digital conversion hardware for each channel, it captures the x, y, and z channel at the same time. MPU-6050 uses the I2C-bus to interface with the microcontrollers. [5]

In this prototype we will use the MPU-6050 with dev kit :

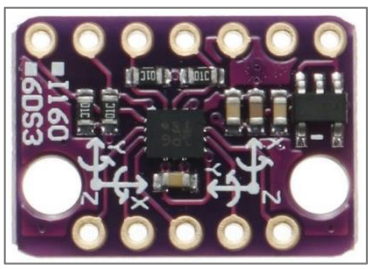

Figure 10 MPU-6050 with dev kit hardware

G. DS3231 real-time clock (RTC) module

The DS3231 is a low-cost, extremely accurate real-time clock (RTC) with an integrated temperature-compensated crystal oscillator (TCXO) and crystal. It uses the I2C-bus to interface with the microcontrollers or processors and it incorporates a battery input. [6]

Figure 11 DS3231 REAL-TIME CLOCK

H. Micro SD Card Adapter (Catalex)

The Catalex micro sd ward adapter is low-cost module to allow to use the Secure Digital card memory with the microcontrollers and processors, it uses the SPI bus to interface it with the microcontrollers.

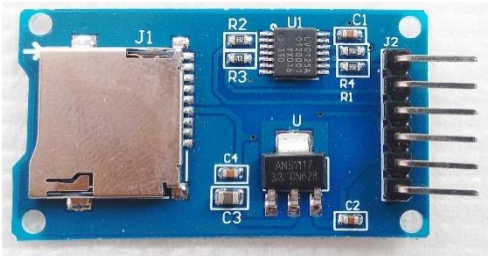

Figure 12 MICRO SD CARD ADAPTER (C

2.1.3. Hardware prototype architecture using chosen modules

We put here all the modules we chose in the main hardware architecture of the prototype:

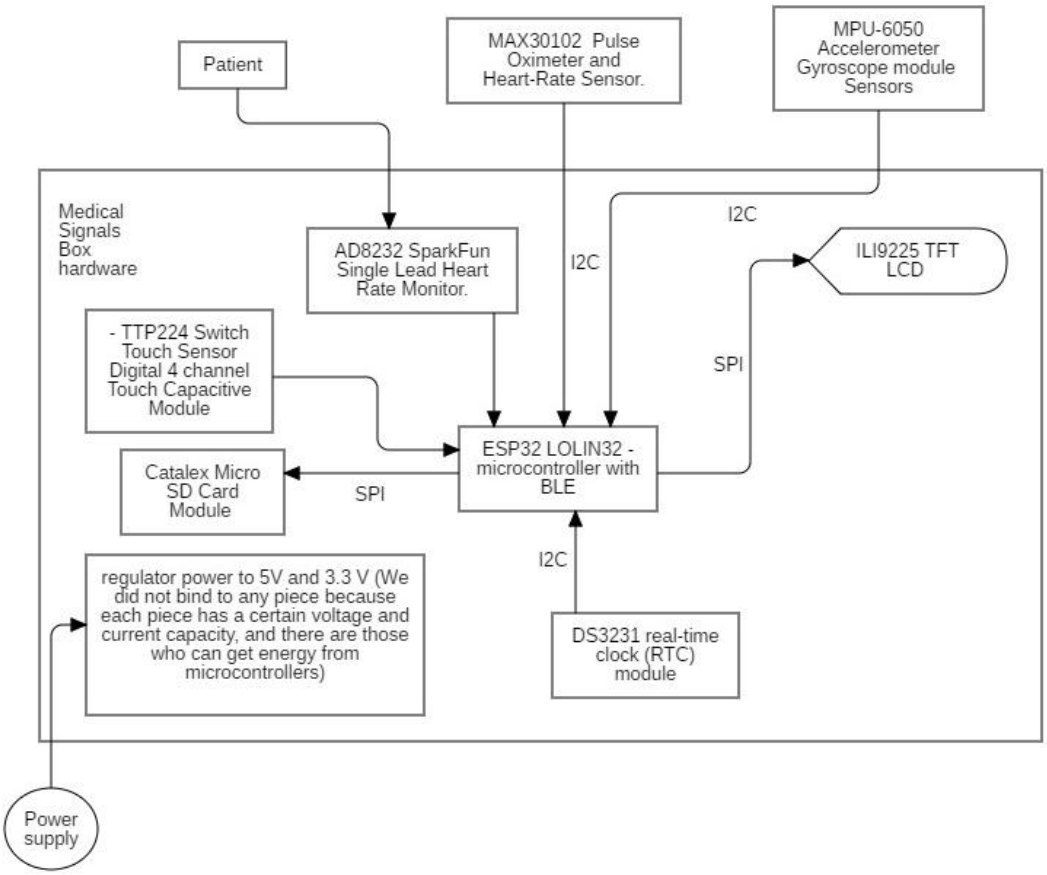

Figure 13 Hardware prototype architecture using hardware modules

2.2.Software prototype

In the software prototype, we decided to minimize all the operations to just three operations: receiving data in the software via Bluetooth from the hardware, displaying the data received in the software interface and saving the data received to a database. But we will focus on reception as a beginning.

We base to development of the software prototype to Linux OS platform with technology multiplatform to simply move it to windows, macOS and Android.

3. From prototype to platform

In this development, we use the modules to realize the project, but in our real project, we need to make two platforms, the first we call it device platform and the second is the developers and students platform.

3.1. Device platform

It is a Medical signals box device platform, that is used by the non-engineer like the doctors or normal persons, they just get the data from the human body, and cannot develop to play with it and its development is just done by the creators of this device.

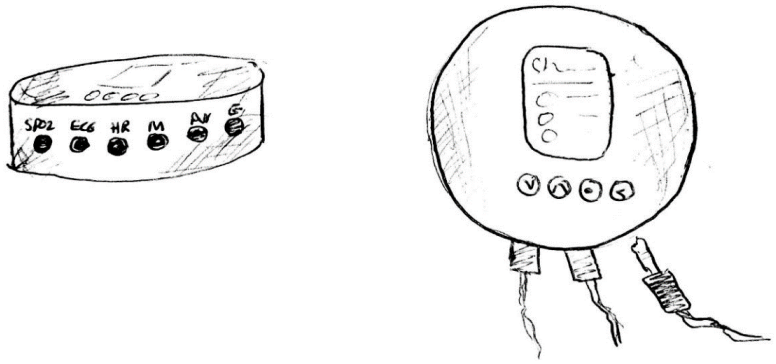

Figure 14 Mock-up of medical signals box device platform

3.2. Developer and student platform

It is a Medical signals box dev platform that is used by the engineer, students, developers, and hackers to learn how to get the data from human body. In this platform, they can develop or play to it, can add more thing like the GPS or the GSM modules. The first design look of the medical signals box developers and student platform is look like raspberry pi board.

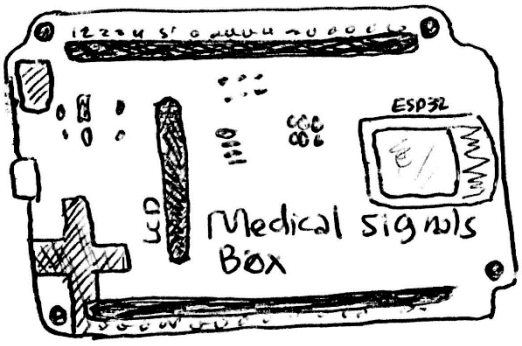

Figure 16 Mock-up of medical signals box developer and student platform

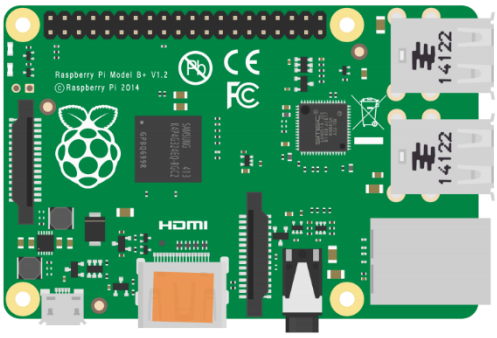

Figure 15 Raspberry pi board design board look

Conclusion

In this chapter, we describe in details the hardware architecture and the software architecture. This description is done regarding the different modules and sensors that we have and those that we can get easily with a minimum cost, and this to be in phase with the first specification needs.

CHAPTER V
REALIZATION

Introduction

This chapter focuses on the prototype realization. We will divide the system in several subsystems, and after that, we will optimize each one to get more performance and efficiency. When we finish the development of those parts, we collect them in the entire system.

1. partial development

We cannot make a big system in one-step, for that we do the partial development, this method will help us to debugging the problems of development and the integration of the modules to the system in an easy way. We divide the system to sub-system, that these subsystems, that have a common thing, like the protocol or interface of communication, the type of the module or other things.

1.1.modules interface by spi interface

The SPI or Serial Peripheral Interface is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices or other microcontrollers quickly over short distances. In SPI, there is master device, which controls the peripheral devices. Typically, there are three lines common to all the devices: the MISO or Master In Slave Out is the Slave line for sending data to the master, the MOSI or Master Out Slave In is the Master line for sending data to the peripherals, the SCK or Serial Clock is the clock pulses which synchronize data transmission generated by the master and one line specific for every device call SS or Slave Select and it is the pin on each device that the master can use to enable and disable specific devices. When a device's Slave Select pin is low, it communicates with the master. When it's high, it ignores the master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines. [1]

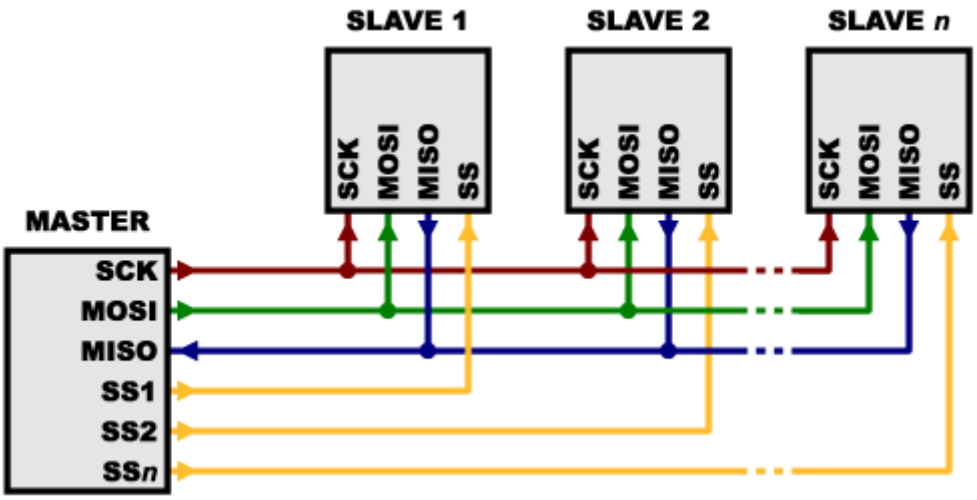

FIGURE 1 THE LINKS BETWEEN MODULES IN SPI

- **ILI 9225 TFT LCD**

The ILI9225 TFT LCD can interface it by SPI interface, when we see the TFT from the back; we can see its pinout:

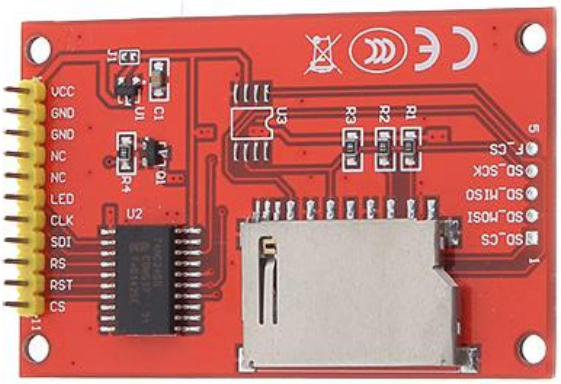

FIGURE 2 ILI9225 TFT LCD FROM THE BACK

This table summarizes the descriptions the pins of the TFT LCD:

TABLE 1 TABLE DESCRIBES THE PINS OF ILI9225 TFT LCD

TFT Pin Label	Description
VCC	3.3 V Supply (look at the note below this table to understand why we used 3.3V not 5V)
GND	Ground
GND	Ground
NC	No Connection
NC	No Connection
LED	LED Backlight
CLK	SPI Clock
SDI	SPI MOSI
RS	SPI Data/Command Select
RST	Reset
CS	SPI Chip Select

In our ILI9225, when we tried to test it with Arduino (we tested all the modules with Arduino before ESP32, to make sure that it is not damaged, and we have just one ESP32 module), a problem occurred in it, and the voltage regulator U1 are damaged, so in the last image from the back of the TFT you can see there is J1 on top of U1, we made a bridge in the J1 to not use the voltage regulator that transform the 5V to 3.3v and we used directly 3.3 v in the VCC of the TFT. For that we put 3.3 V not 5V, but when someone needs to make a new prototype with correct TFT board, he needs to put 5 V not 3.3 V.

