POLITECNICO DI TORINO

MASTER OF SCIENCE IN MECHATRONIC ENGINEERING
DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)

DESIGN AND DEVELOPMENT OF A WEARABLE DEVICE FOR CONTINUOUS MONITORING OF VITAL SIGNS

Supervisor
Guido Masera

Co-Supervisors
Mahmoud Tavakoli

Candidate
Detjon Brahimaj - 252846

ACADEMIC YEAR 2019/2020

Alla mia famiglia,
che mi ha sostenuto ed incoraggito,
vedendomi crescere in ogni mia decisione.

Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza"

A Nora,
dolce compagna,
amante preziosa.

ACKNOWLEDGEMENT

Arrived at the end of this period of my life, I want to express my gratitude to all the people that encourage me, especially during this thesis work.

Thanks to the precious support of my father Arturo, my mother Suzana and my sister Ana that encouraged and supported me in these years and, in general, in my life. Thanks to Luisella for the help, the encouragement and for being my second loved mom.

Thanks to the guys of the ISR laboratory, Daniel F.F.F. and André Silva for their friendship and encouragements. To Pedro Lopes that helped me during the development process and to João Vilarinho that printed for me the device case.

To all my friend in Turin that weren't with me in Coimbra during the thesis project, but that were with me during all these years.

Last but not least, I want to thank my girlfriend Nora, who demonstrates patience and love during these last months of intense work in the lab. Thanks to be so special and for making me be a dad.

ABSTRACT

Wearable devices are an emerging trend in different subject areas. In the market there is a wide range of wearable devices for fitness, but researches in healthcare areas are rapidly increasing. The spread of these devices introduces the possibility to have patient monitoring directly from home. Indeed, in some instances like Obstructive Sleeping Apnea (OSA), the patient is forced to stay in the hospital for a long period in order to monitor, for example, the heart rate. The scope of hospitalization is to collect data, while the analysis is performed when the patient is no more required to be in the hospital. This and the recent discovery of a new bio-signal for OSA detection based on body temperature measurements, are the motivation and the guidelines of this work.

To respond to the necessity of continue monitoring of temperature, this work aims to design and develop a possible wearable device able to accomplish different tasks related with OSA disorders. At first, an overview of wearable devices available in the market will be considered and then, the design and the applied methods will be discussed. Finally, a deep view of the firmware will be performed for the microcontroller and the Graphic User Interface (GUI).

As a result, the device has been tested showing precise and accurate measurement of temperature but poor value of heart rate and blood oxygen levels. Furthermore, it has been tested for a 72 hours window showing the capability to acquire samples for the required amount of time and allowing the possibility to use these data for OSA detection algorithms.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

CHAPTER 1

INTRODUCTION

In the last years, interest in wearable devices has rapidly increased. In particular, wearable health devices (WHDs) have gained a lot of attention since show powerful possible applications. This area is important since opens the possibility to have patient monitoring, but also because they reduce medical analysis cost, waiting times and the hospital's patients load.
To have an idea of the impact that WHDs has in the market, the worldwide revenue was 34 billion dollars in 2019 and healthcare and medical environments represents more than 40% of the total revenue.
These devices can be used for a lot of different environments that require these kind of purposes:

- Ambulatory acquisition and monitoring of vital signs over extended periods (days/weeks).
- Sport, activity and fitness monitoring for athlete's performance before, during and at the end of competition or test.
- Soldier or workers evaluations in different hazardous situation and further improvement to better accomplish their efforts.
- Help the recovery from body injury or similar body situations
- Improve health monitoring in underdeveloped countries.

Technology play a key role in devices research and development.
These devices are a combination of different science disciplines such as

micro and macro nanotechnologies, materials and biomedical technologies and last but not least electronics and communication technologies. Indeed, miniaturization of electronics components and devices is allowing the possibility to design and develop more reliable and adapted WHDs. This is changing the idea of health monitoring worldwide, becoming a crucial healthcare revolution. Traditional monitoring tests require the patient to be physically in the hospital or laboratory, requiring a huge amount of time that can be described as the sum of: waiting list, time to reach the hospital/laboratory and analysis time.

Furthermore, the possibility to monitor vital signs' patients from home, with a moderate cost of the devices, allows the possibility to identify more serious health problem earlier. This is due to the great amount of data collected and the possibility to use powerful detection algorithms that give results in a short period of time.