In addition, when we look at part of pinout of ESP lolin32:

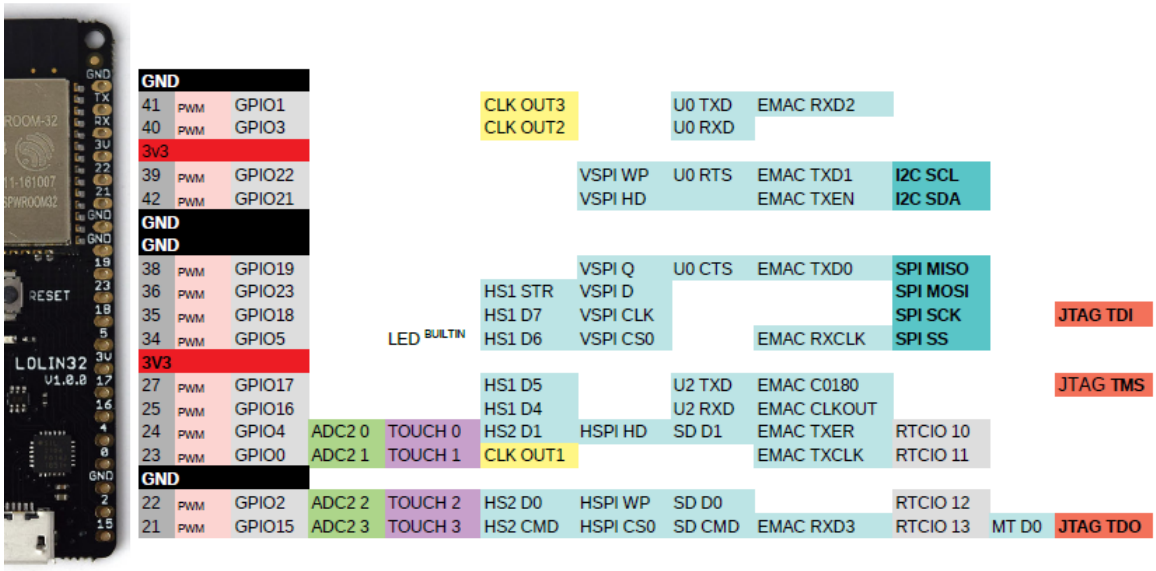

FIGURE 3 PART OF ESP LOLIN32

We see that the SPI MISO in the pin 19 (GPIO19), the SPI MOSI in the pin 23 (GPIO23), the pin SPI SCK in the pic 18 (GPIO18) and the SPI SS in the pin 5 (GPIO5). For that, we make a table that connects the pins of TFT with the pins of ESP2:

TABLE 2 THE TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND TFT ILI9225

TFT	ESP32 pin number	ESP32 pin name
CLK	18	SPI SCK
SDI	23	SPI MOSI
RS	2	/
RST	15	/
CS	5	SPI SS
LED0	/	3.3V
VCC	/	3.3V
GND	/	GND

We do not connect SPI MISO (GPIO19 in ESP32) with the pins of TFT, because the TFT does not send anything to the ESP32, it just receive the data and display it. In addition, if the TFT is touch screen we need to connect the SPI MISO because the TFT with touch screen is send the data to the microcontroller not just receive.

To display a test, we use Arduino programming language to make a firmware based ILI9225 Arduino library; this diagram (algorithm) explains how to display an image (after transfer, it to 2bytes hex or 0xxx), we note that we use an Arduino library special for ILI9225:

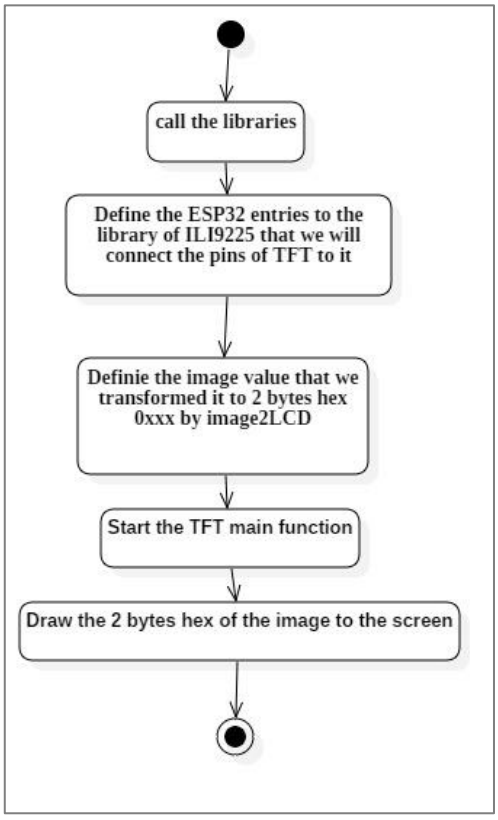

FIGURE 4 DIAGRAM EXPLAIN HOW TO DISPLAY AN IMAGE TO TFT ILI9225

The next figure is about the Image2LCd, this software we use it to convert image (Like the logo of Medical signals box in the picture) to array of 2bytes hex or 0xxx.

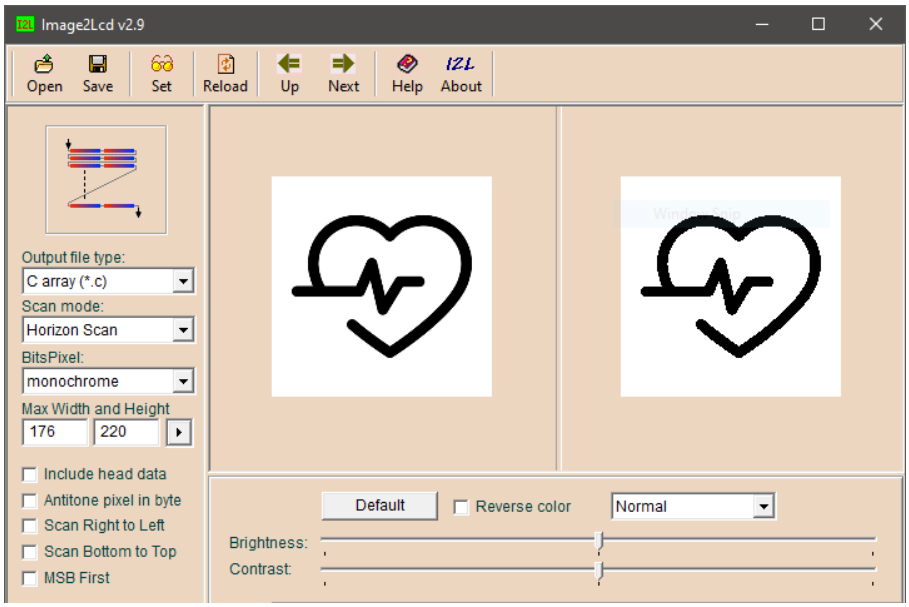

FIGURE 5 IMAGE2LCD THAT WE USE IT TO CONVERT FROM IMAGE TO ARRAY OF 2BYTES HEX

The result from the past example of converting the logo of Medical signals box project is this array (We remove a big part from this array because it consumption of a large area of papers):

```
const unsigned char Medical_signals_box_logo[3872] = { /* 0X00,0X01,0XB0,0X00,0XB0,0X00,
*/
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X1F,0XFF,0XFF,0XC0,0X00,0X00,0X00,0X00,0X00,0X00
,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X07,0XFF,0XFF,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
…
…
…
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,0X00,
};
```

We note that this TFT does not has a function to display the color image, or RGB image, and the last array we will use it to display with one color.

- **Micro SD Card Adapter Catalex**

The micro SD card adapter Catalex can interface it by SPI interface, and when we see it from the back; we can see its pinouts:

We connect this module with the ESP lolin32 like what we do with TFT, and this table

FIGURE 6 MICRO SD CARD READER CATALEX

link the pins of the SD card reader module with the pins of ESP32:

TABLE 3 TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND SD CARD READER MODULE

Micro SD Card Adapter Catalex	ESP32 pin number	ESP32 pin name
CS	5	SPI SS
SCK	18	SPI SCK
MOSI	23	SPI MOSI
MISO	19	SPI MISO
VCC	/	5V
GND	/	GND

In firmware part, to communicate with SD card there is two mode: the SD mode or the SPI mode. By default, the SD card operates in the SD mode. However, we work with the SPI mode and communicate with it using the SPI protocol because the SPI easy to use and easy to integrate ether modules. On other, Communication with the SD card is performed by sending commands to it and receiving responses from it.

However, why there is no talking about SD card reader with I2C mode? In the first SD is standards from SD Association, so the SPI is integrated in the SD card by default. The second, the SPI is faster than I2C and for that, no wonder it is so hard to find SD card with i2c interface. There is a solution to use I2C with the SD card but this solution is that we need to make a new driver, the solution is to use a bridge from the I2C to SPI.

In practice of this prototype, we use Arduino programming language and its SD library, but not the original library that works with Arduino board but a modified version to adapt with ESP32, this diagram (algorithm) explain how to write a data to data.txt file in the SD card based the SD card library of Arduino:

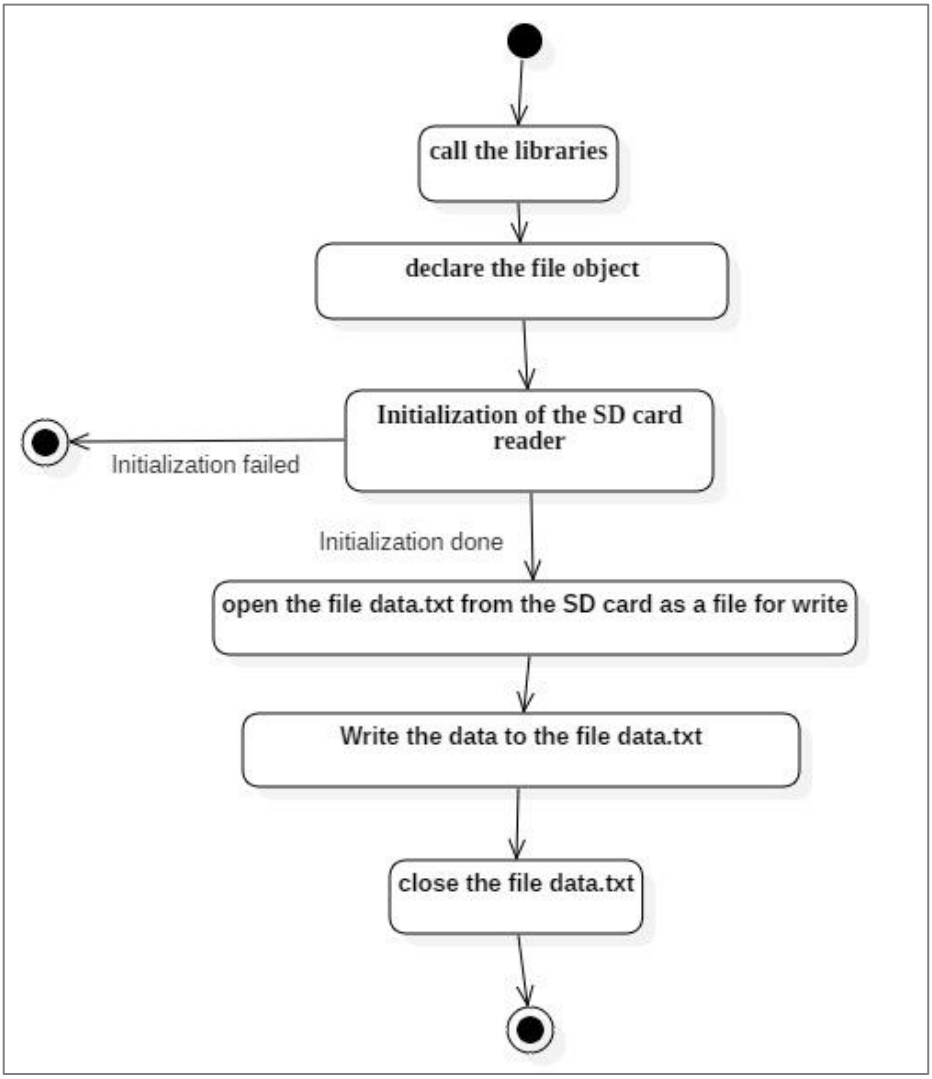

FIGURE 7 DIAGRAM EXPLAIN HOW TO SAVE A DATA TO DATA.TXT IN SD CARD

1.2.modules interface by I2C interface

The I2C is a serial protocol with two wire (SCL or serial clock and SDA or serial data) to connect low-speed devices like sensors, microcontrollers, I/O interfaces and other similar modules in embedded systems. In I2C, each slave device has a unique address and it can transfer from and to master device is serial (it is split into 8-bit packets).

To use the I2C, we need to get the I2C address for each module that we connect it to the bus, we can read the datasheet of each module but we can use an Arduino script call I2C scanner that can give us the I2C address of any I2C device in the bus [1].

We note that we need to put a pulled up resistors to VCC for each wire from the two wires; the pull-up resistor value depend on a number of factors. In example, Texas Instruments recommends in " I2C Bus Pull up Resistor Calculation in Application Report SLVA689 that published in February 2015 ", the following formulas to calculate the correct pull-up resistor value:

VOL: the logic low voltage

IOL: the logic low current

Tr: the maximum rise time of the signal

Cb: is the bus (wire) capacitance.

EQUATION 1 FORMULA OF TEXAS INSTRUMENTS TO CALCULATE THE CORRECT PULL-UP RESISTOR

$$R_p(min) = \frac{V_{DD} - V_{OL}(max))}{I_{OL}}$$

$$R_p(max) = \frac{t_r}{0.8473xC_b}$$

However, the pull up resistor typical is 4.7 kΩ if we do not want to calculate it.

Having pull-up resistors is an open-drain scheme. Bus devices pull-down the voltage on the bus instead of using their own operating voltage. This allows devices with different operating voltages to be usable but only if the lower voltage is still readable by the higher-voltage device. For example, it is OK to connect a 3.3V I2C sensor to a 5V Arduino because the latter can still read 3.3V. Nevertheless, a 5V I2C sensor will not work with a 3.3V microcontroller. In our project, the ESP32 is 3.3V module and can the modules that has I2C with 5V would not work with it without pull-up resistors [2].

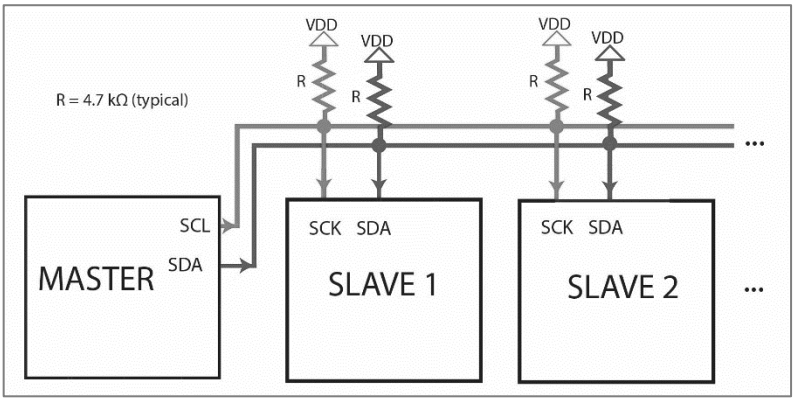

FIGURE 8 THE LINKS BETWEEN MODULES IN I2C

In ESP lolin32, the I2C pins: in GPIO 22 the I2C SCL and in GPIO 21 the I2C SDA.

- **MAX30102 Pulse Oximeter and hart rate sensor**

MAX30102 can interface it by I2C interface, when we see it from the back; we can see its pinouts:

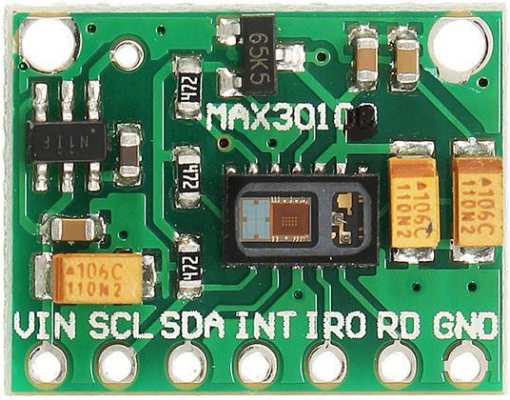

FIGURE 9 MAX30102 FROM THE BACK

We just use the SCL and SDA for communication, the VIN put it to 3.3V; IR0, RD, and GND to GND. This table summarized how we linked it with the ESP LOLIN32:

TABLE 4 TABLE SUMMARIZED HOW WE LINKED MAX30102 WITH THE ESP LOLIN32

MAX30102	ESP32 pin number	ESP32 pin name
VIN		3.3V
SCL	GPIO 22	I2C SCL
SDA	GPIO 21	I2C SDA
INT	/	/
IR0		GND
RD		GND
GND		GND

We use I2C scanner Arduino script to get the I2C address of this module to not wast the time reading the Datasheet to search for it, and we found that the I2C address of the max30102 is 0x57:

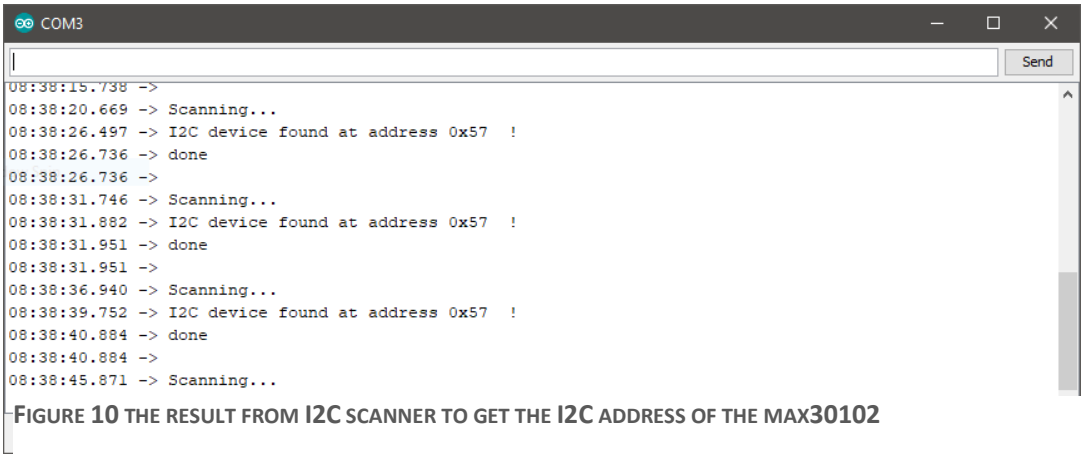

FIGURE 10 THE RESULT FROM I2C SCANNER TO GET THE I2C ADDRESS OF THE MAX30102

We use Arduino programming language with Max30102 library modified from the library of Max30105, this diagram (algorithm) explains how to get SPO2 and heart rate (HR) value from the sensor:

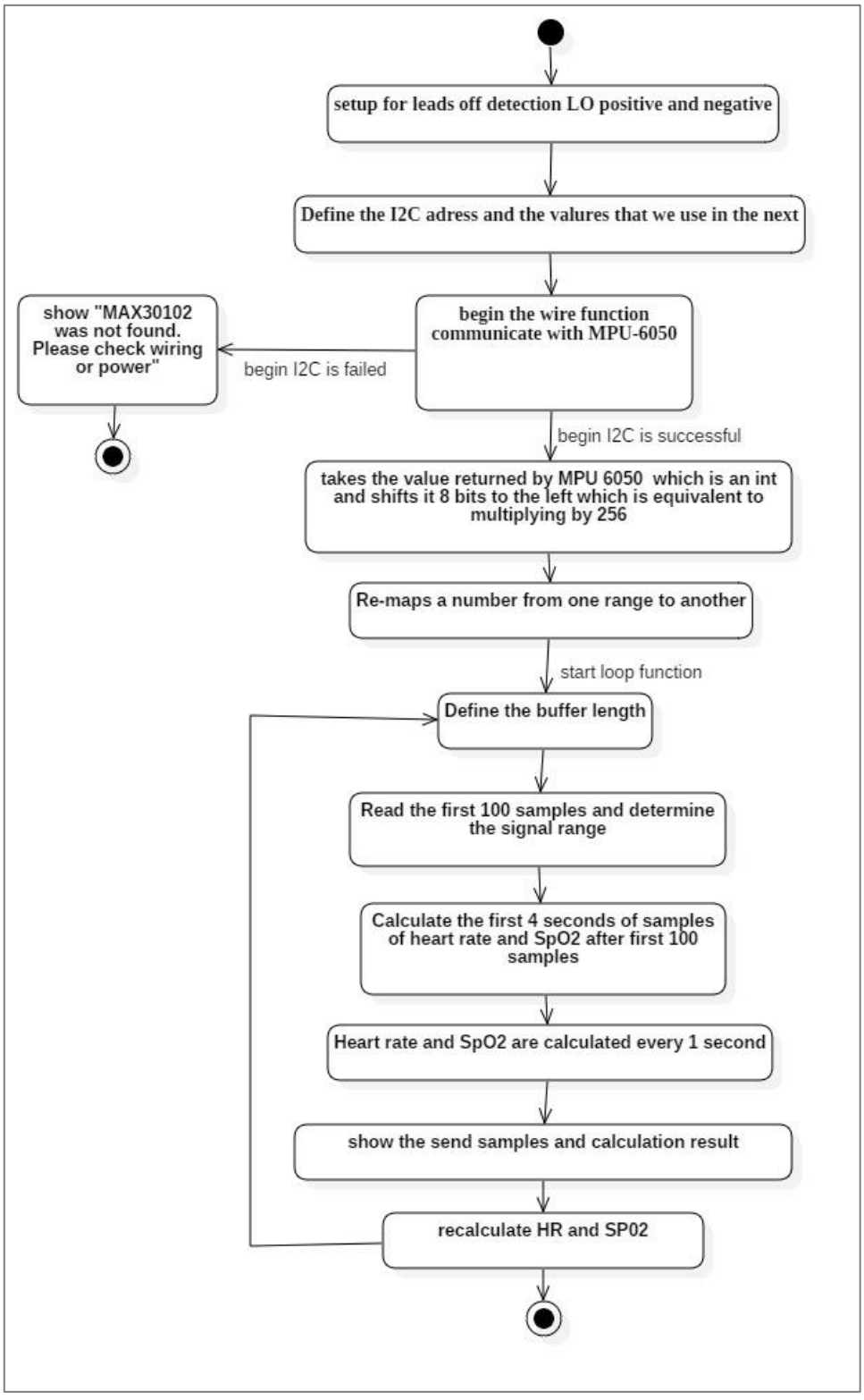

FIGURE 11 DIAGRAM EXPLAIN HOW TO GET THE SPO2 AND HART RATE FROM THE SENSOR

- **MPU-6050 body position**

MPU-6050 can interface it by I2C interface, when we see it from the back; we can see its pinout:

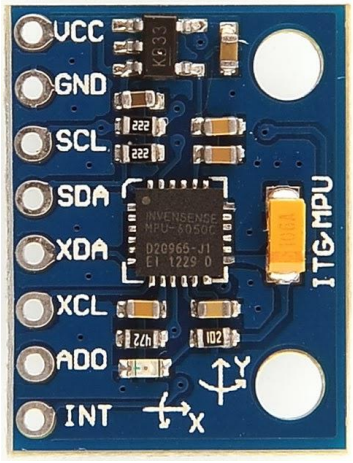

FIGURE 12 MPU-6050

We just use the SCL and SDA for communication, the VCC put it to 3.3V; and GND to GND. This table summarized how we linked it with the ESP LOLIN32:

TABLE 5 TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND MPU-6050

MPU-6050	ESP32 pin number	ESP32 pin name
VCC	/	3.3V
GND	/	GND
SCL	GPIO 22	I2C SCL
SDA	GPIO 21	I2C SDA
XDA	/	/
XCL	/	/
AD0	/	/
INT	/	/

We use I2C scanner to get the I2C address of this module, and we found that the I2C address of the MPU-6050 is 0x68, we note that the 0x57 address is the I2C address of max30102 because we put the two module MPU-6050 and MAX30102 together to the I2C interface for this test:

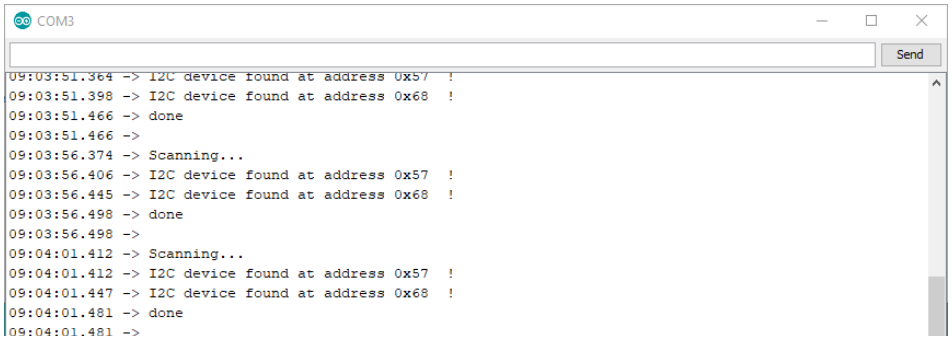

FIGURE 13 THE RESULT OF I2C SCANNER TO GET THE I2C ADDRESS OF THE MPU-6050

We use Arduino programming language with Max30102 library modified from the library of Max30105, this diagram (algorithm) explains how to get SPO2 and heart rate (HR) value from the sensor

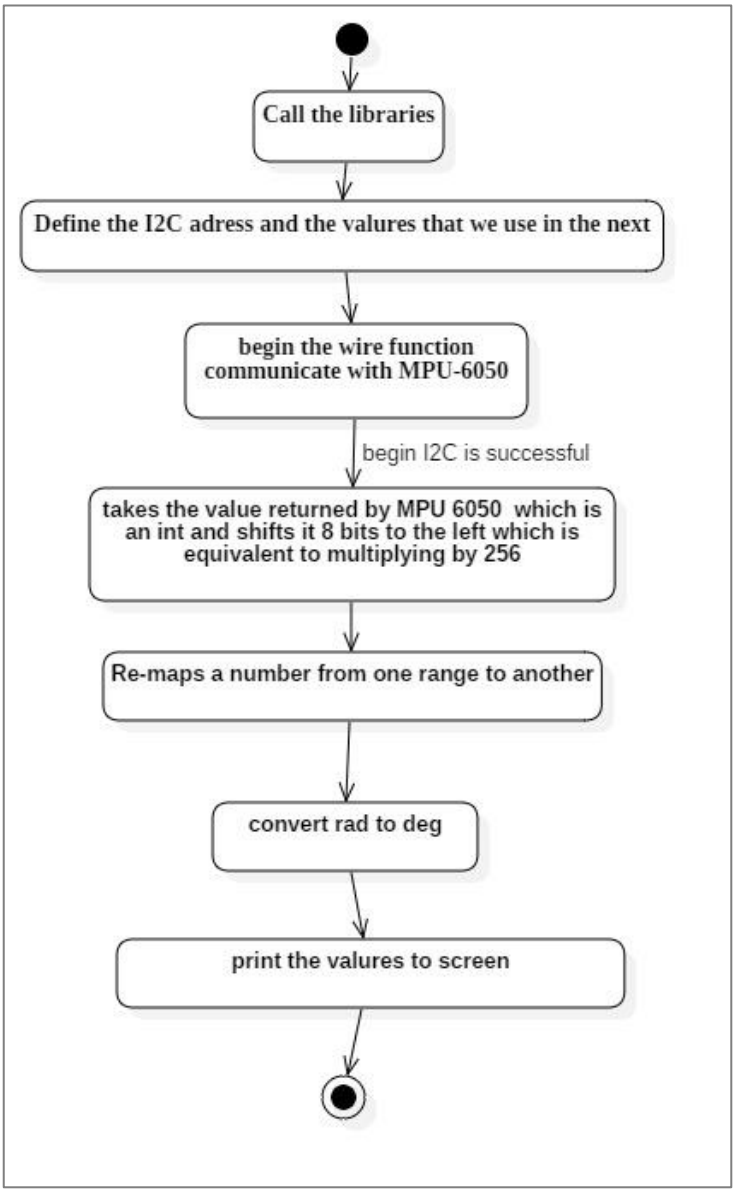

FIGURE 14 DIAGRAM EXPLAIN TO GET THE ANGLE IN DEGREES BY MPU-6050

- **DS3231 REAL-TIME CLOCK**

DS3231 can interface it by I2C interface, when we see it from the back; we can see its pinout:

FIGURE 15 DS3231 REAL-TIME CLOCK

We just use the SCL and SDA for communication, the VCC put it to 3.3V; and GND to GND. This table summarized how we linked it with the ESP LOLIN32:

TABLE 6 TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND DS3231

DS3231	ESP32 pin number	ESP32 pin name
SCL	GPIO 22	I2C SCL
SDA	GPIO 21	I2C SDA
VCC	/	3.3 V
GND	/	GND

We use I2C scanner to get the I2C address of this module, and we found that the I2C address of the DS3231 is 0x68, It is the same address of MPU-6050, and to fix this problem there is a pin in MPU-6050 call AD0 just link it with VCC (or 3.3V) and the address of MPU6050 will change from 0x68 to 0x69, and for that the address of DS3231 is 0x68 and MPU-6050 is 0x69:

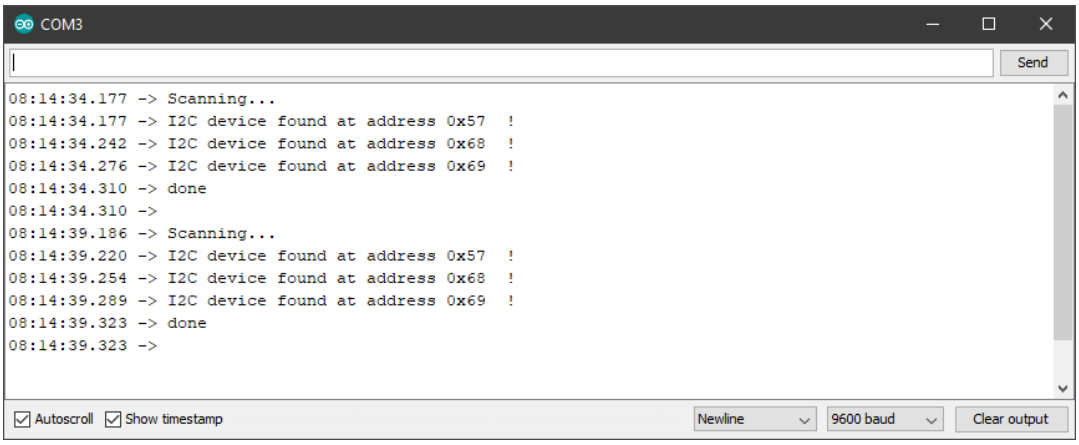

FIGURE 16 THE RESULT OF I2C SCANNER TO GET THE I2C ADDRESS OF THE DS3231

We use Arduino programming language with ds3231 library to get the date and time from RTC, we note that if the RTC has incorrect information we need to correct it by using a small code to set the time and the date, this diagram (algorithm) explain how to get date and time value from the DS3231:

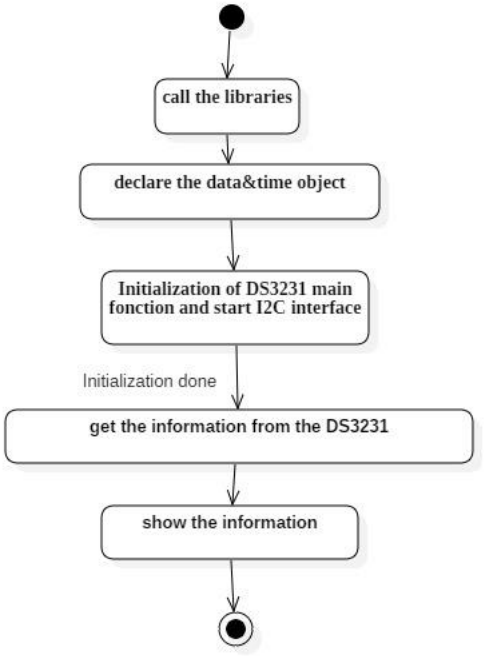

FIGURE 17 DIAGRAM EXPLAIN TO GET THE DATE AND TIME FROM DS3231

1.3. AD8232 ECG SENSOR

The AD8232 output the signal of the data as an analog reading, for that the ESP lolin32 read it by pin has ADC (Analog to Digital Converter), in the front of the module, we can see the pinout:

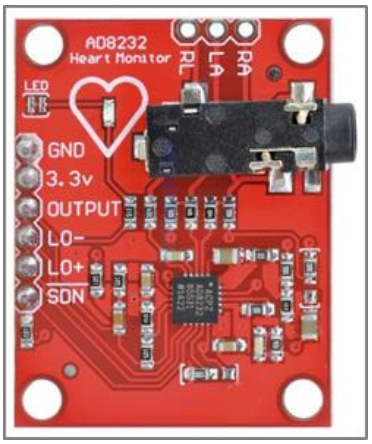

FIGURE 18 THE MODULE AD822 FROM FRONT

We summarize how we linked it with ESP Lolin 21 in this table:

TABLE 7 TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND AD8232

AD8232	ESP32 pin number
GND	
3.3V	
OUTPUT	GPIO 25
LO-	GPIO 33
LO+	GPIO 32
SDN	/

This pinout shows the location of the pins that we will use with AD8232:

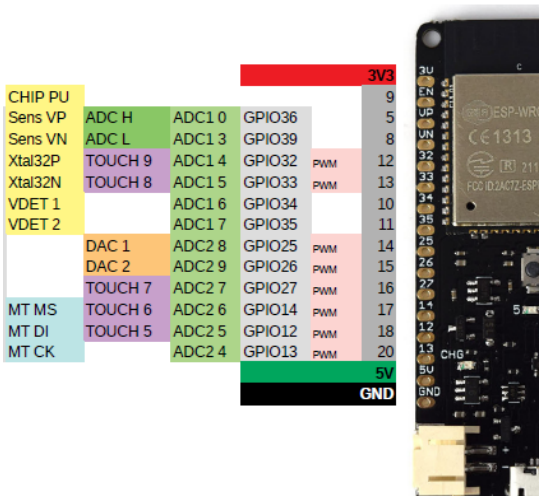

FIGURE 19 PART FROM PINOUT OF ESP LOLIN32

The AD8232 does not need an Arduino library to program the microcontroller to work with it, we just need to check no LO+ or LO1 equal 1 and read the analog input GPIO 25. This diagram (Algorithm) explains how to programming the ESP lolin32 to get the data from the AD8232:

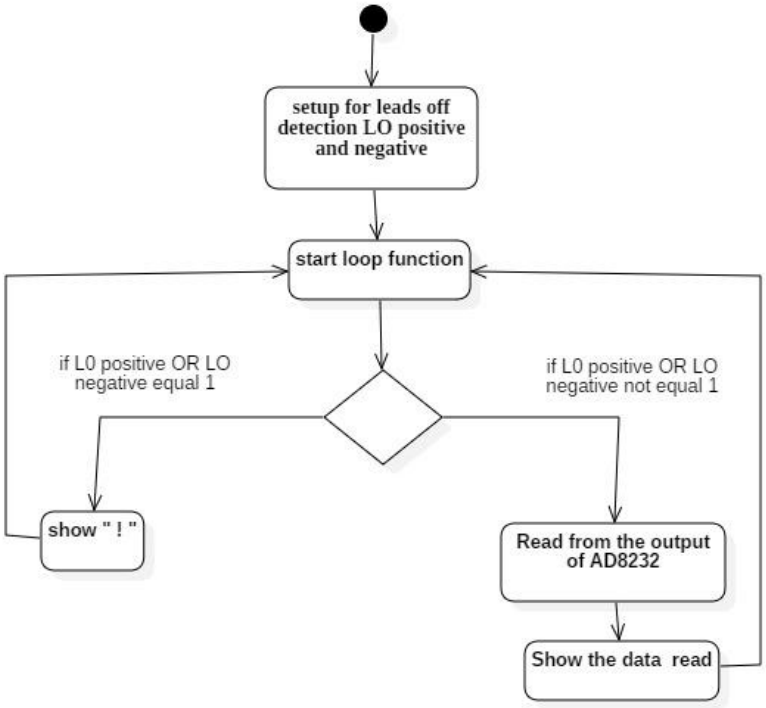

FIGURE 20 DIAGRAM EXPLAIN TO GET THE DATA FROM AD8232

When we tested and made the scripts in this prototype v1, we did not focus to get real data, because in the first we do not have an original modules, we bought a low cost modules from china, just to make our prototype in low cost, and we do not have an ideal modules to test with it or know the real reaction of the modules to the ESP lolin32. We know that we can make a prototype and like the original one, and when we succeed with it, we can just change the modules and change some things in the scripts like the parameters or the number of the data that we will get it from the modules.

We note that we placed the AD8232 electrodes like that:

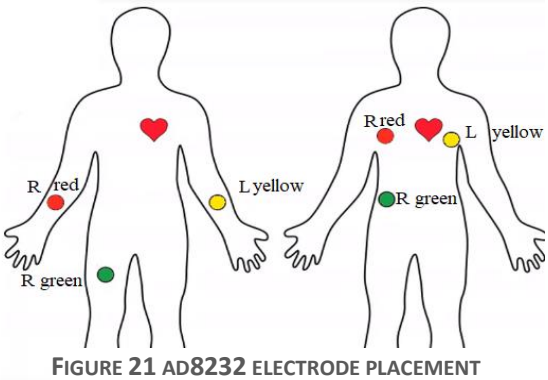

FIGURE 21 AD8232 ELECTRODE PLACEMENT

1.4. TTP224 switch touch sensor digital

The ttp224 is the simple module in all the prototype, it just need the power and GND, and if the touch one active (touched by finger) the out1 be HIGH or 3.3 V, and if not active (not touched by finger) the out1 be LOW or 0 V.

FIGURE 22 TTP224 MODULE

We summarize how we linked it with ESP Lolin 21 in this table:

TTP224	ESP32 pin number
OUT1	GPIO 13
OUT2	GPIO 12
OUT3	GPIO 14
OUT4	GPIO 27
GND	GND
VCC	3.3 V

FIGURE 23 TABLE EXPLAINS THE LINKS BETWEEN THE PINS OF ESP LOLIN32 AND TTP224

This module is very simple to use and we does not need a diagram to explain how to use it. We just define that the GPIO12, GPIO12, GPIO14 and GPIO27 as input pins, after that we watch any change in the input, so if the pin equals 1 then it is active, in electric the 1 means 3.3V

1.5. Bluetooth low energy (BLE) server with ESP Lolin32

In the hardware, to send the data to the computers and the smart devices like the smartphone, we use in the ESP loling32 the feature of BLE, in the scenario of how we use BLE, the user after recording the data, chooses to open the BLE server that the system put the data recorded into it.

Before we make the diagram (algorithm) that help us to develop a firmware to test the BLE server, we need to talk about some idea in BLE.

In BLE theory, there is some terms we need to know. The first, the Generic Attribute Profile (GATT), it specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile. In other words, it is a set of rules describing how to bundle, present and transfer data using BLE. The second, the service, it is a collection of data and associated behaviors to accomplish a particular function or feature; In other words, a service is a collection of information, like e.g. values of sensors. The third, the characteristic, it is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. In other words, the characteristic is where the actual values and information is presented. The fourth, the UUID, and it is an abbreviation you will see a lot in the BLE world. It is a unique number used to identify services, characteristics and descriptors, also known as attributes. These IDs are transmitted over the air so that e.g. a peripheral can inform a central what services it provides.

In the other hand, this terms are not independent from each other, the relationship between the service and the characteristics can be explained by this example, imagine a BLE sensor as an information server that gives some kind of information, for instance the Battery Service, provides some characteristics (ex: Battery voltage level, current, time of recharge, etc... Therefore, a service will have one or more characteristics that can be read or written by another device (i.e. the phone).

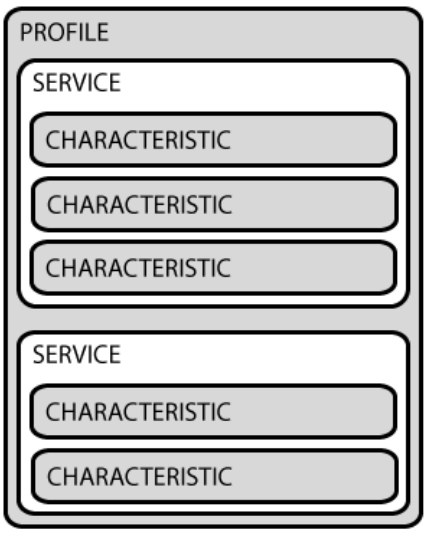

FIGURE 24 THE SERVICE AND THE CHARACTERISTICS IN BLE

To program the ESP lolin32 to make BLE server we use an Arduino library that makes it easier for us to develop it.

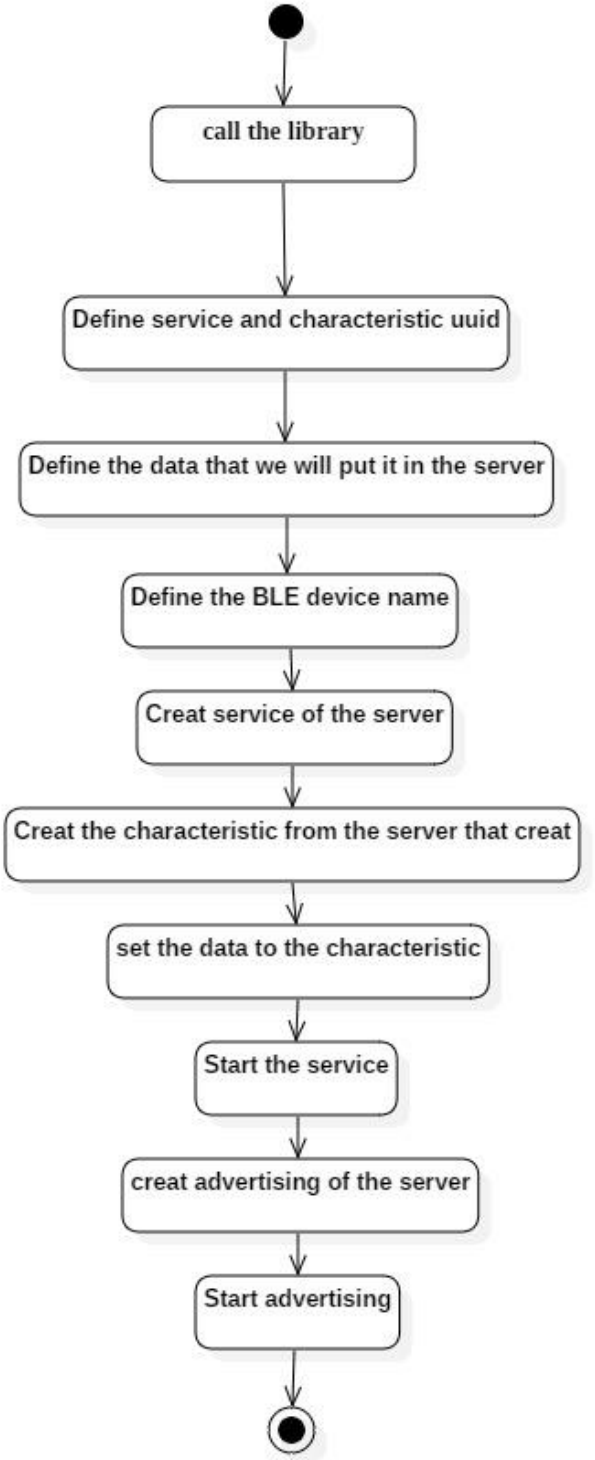

FIGURE 25 DIAGRAM EXPLAIN HOW TO CREAT BLE SERVER AND PUT THE DATA INIDE THE CHANAL

we programmed and uploaded and ran the ESP lolin32, we went to test the receiving of the data in the computer. In Linux (Ubuntu 18.04), we use our software that we made for the project and we see it in the next, but in Windows (Windows 10), we use a Windows Store open source app that allows us to interact with Bluetooth Low Energy devices by GATT protocol, this app can do many things like, enumerate BLE devices that paired with the computer, see device MAC address, name, list of services and characteristics, see names and descriptions for services and characteristics, read characteristics value, display them in various, and more things.