In general, wearable devices(WDs) should have a few common key properties in order to be a successful market product:

- Cost is crucial because allows or not the possibility to use WDs in a large scale or in underdeveloped countries.
- Low power consumptions to ensure continuous monitoring for a long period of time.
- Patient comfortability to make the WD easy to be applied and "invisible" during monitoring from the patient point of view (the patient should do its daily routine ideally without any limitation due to the device.

These are just a few requirements; more will be discussed in the next sections, while the reader is invited to see [1] for more details.

For WHDs some requirements are more stringent. Indeed, health care monitoring should follow national/international laws and standards, especially in vital signs measuring accuracy, precision and reliability. For this

reason and also for what regards power consumptions, electronic components selection represents a crucial step during design.

CHAPTER 2

MEASURING VITAL SIGNS

This chapter intends to give a brief introduction on obstructive sleeping apnea (OSA), explaining what it is, how this disease is detected and which are the methods used to treat it.
After that an introduction of the principles behind non invasive vital signs monitoring will be given, exploring body anatomy and organs involved, explaining physical and mathematical methods currently used.
A new biomarker, that represents the main motivation of this dissertation, will be introduced and explained.
The discussion will be general in some cases while a focus on these arguments will go around the vital signs involved in the measuring device that will be developed and discussed in Chapters 4 and 5.

2.1 Obstructive Sleeping Apnea

The motivation of this work comes from the discovery of a new biomarker for OSA detection that will be discussed in section 2.2. Based on this discovery, the developed device has the scope to collect data for a certain amount of time and store them. For these reasons it is important to have an in deep view of OSA disorder.
The obstructive sleeping apnea is one of the most common sleep disorders. It is characterized by a cessation or reduction of air flow and causes breathing to constantly stop and start during sleep.

It is called obstructive because when it occurs, the throat muscle spasmodically relaxes and block the airway. In order to be considered an apnea event, the airflow must be blocked or strongly reduced for a few seconds. After that period the normal breathing is restored.
Another case of OSA is due to a structural anomaly in the mandible or in the airway that provokes apnea events.
A common sign of OSA is snoring: the air flow passes relaxed tissues in the throat causing the tissue to vibrate while breathing and all these result as the typical noise of snoring.
Snoring is just one of the typical complications of OSA; more serious are heart problems or hypertension. Consecutive low values of blood oxygen levels during apnea events increase the blood pressure and stress the cardiovascular system. This increases the fatigue on the heart and might also increase the probability to have other cardiovascular disorders such as abnormal heartbeats or heart attacks.
Another important problem with OSA is the daytime fatigue. The frequent awakening associated with sleeping apnea events makes sleeping very hard, producing irritability and severe fatigue during all day. The quality of life is reduced while stress levels are increased.
Between 20 and 40 years old the percentage of people diagnosed with OSA is less than 4% while between 65 and 100 years the percentage are greater: 18.1% for males and 7% for female. Unluckily, more than 80-90% of people are undiagnosed and the occurrence of people that have OSA should be higher [2].

2.1.1 OSA: Treatment

The typical treatment for OSA consists in a few steps. The first one is to adopt changes in the lifestyle in order to mitigate symptoms associated with OSA. Indeed, lose weight, avoid alcohol and quit smoking are the first

thing to do to start a healthier life.
The cause of sleep apnea disorder are multiple but, in some cases, the unique treatment is a medical surgery, such as upper airway surgery to remove tissue in the airway. In the majority of cases, the treatment of choice is the Continuous Positive Airway Pressure (CPAP) device.
It is a respiratory ventilation machine used in different sleep apnea disorders, in patients that have severe respiratory failures, including babies who are born premature.

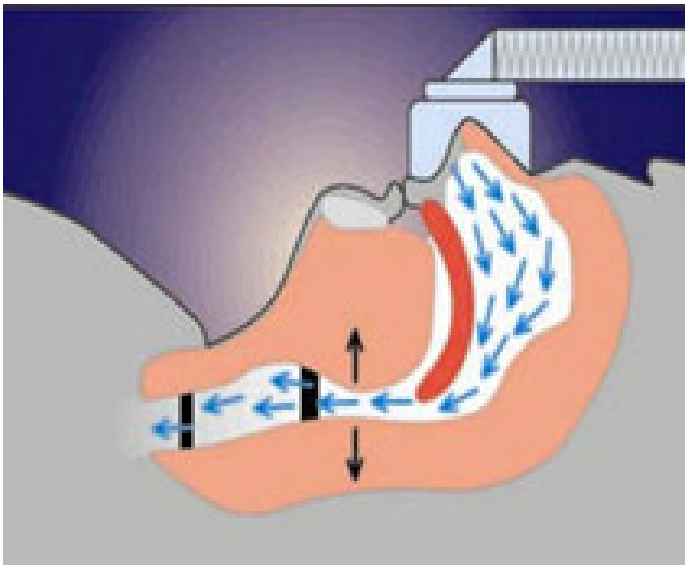

Figure 2.1: Continuous positive airway pressure device gentry pumping air into the airway

The CPAP is a mask that fits the nose and/or the mouth of a subject and then gently blows air into the airway to help keep it open during sleep, as showed in Figure 2.1.
This method is highly effective and, for OSA positive subjects, it is recommended to use the CPAP whenever they sleep. The health profits of this treatment are enormous when this device is used correctly. Indeed, it is possible to obtain a reduction of obstruction episodes per hour of sleep from $47.8(\pm 5.4)$ to $1.2(\pm 0.2)$ in four months of CPAP treatment (the mask was used every time the subject goes to sleep).

2.2 Body temperature

An important vital sign is represented by the body temperature measurements. Through its values, we evaluate the efficiency of the body thermoregulation.
When we talk about body temperature measurements, it is important to understand the difference between Core Body Temperature (CBT) and body temperature (BT). BT is also called peripheral temperature because it is the measure of temperature of the outside part of the body (skin, muscles, extremities such as arms, etc.) while CBT (a gold standard for rhythmicity studies) is the operating temperature of an organism and it is typically measured with invasive methods such as rectal (or vaginal) temperature that are considered the most accurate. CBT is a typical circadian biomarker that reflects the endogenous rhythm regulated by the hypothalamus.
BT measurements are usually less precise then CBT because the outside part of the body, like the skin, is more exposed to the environment temperature. This can lead to a fast change in the measured temperature with respect the CBT. This increases the possibility to have noise and disturbances while measuring due to the environment. This error depends strongly on where the temperature sensor is positioned in the body. Greater errors for devices positioned in zones that are in direct contact with the environment (like the wrist, the arm, the neck, etc.). Less affected will be the data if the device is positioned in a zone that is usually covered by a tissue(like the chest, the back, the belly, etc. that are covered with a t-shirts or similar).
The University of Coimbra, in collaboration with the city of Coimbra, starts a new study on OSA disorders. Their aim was to prove a correlation between human body temperature and OSA disease. After different experiments and tests, they achieve their goal and temperature becomes an OSA biomarker.
The method used to collect data from the patients consist in asking them

to stay one day in the hospital in order to perform the test. They collected one sample of temperature, from the subject's axilla, every 3/4 hours. The process was then repeated after 4 months and after two years for most of the patients. During this period the OSA positive subjects was asked to use the CPAP every night.