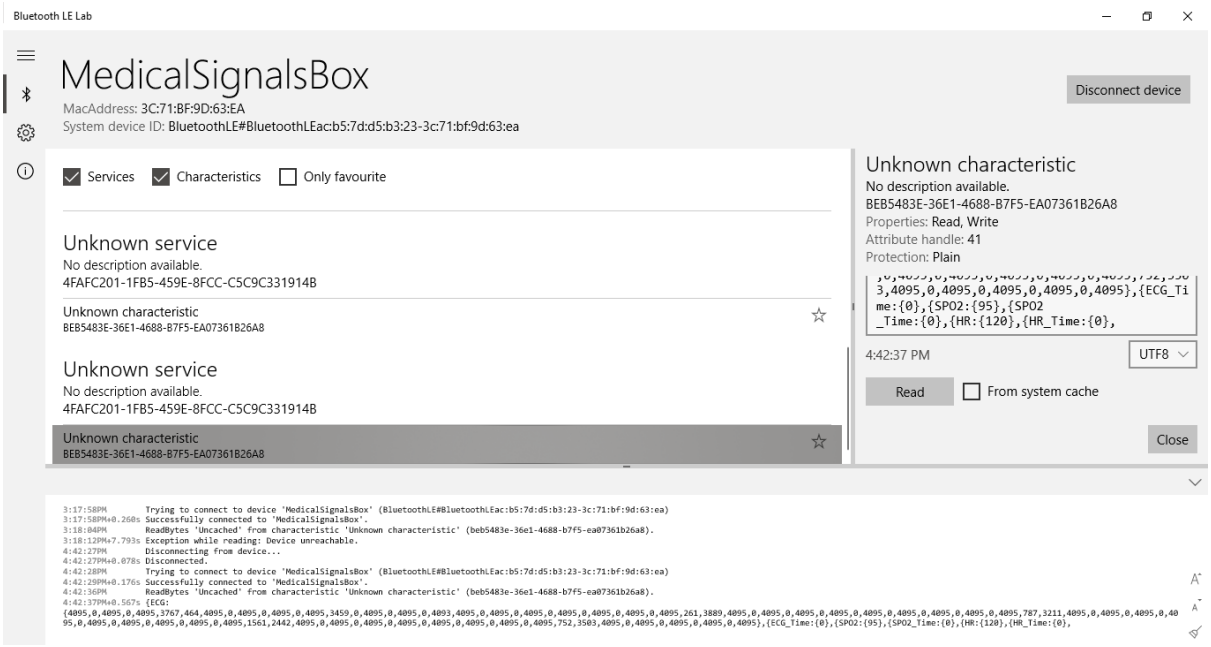

FIGURE 26 SCREENSHOT OF THE WINDOW OF THE BLUETOOTH LE LAB WHEN RECIEVE AND READ THE DATA FROM THE MEDICAL SIGNALS BOX HARDWARE

1.6. Receive the data from the hardware via Bluetooth in computer or smart device and save it to an intern database:

This part is of the Medical signals box Software, the first thing we do here to make the software of the system is to receive the data that we put it in the BLE server channel. For that we start to work with a small code by JAVA programming language to receive, but after spending a long time testing many JAVA Bluetooth libraries we know that java is not for our for this problem, and we note that we test in the both Linux and Windows. We move to work with Python programming language, in Linux working with python is so easy, but in Windows we look like python is not the programming language for Windows, so we stop again working with python. The next solution that we tried, it is to use Node.js; Node.js is an open source server environment runs on various platforms (Linux, Windows ...) and it uses JavaScript on it. However, why we chose this language? Because this solution developed rapidly and got a strong community of programmers who developed solutions for everything.

Therefore, our solution is to make a JavaScript code run in Node.js server to receive the data from Bluetooth. We use noble module, which is a Node.js BLE central module to easy use BLE in JavaScript that run in Node.js server. This solution work perfectly in Linux (Ubuntu 18.04), but it is not great solution for Windows (Windows 10), because it does not work and big configuration to run, and the user using it can make this configuration because is so hard. For this reason, we decided to stop working with Node.js in Windows and just use it in Linux. Because in this type of development that use a hardware communication, the philosophy of windows with the hardware is not like in Linux, in Linux anything is file which makes hardware manipulating so easy. In addition, the one best solution to develop the Medical signals box software for Windows is to use C# because it’s developed by Microsoft and compatible with the philosophy of Windows.

We develop an algorithm and based it on JavaScript code that receives and saves the data received to a database:

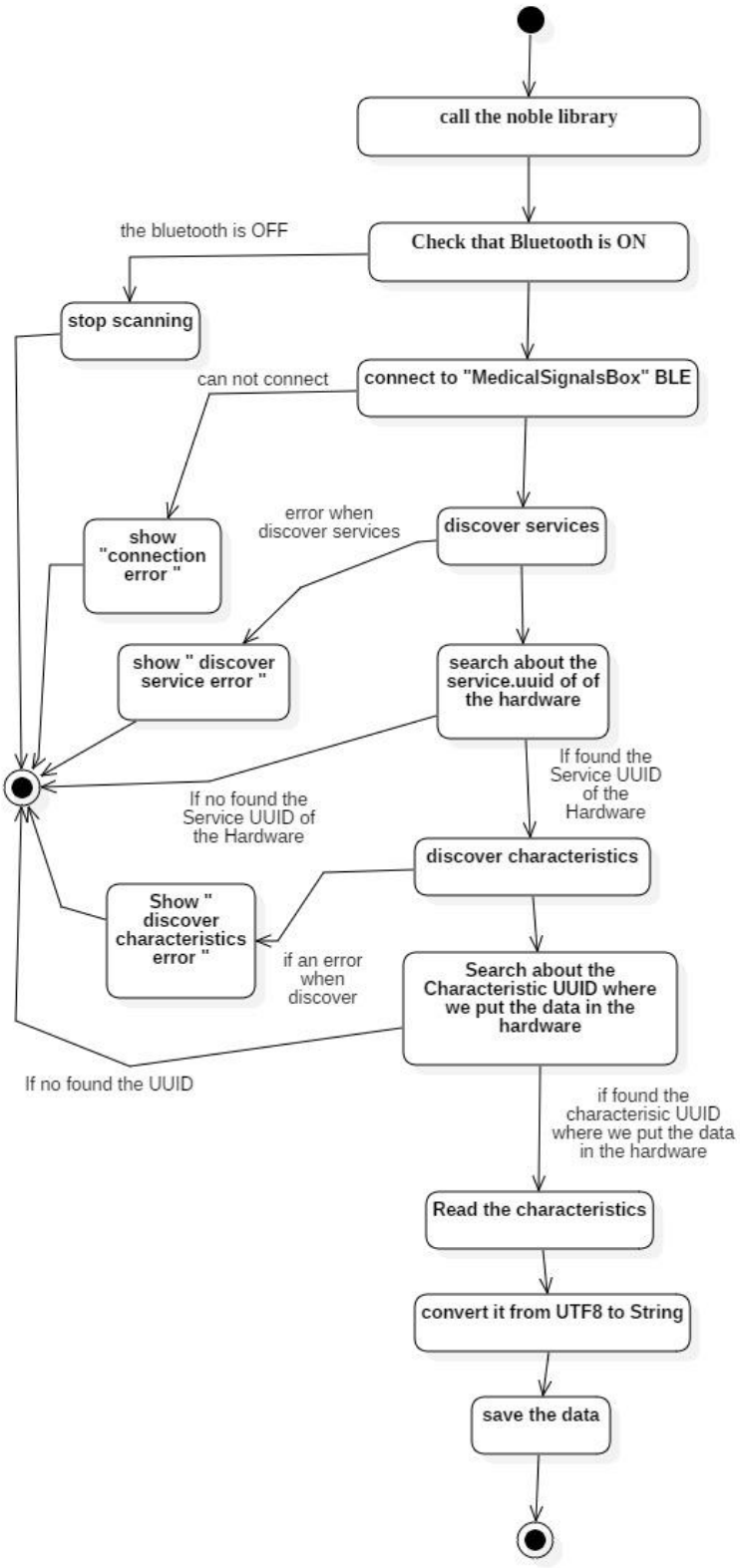

FIGURE 27 DIAGRAM (ALGORITHM) EXPLAIN HOW TO RECEIVE AND SAVE THE DATA FROM THE HARDWARE TO COMPUTERS OR SMARTPHONES

1.7. System power circuit with Rechargeable Battery Backup

In the prototype, we use the USB power supply to the ESP Lolin32 to power the system, but we can use a battery or power circuit with the external power port in ESP Lolin32.

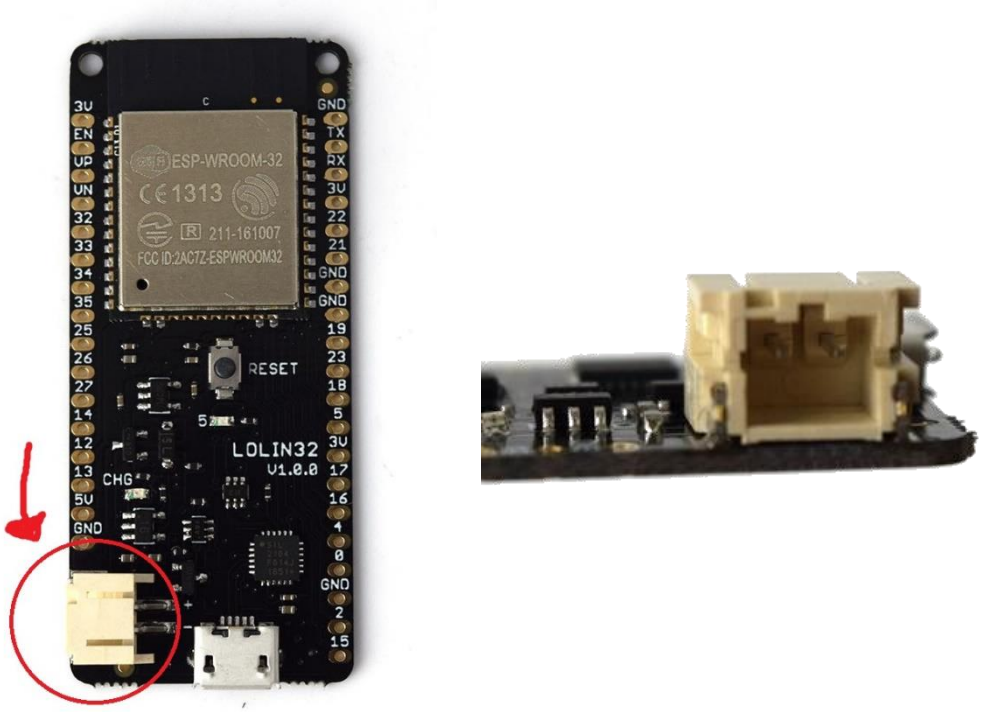

FIGURE 28 PICTURES SHOW WHERE THE PORT OF THE POWER CAN USE TO POWER THE SYSTEM WITHOUT USB

We can use a power circuit that allows us to use power supply with Rechargeable Battery Backup in same time, this circuit uses three diodes, power supply source and rechargeable battery.

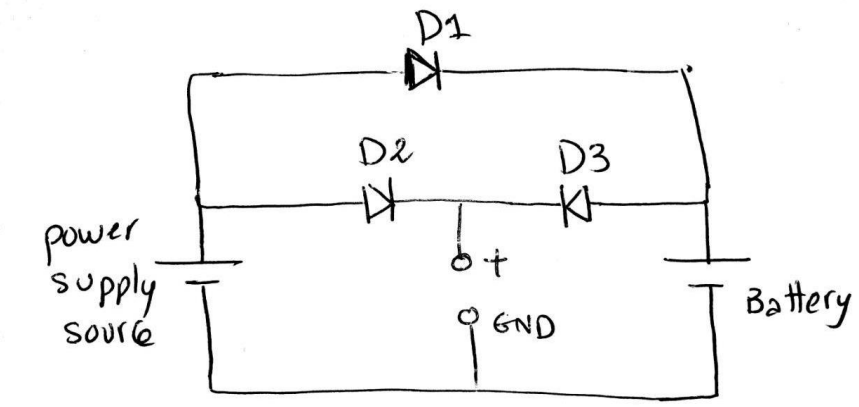

FIGURE 29 SYSTEM POWER CIRCUIT WITH RECHARGEABLE BATTERY

The power supply provides power to the device through D2, or the battery provides power through D3, depending on which power source has a higher voltage. If the battery discharge, the power supply flows through D2 to power the device and through D1 to recharge the battery. If the power supply disconnect, the battery power the device.

2. Integrated and inclusive development

To make a full system, the first step we need to think about it is to make a scenario of how the user use the hardware, after this we can create other a scenario that combine between the buttons and what the user see in the TFT LCD and create a code that we call it the body of the firmware, after that we can integrate the modules to the hardware with add its small codes into the body of the firmware.

2.1. The user scenario

The user has four buttons that can use it to move between the interfaces of the hardware, this buttons are DOWN button, UP button, OK button and EXIT button. And to track the user, or to know where the user when he moves between the interfaces, we use a flag number, this flag changes when the user changes the interface, so if the user is in page one the flag is 1 and if for example the user presses the button DOWN and user navigates to page two and the flag changes to 2.

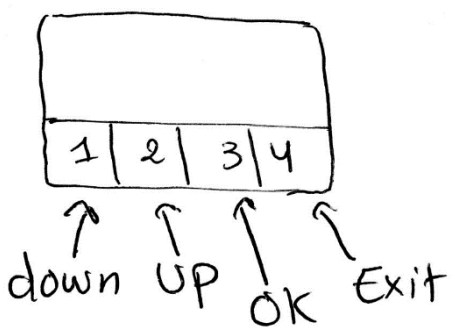

FIGURE 30 THE FOUR BUTTONS THAT CAN USER USE IT TO MOVE BETWEEN THE INTERFACES

The following hand drawing explains how to use an idea of flag:

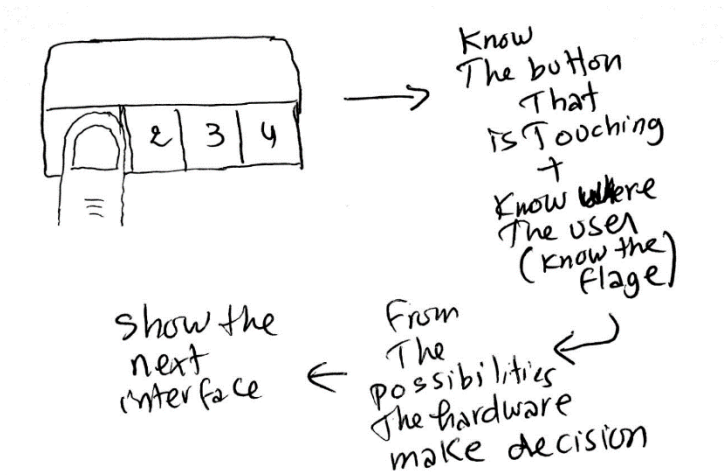

FIGURE 31 A SIMPLE PARTIAL EXPLANATION OF ONE EXAMPLE OF A USER SCENARIO

The full scenario with the flags, and what button should we press to pass to the other interface in this diagram:

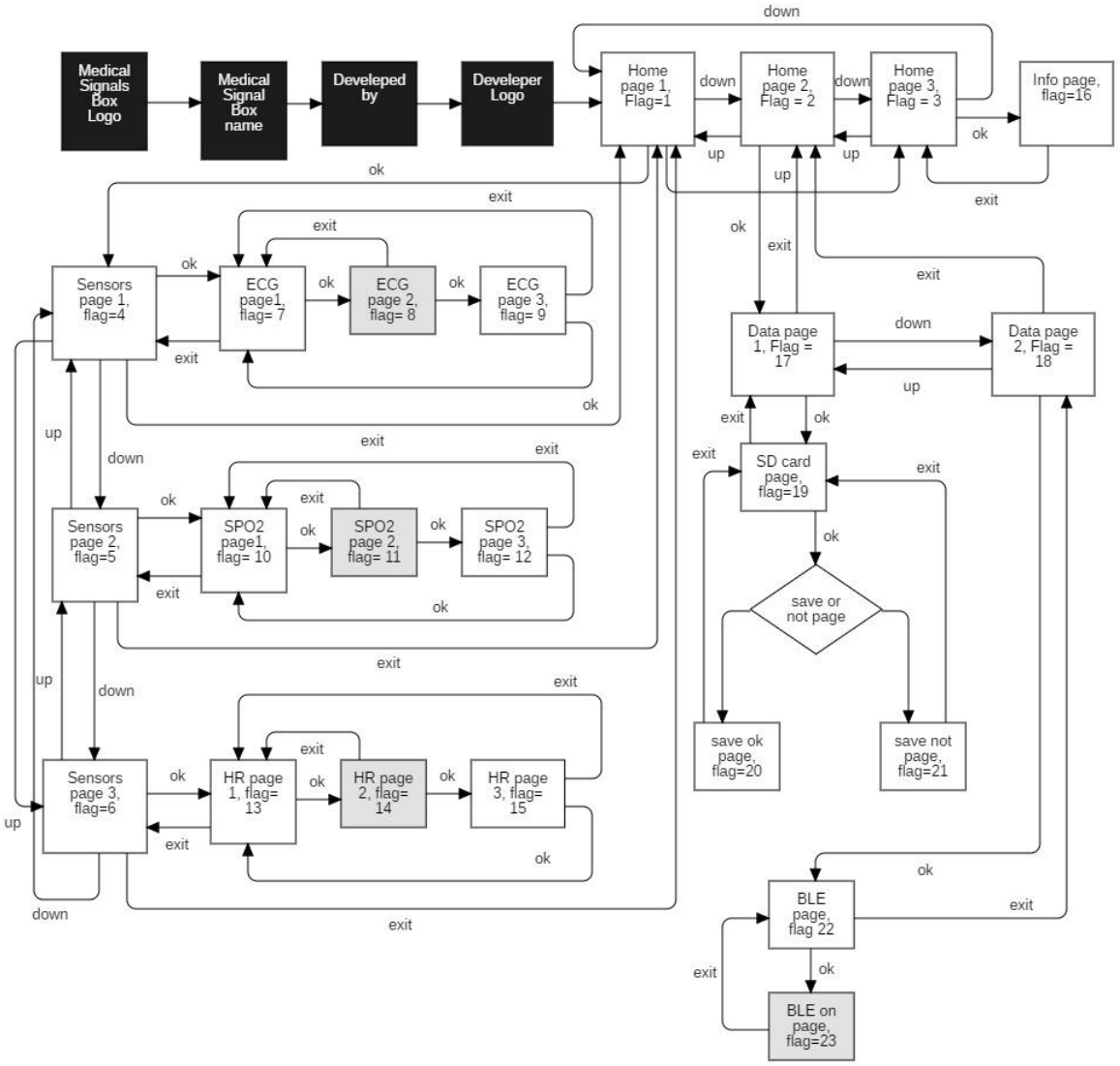

FIGURE 32 THE USER SCENARIO WITH THE BUTTONS AND INTARFACE FLAGES AND NAMES

In this next step, we need to make the interfaces, we have two methods: make the interfaces by code or make the interfaces as images. To make the development faster, we chose the second method, and we use Adobe Photoshop to make this interfaces, this method is not good when the development is not done or the scenario is not defined yet. Because for example, if we develop an interface showing that there is two sensors we can use, and after that we add another sensor, we will need to update all the interfaces that show the list of the sensors, but when we use the first method and code the interfaces, we can easily change the code to develop the interfaces. The disadvantage of the make the interfaces by the code is the design, coding an interface is so hard to personalize, in the software development it is easy but in an embedded system it is so hard, that's why we chose the second method.

This figure show all the images that we make it use interfaces:

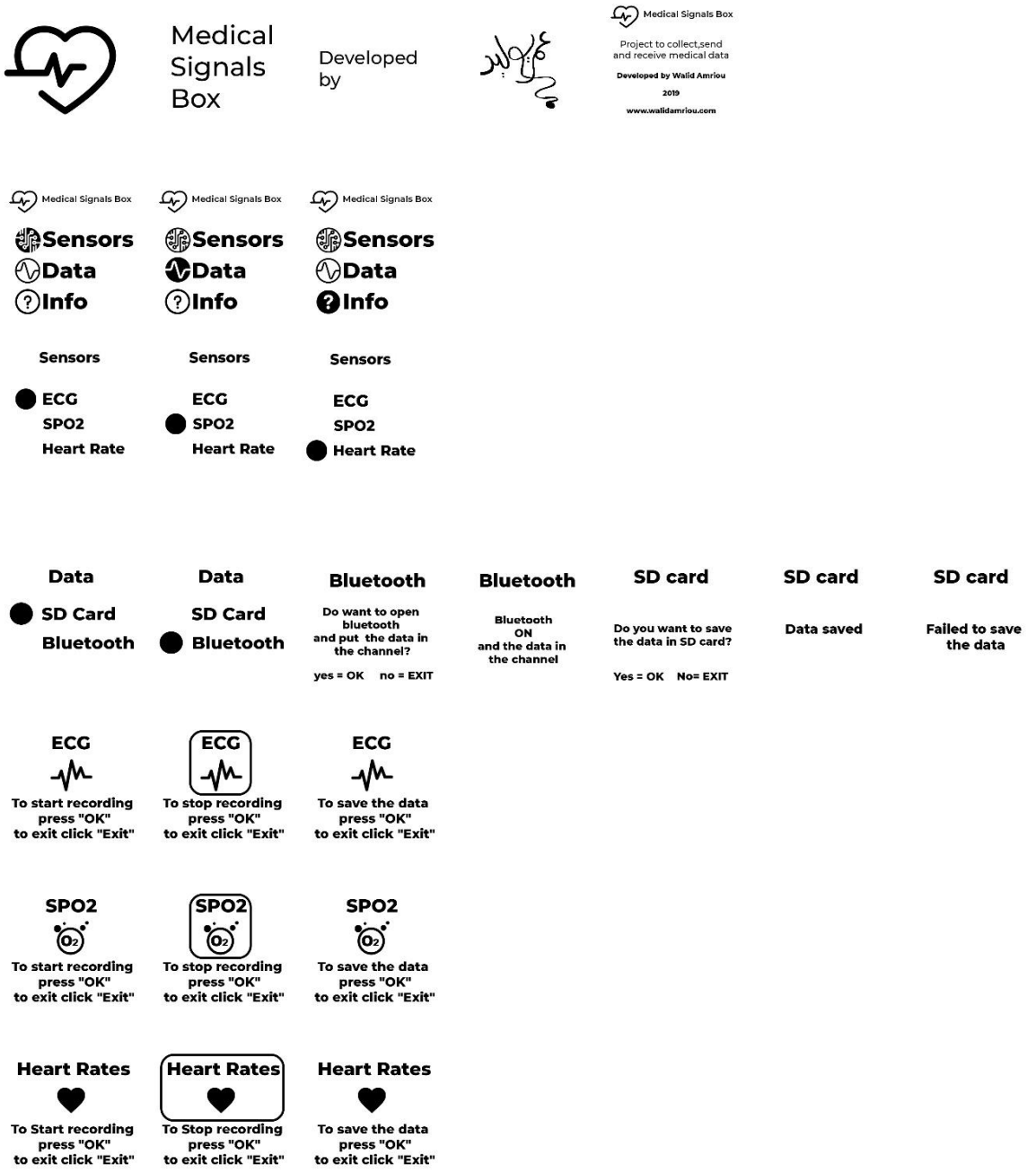

FIGURE 33 THE INTERFACES OF TFT LCD THAT WE USED IN THE PROTOTYPE V1

2.2. Notes when integrate the modules to build a complete system for the prototype version one

By following the partial development, the integration of the modules with the system is easy, we define all the pins where we need to put them and how the communicate with the ESP Lolin 32. When we developed the firmware's body (The user scenario code), we noted comments where we needed to put the codes of the modules, and when we connected one we put the code in this firmware.

In this table, we summarized all the linked between the pins of ESP Lolin32 and the modules:

TABLE 8 TABLE SUMMARIZED THE LINKED BETWEEN THE MODULES AND ESP LOLIN32

ESP Lolin32 pin number	ESP Lolin32 pin name	Module or Modules
GPIO01	TXD	
GPIO03	RXD	
GPIO22	I2C SCL	MAX30102 SCL MPU-6050 SCL
GPIO21	I2C SDA	MAX30102 SDA MPU-6050 SDA
GPIO19	SPI MISO	Micro SD card reader MISO
GPIO23	SPI MOSI	TFT SDI Micro SD card reader MOSI
GPIO18	SPI SCK	TFT CLK Micro SD card reader SCK
GPIO5	SPI SS	
GPIO17		TFT CS
GPIO16		Micro SD card reader CS
GPIO4	ADC	
GPIO0	ADC	
GPIO2	ADC	TFT RS
GPIO15	ADC	TFT RST
GPIO36	ADC	
GPIO39	ADC	
GPIO32	ADC	AD8232 LO+
GPIO33	ADC	AD8232 LO-
GPIO34	ADC	
GPIO35	ADC	
GPIO25	ADC	AD8232 OUTPUT
GPIO26	ADC	
GPIO27	ADC	TTP224 OUT4
GPIO14	ADC	TTP224 OUT3
GPIO12	ADC	TTP224 OUT2
GPIO13	ADC	TTP224 OUT1

- **Problem of ILI 9225TFT and SD card reader**

The ILI 9225 TFT and SD card reader use the same SPI bus interface to communicate with the ESP Lolin32, the SPI separated by the chip select, but this doesn't work in the prototype v1. We try many things but after that we decide to unintegrated the SD card reader from the prototype, because we think the problem is when the ILI 9225 TFT start the main function that prepared to use it, it activate the hardware SPI and it is something that faster but use SPI pins specific to each board, for that when the hardware SPI activate we cannot use SD card reader, the ILI9225 TFT mastery the SPI bus.

After we analyse the problem, we think that the main problem in the code of the ILI 9225 TFT Arduino library, we tried to read all the library and test disabling the Hardware SPI but couldn't fix the problem while not seeing any solution. We searched about this problem without any chance.

The best solution is to make a bridge by a microcontroller with ILI9225 TFT and just send the command to this microcontroller and the microcontroller controls what is showed in ILI9225.

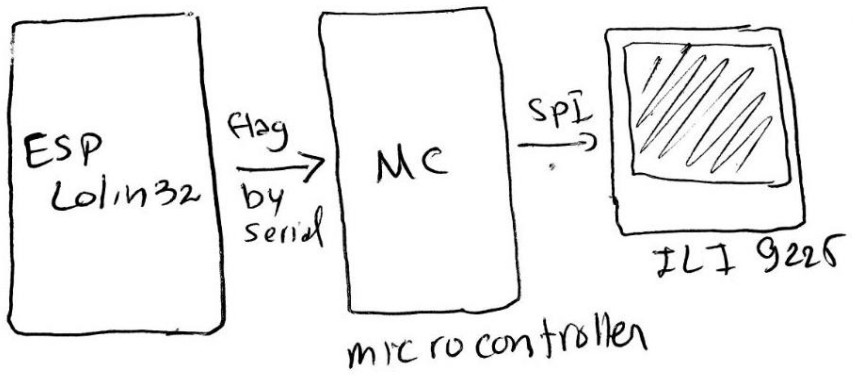

FIGURE 34 A THEORETICAL FORM TO SOLVE A PROBLEM OF TFT WITH SPI

The linked between the ESP lolin32 and the microcontroller (example Arduino Nano) it is done by Serial Communication (Rx, Tx communication), the wires of this communication it just two: receiver Rx and transmitter Tx. In ESP Lolin32 the Rx in GPIO03 and Tx in GPIO01.