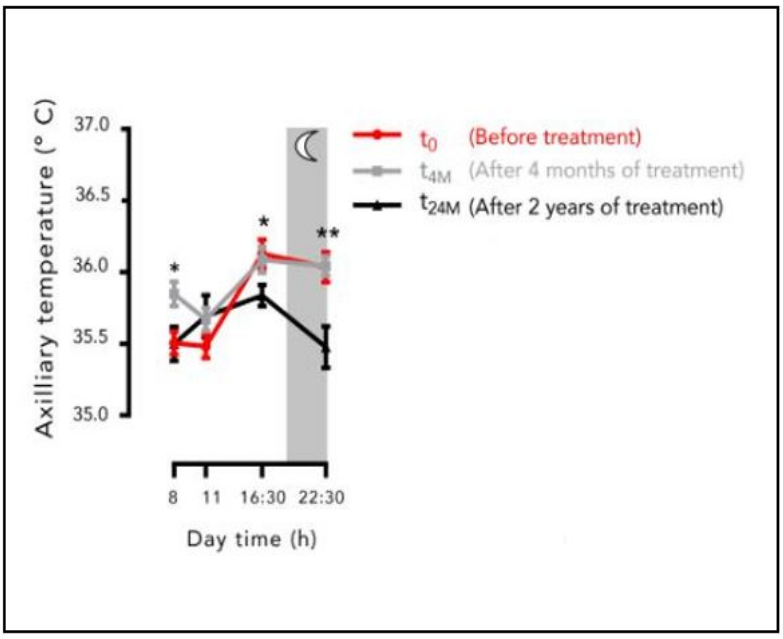

Figure 2.2: OSA impacts on the diurnal profile of axillary body temperature.

The results of this experimental test are shown in Figure 2.2. The OSA positive subjects have a higher temperature in the early morning and during the night. After two year of treatment, the temperature signal changes, demonstrating the correlation between temperature as OSA biomarker and confirming the importance of using the CPAP. For more information the reader is invited to see [2], [3], [4] and [5]. The further step of the project is then to design and prototype a device that respect some specifications like being small, have low power consumption and the capability of 48/72 hours of measuring period.

This represents the motivation and the starting point of this work.

2.3 The Human Heart

According with the Word Health Organization (WHO), heart associated diseases account for a large percentage in the associated mortality rate worldwide. Cardiovascular diseases (CVDs) are the number one cause of death globally, representing 31% of all global deaths, in 2016 according to the WHO [6]. For this reason, it is considered particularly relevant to monitor the health of the heart.

2.3.1 Anatomy

The heart is an involuntary muscle that pumps blood (provides nutrients, oxygen etc. to the body) through the blood vessels of the circulatory system. Structurally it has four chamber, two upper atria (they receive blood from the body) and two lower ventricles (they discharge blood in the body). It is usually divided in right and left side since the two chambers are independent even if they have common structures. Even if left and right cavities have substantial differences due to their role, the fundamental structure of ventricles is similar and the same holds for atria.
A wall of muscle, the septum, divide the left and the right atria/ventricles. An image of this structure is illustrated in Figure 2.3. In the image there are also indicated the pulmonary veins and artery, the superior and inferior vena cava (the largest veins in the body) and the sense of blood circulation. Now a brief view of the working steps of the heart will be considered:

- **Heart: right side**

  The right atrium is positioned slightly below the left atrium, in the upper side of the heart. The deoxygenated blood coming from the body, returns to the heart through the superior and inferior vena cava. In order to help blood accumulation in the cavity, the atria do not have valves at their entrance.

  To prevent blood coming from inferior vena cava to flow down via

gravity, a valve is present in the entrance of the right ventricles.
The right ventricle is situated below the right atria. When the cavity is full, it is responsible to pump the blood in the pulmonary artery through the pulmonary valve, where the blood will exchange carbon dioxide with oxygen.