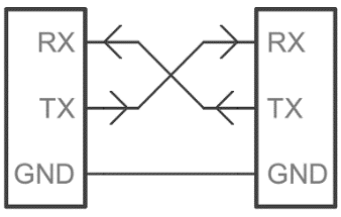

FIGURE 35 HOW TO LINKED BETWEEN TWO DEVICE WITH SERIAL

3. Test the hardware of the prototype version one

In this section, we will show the pictures with the test of the prototype version one with the modules.

This picture shows the full system of the prototype version 1:

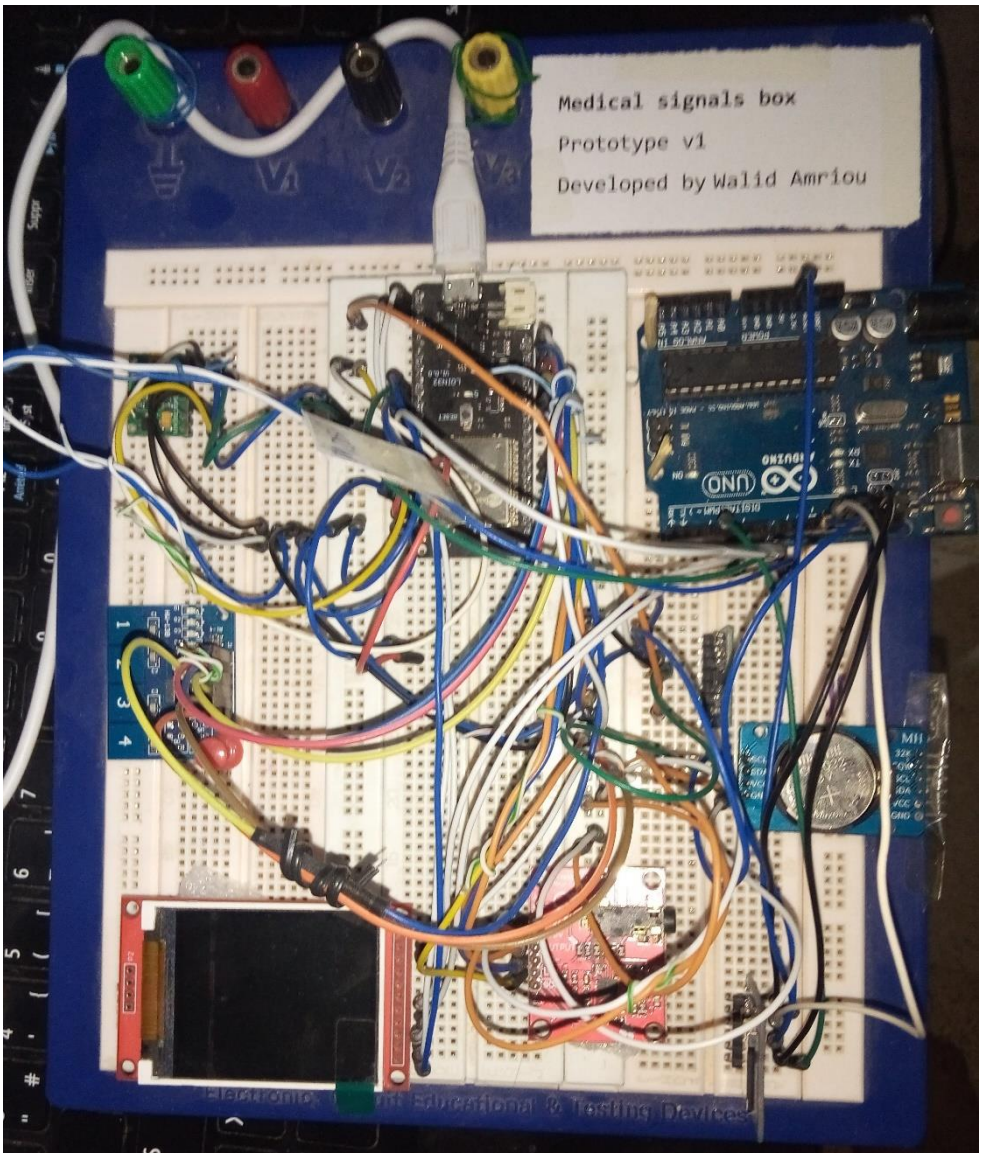

FIGURE 36 PICTURE OF THE FULL PROTOTYPE SYSTEM OF PROTOTYPE VERSION 1

This picture shows the home page in the TFT LCD, where the user can choose between sensors, data and info:

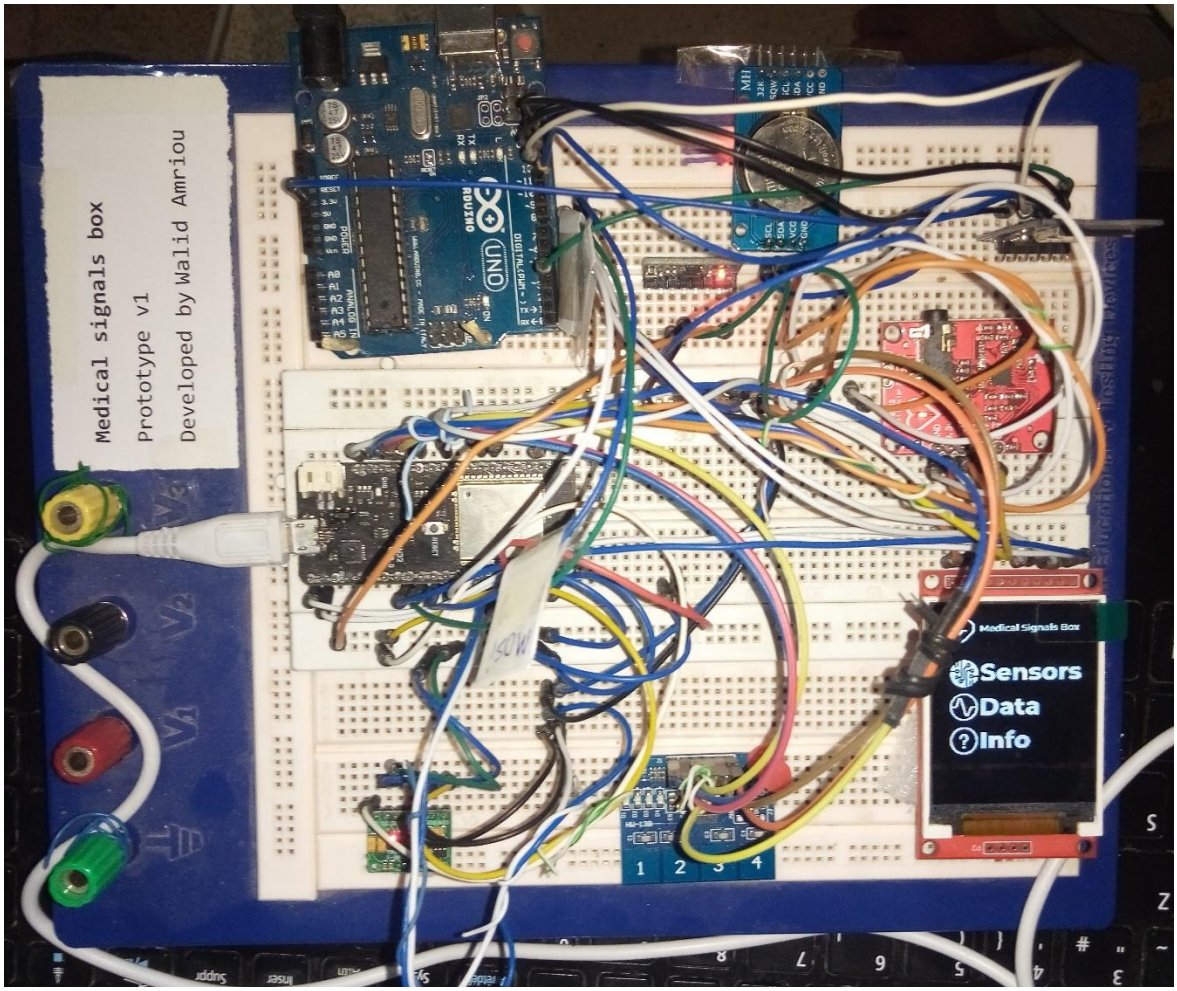

FIGURE 37 PICTURE SHOWS THE HOME PAGE IN THE TFT LCD

The info page has the info of the project and developer, in the other hand we can add in the info page the version of the system and the update (if we ever decide to connect the system to the cloud or internet using OTA):

FIGURE 38 PICTURE SHOWS THE INFO PAGE IN THE TFT LCD

This picture shows the list of sensors in the TFT LCD:

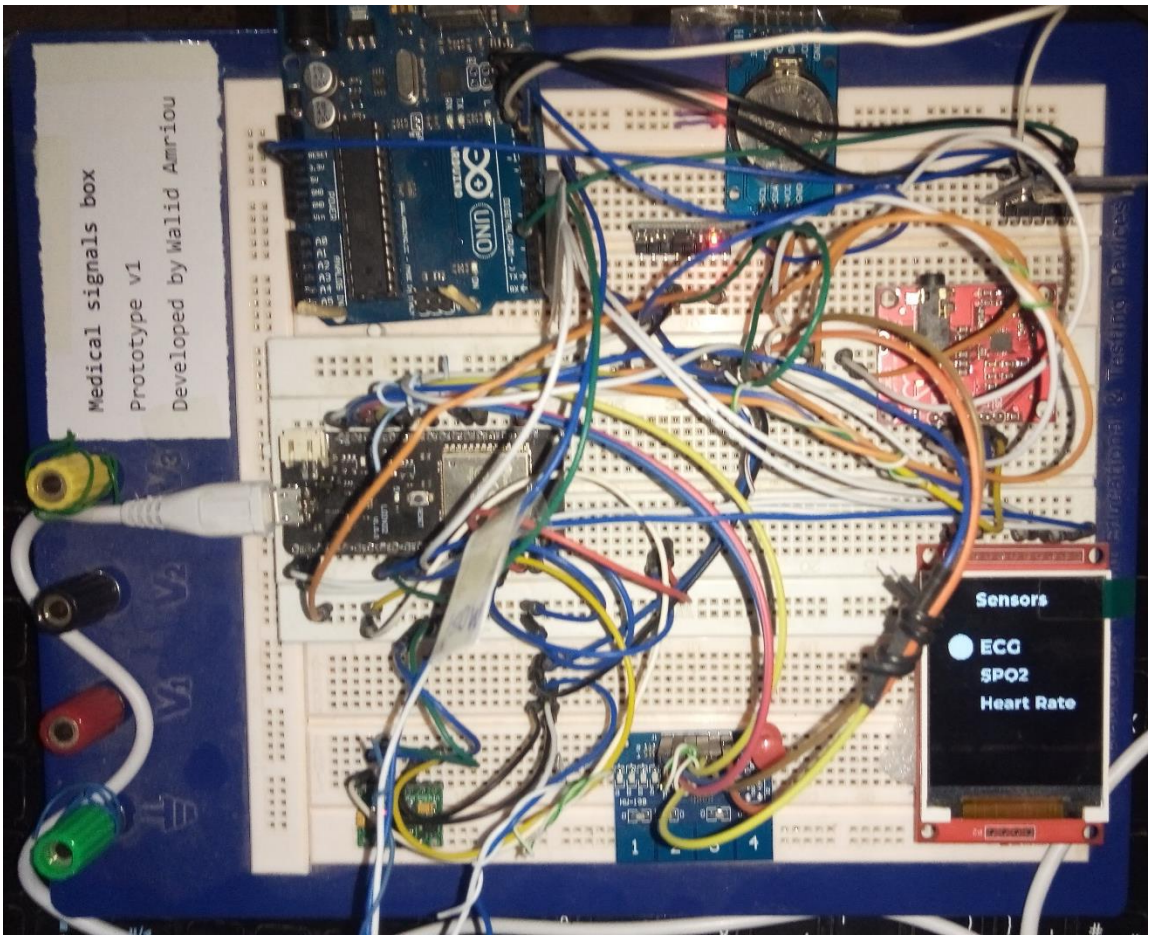

FIGURE 39 PICTURE SHOWS THE LIST OF THE SENSORS IN THE TFT LCD

We cannot add all the pictures of the test, because the number of the interfaces pages is 23 page.

4. Test the Software of the prototype version one

We said that we will develop the Software of the prototype just in Linux, but we develop the receive from BLE in Linux Ubuntu 18.04 and the saver of the data received to database in macOS (because we get a help from an engineer use macOS).

This is the result from the test of receive in Linux Ubuntu 18.04:

FIGURE 40 SCREENSHOT THAT SHOWS THE TERMINAL WINDOW THAT SHOW THE RESULT OF RECEIVE THE DATA FROM THE HARDWARE

This is the result from the test of save the data received from the Bluetooth in macOS:

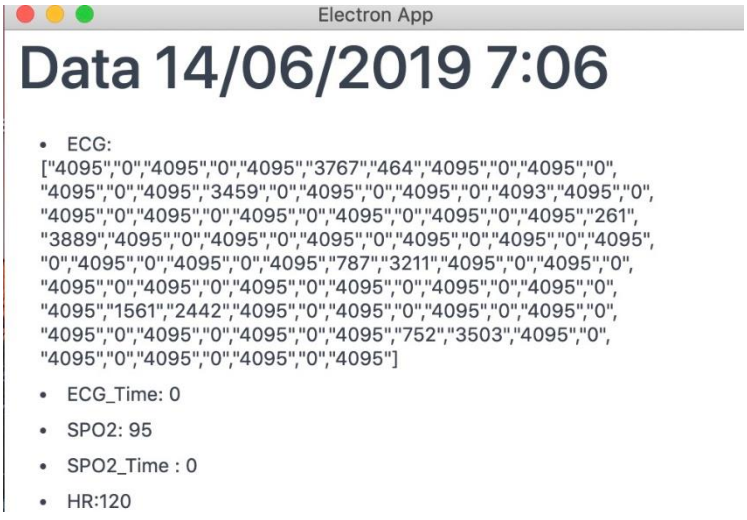

FIGURE 41 SCREENSHOT FROM THE WINDOWS OF THE PROTOTYPE OF THE SOFTWARE OF THE SYSTEM THAT SHOWS THE RESULT OF THE RECEIVED FROM THE BLUETOOTH

The file that create like a database intern (JSON file):

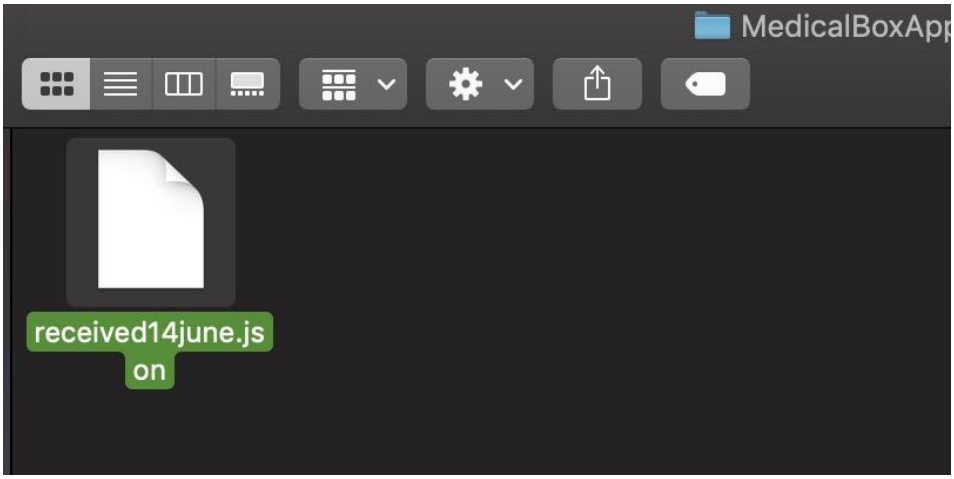

FIGURE 42 SCREENSHOT SHOWS THE FILE JSON THAT SAVED THE DATA IN IT

The content of the JSON file that created:

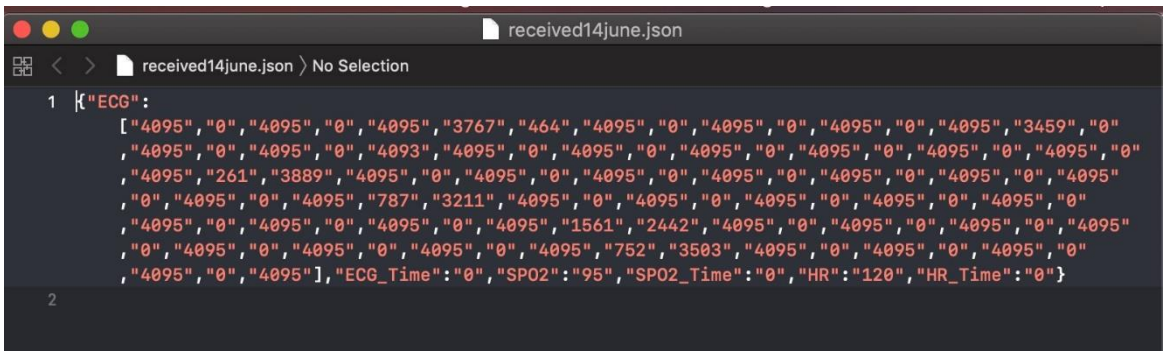

FIGURE 43 SCREENSHOT SHOWS THE CONTENT OF THE JSON FILE.

Conclusion

In the beginning of this chapter, we had the full design of the prototype of the system, from the software and the hardware, and in the chapter, we talked about how we developed codes of the firmware and what are the problems that we found in our way.

After develop the so-codes of the firmware, we put it in the body of firmware (the scenario). And in the last papers, we put the pictures of the test.

The pictures of the test show that we succeeded to make the prototype version 1 of the project.

General conclusion

Our work in this project is to design a platform that provides continuous monitoring at any time to patients and the injured. A platform aims to provide a device easy to use, portable, that interacts with mobile devices, computers and cloud. In addition, the platform can be a device or platform for students and developers. After designing it, we made a prototype based the design.

To do that, we used one of our methodologies of designing an embedded system project. Before that, we understood the field of the work, and then we started directly to our methodology. First, we made a mock-up from the idea of the project. Next, we started working to make the specification design. Next, from the specification design we created the architecture and function design. Next, from the architecture and function design, we created the deep design and it is the last look to our project. Second, from the last design (deep design), we divided the full system into subsystems, we developed these subsystems one by one, after that we collected them all into one system.

In the part of design and realization of a prototype of the main design, we used many modules in the hardware part such as:

- ILI 9225 TFT LCD.
- ESP LOLIN32 microcontroller.
- MAX30102 Pulse Oximeter and Heart-Rate Sensor.
- TTP224 Switch Touch Sensor Digital 4 channel Touch Capacitive Module.
- AD8232 SparkFun Single Lead Heart Rate Monitor.
- MPU-6050 Accelerometer Gyroscope module Sensors.
- DS3231 real-time clock (RTC) module.
- Micro SD Card Adapter (Catalex).
- Arduino Uno.

In software part and firmware development:

- Node.js (Core of the software)
- Electron software framework (to make a server of Node.js and interfaces of the software).
- Arduino programming language (Firmware).
- C++ (Help Arduino programming language to develop the firmware)

When we compared our work with others from the market, our project has the following strengths:

- The possibility of using many medical sensors or integrate others in the future by I2C or SPI protocols.
- The possibility to send the data record via Bluetooth low energy (BLE).
- Special software for the project, which can receive the data from the hardware via BLE and save it to a database.
- The system consumes low energy because it uses BLE.
- The system is reconfigurable.
- The software of the project works with Modern and powerful technology.
- Open source project and developed by a community.

- Low production cost and suitable for developing countries.
- It has backup battery.
- The system is scalable.

The problems and Challenges that limited our realization of the prototype version 1:

- Absence of tools providing ideal data like ECG signal generator.
- Chinese-made and cheap modules that have caused problems in knowing the correct data of the wrong.
- Problems in library codes of the modules that take time to fix it, like the ILI 9225.TFT LCD Arduino library use the SPI hardware, and make the problem that you cannot use the SPI with other modules, so if you use TFT you can use the SPI bus.
- Problems with some of the popular programming languages we like to use like JAVA with Bluetooth.
- Problems in Windows philosophy with hardware, it is so hard to develop a great software that get the data from Bluetooth without using C#.
- There are no simulations for many modules like ESP Lolin32, for that we worked directly in the reality by cautiously.
- And more.

In the development, we tried to fix all the problems and this is the variable that killed our time, some problem we cannot fix it normally like the problem of TFT with SPI, that's why we used Arduino Uno to communicate with SD card reader that use the SPI.

To develop this work we suggest:

- Working in an open and wide development community.
- Using original tools, modules and sensors to get real signals and data.
- Developing the software of the system in Windows with C# and Microsoft Visual Studio to allow using APIs.
- Developing the software of the system in Linux, macOS and android, based Node.js to speed the development.
- If you use the same TFT module, place the TFT in a small microcontroller that has a serial connection (RX, TX) and SPI protocol.
- Use a medical signal generator to get ideal signals to develop great system.
- Build the Arduino library for the project, but pay attention to solve the problem of moving the variables of the TFT to .cpp file, because this variables is constant and for that when you will put it in the .h file the linker did not found it and the firmware crash (The linker is a program that makes executable files).

Bibliographies & Webographies

Chapter I Health field

[1]	«businessdictionary,» 24 april 2019. [En ligne]. Available: http://www.businessdictionary.com/definition/health-care.html. [Accès le 24 april 2019].
[2]	«Healthcare Domain,» [En ligne]. Available: http://www.technofunc.com/index.php/domain-knowledge-2/healthcare-industry. [Accès le 27 may 2019].
[3]	F. Barbabella, M. Gabriella, M. S. Quattrini, R. Papa et G. Lamura, «How can eHealth improve care for people with multimorbidity in Europe?,» NIVEL and TU Berlin, Berlin , 2016 .
[4]	D. Industrial, «Delkin devices,» [En ligne]. Available: https://www.delkin.com/blog/using-embedded-systems-in-healthcare/. [Accès le 2019 April 24].
[5]	«circuitglobe,» [En ligne]. Available: https://circuitglobe.com/difference-between-sensor-and-transducer.html. [Accès le 26 april 2019].
[6]	«codrey,» [En ligne]. Available: https://www.codrey.com/electronics/different-types-of-sensors/. [Accès le 26 April 2019].
[7]	D. Leader, «verywellhealth,» 2018 November 21. [En ligne]. Available: https://www.verywellhealth.com/oxygen-saturation-914796. [Accès le 26 April 2019].
[8]	«Body temperature norms - Medlinepluse,» A.D.A.M., Inc, [En ligne]. Available: https://medlineplus.gov/ency/article/001982.htm. [Accès le 09 may 2019].
[9]	«omega,» omega, [En ligne]. Available: https://www.omega.co.uk/temperature/z/thermocouple-RTD.html. [Accès le 03 may 2019].
[10]	P. A. I. (eds.), Handbook of Cardiac Anatomy, Physiology, and Devices, Springer International Publishing, 2015.
[11]	P. Bryn Farnsworth, «What Is ECG and How Does It Work?,» January 15th, 2019, 15 january 2019. [En ligne]. Available: https://imotions.com/blog/what-is-ecg. [Accès le 19 june 2019].
[12]	D. A. Mondry, «Patient monitoring and care,» [En ligne]. Available: http://www.bii.a-star.edu.sg/docs/education/lsm5191_03/notes/02%20PatMonCarSys.pdf. [Accès le 19 june 2019].
[13]	«semiconductorstore,» [En ligne]. Available: https://www.semiconductorstore.com/patient-monitoring-system/A/51. [Accès le 19 June 2019].
[14]	«MySignals HW v2 - eHealth and Medical IoT Development Platform for Arduino,» [En ligne]. Available: https://www.cooking-hacks.com/mysignals-hw-ehealth-medical-biometric-iot-platform-arduino-tutorial/. [Accès le 29 may 2019].

Chapter II Embedded system design

[1]	U. Aastik and S. D. Abhimanyu , "Embedded Systems And its Application in Medical Field Aastik Upadhyay and Abhimanyu S.Dhapola," in *Emerging Trends in Computer Science and Information Systems(NCETCSIS-2015)*, New Delhi, 2015.
[2]	B. Michael , Programming Embedded Systems in C and C++, O'Reilly Media, 1999.
[3]	R. Keim, "What Is Embedded System Design? Defining an Electrical Engineering Field," [Online]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-embedded-design-embedded-system-design-firmware/. [Accessed 10 may 2019].

[4]	V. Frank and G. Tony , Embedded System Design: A Unified Hardware/Software Introduction, Frank Vahid and Tony Givargis, 2002.
[5]	"Real-Time Embedded Systems," [Online]. Available: https://www.tutorialspoint.com/Real-Time-Embedded-Systems. [Accessed 05 may 2019].
[6]	"A Brief About Embedded System their Classifications and Applications," [Online]. Available: https://www.efxkits.us/classification-of-embedded-systems/. [Accessed 28 may 2019].

Chapter IV The detailed design

[1]	"prototype," [Online]. Available: https://dictionary.cambridge.org/dictionary/english/prototype. [Accessed 25 may 2019].
[2]	"the internet of things with ESP32," [Online]. Available: http://esp32.net/. [Accessed 25 may 2019].
[3]	"2.0inch Arduino SPI Module ILI9225 SKU:MAR2001," [Online]. Available: http://www.lcdwiki.com/2.0inch_Arduino_SPI_Module_ILI9225_SKU:MAR2001. [Accessed 29 may 2019].
[4]	"MAX30102," [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf. [Accessed 29 may 2019].
[5]	"MPU-6050 Accelerometer + Gyro," [Online]. Available: https://playground.arduino.cc/Main/MPU-6050/. [Accessed 30 may 2019].
[6]	"DS3231 Extremely Accurate I²C-Integrated RTC/TCXO/Crystal," [Online]. Available: https://www.maximintegrated.com/en/products/analog/real-time-clocks/DS3231.html. [Accessed 20 may 2019].

Chapter V Realization

[1]	“SPI library,” [Online]. Available: https://www.arduino.cc/en/Reference/SPI. [Accessed 10 june 2019].
[2]	“I2C Info – I2C Bus, Interface and Protocol,” [Online]. Available: https://i2c.info/. [Accessed 11 June 2019].
[3]	R. Pelayo, “What is I2C? | Protocol Guide,” [Online]. Available: https://www.teachmemicro.com/i2c-primer/. [Accessed 11 June 2019].

Annexes

- **The source of project in GitHub:**

https://github.com/walidamriou/MedicalSignalsBox

- **We had put the project source of the prototype version 1 in GitHub:**

https://github.com/walidamriou/MedicalSignalsBox/tree/master/Prototype_v1

- **Codes of partial development of prototype version 1 in GitHub:**

https://github.com/walidamriou/MedicalSignalsBox/tree/master/Prototype_v1/Hardware/Hardware_test

- **The datasheets of the modules of the prototype version 1:**

https://github.com/walidamriou/MedicalSignalsBox/tree/master/Prototype_v1/Hardware/modules

- **The firmware of the prototype version 1:**

https://github.com/walidamriou/MedicalSignalsBox/tree/master/Prototype_v1/Hardware/Firmware

- **The software of the prototype version 1:**

https://github.com/walidamriou/MedicalSignalsBox/tree/master/Prototype_v1/Software