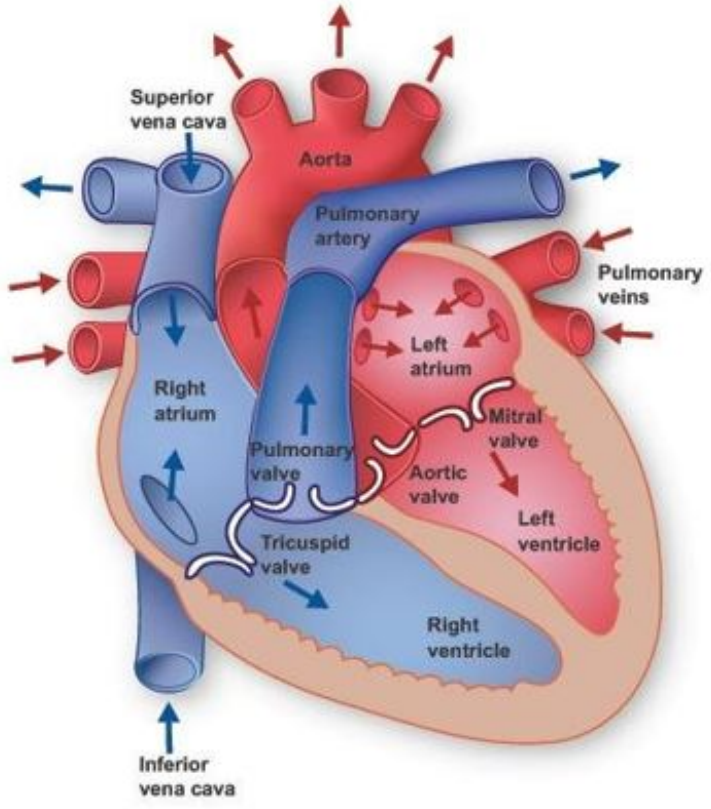

Figure 2.3: Heart's anatomy

- **Heart: left side**
  The left atria, situated in the uppers side of the heart, receives oxygenated blood from the pulmonary veins and push it in the right ventricles through the Mitral valve. When the left ventricle cavity is full, it contracts and pushes the blood in the Aorta artery. From here, the oxygenated blood will return in the body, giving nutrients to all the system.

2.3.2 The Cardiac Cycle

The cardiac cycle can be described starting from the blood that, from the body, arrive in the heart. The deoxygenated blood pass from the right side of the heart where it is accumulated and then pumped in the right ventricle through the tricuspid valve. When the right ventricle contracts, the blood is sent to the lungs via the pulmonary artery. Here, through an interesting process that will not be discussed, the blood is oxygenated again and returns to heart by the pulmonary veins. It enters in the left atria and once passed in the left ventricles, the pumped blood returns to the body. The cardiac cycle terminates when the blood reaches the body's cells, where the oxygen is left, and carbon dioxide taken.
The cardiac cycle presents two specific actions of the heart described in literature. The systole, that is a cardiac contraction, and the diastole, that represents the cardiac relaxation.
From this cycle it is possible to identify four periodic contraction and relaxation steps. Indeed, the cardiac conduction system is responsible for the propagation of electrical signals which control systoles and diastoles. Using the electrocardiogram (EKG or ECG) it is definitely possible to identify all the phases of the cardiac cycle. The ECG is a tool used to measure the electrical activity of the heart.
Analysing the data coming from the EKG, it is possible to obtain information regarding the health of the heart. Even if ECG is considered the gold for heart disease detection and analysis, another important technique is the photoplethysmography (look subsection 2.4.1).

2.4 Heart Rate and Blood Oxygen Saturation Level

Monitoring the heart is very important and, the typical vital signal checked is the heart rate. Another important vital signal, considered in different diseases, is the blood oxygen saturation level (SpO_2).

The technique used for the estimation of the SpO_2 was invented in 1972 by Takuo Aoyagi, a Japanese bioengineer [7]. The success of this technique has to be associated with its non-invasive nature, the possibility to rapidly test a patient and have real-time data to process.
It is interesting to notice that with the photoplethysmography (PPG) technique it is possible to measure both heart rate and SpO_2.

2.4.1 Photoplethysmography Theory

It is become usual to see in hospital patient with a fingerclip or an earlobe. These devices are based on photoplethysmography and are used in clinical environments to measure the heart rate and the blood oxygen level of a patient.
PPG derives from the classic plethysmography, a term that refers to the measure of changes in volume of a generic organ in the body.
This technique works by collecting, through a photodetector, the light emitted from some light emitted diodes (LEDs) that pass a portion of the body like the tip of the finger.
The volume change caused by the pressure pulsation is detected by illuminating the skin with the LED's light. Then, measuring the amount of light either transmitted or reflected, it is possible to calculate SpO_2 and HR. Usually, PPG technique is composed of two LEDs, one RED (635 nm to 700 nm) and one infrared (850 nm to 940 nm).
The blood pumped from the heart to the body, called oxygenated, bring Oxygen (O_2), through Haemoglobin (Hb), to the cells of the body and pick in return Carbon Dioxide (CO_2). The oxygenated blood, called also Oxyhaemoglobin (HbO_2), absorbs more Infrared (IR) light with respect to RED light. Similarly, the deoxygenated blood, called also Deoxyhaemoglobin (RHb), have opposite properties. It absorbs more RED light than IR light.
In Figure 2.4 C1 and C2, we can see the molar absorption coefficients of HbO_2 and RHb depending on light wavelength.

The collected light can be transmissive, when the LED and the photodiode are placed on the opposite side of the human body, or reflective, when LED and photodiode are in the same side.

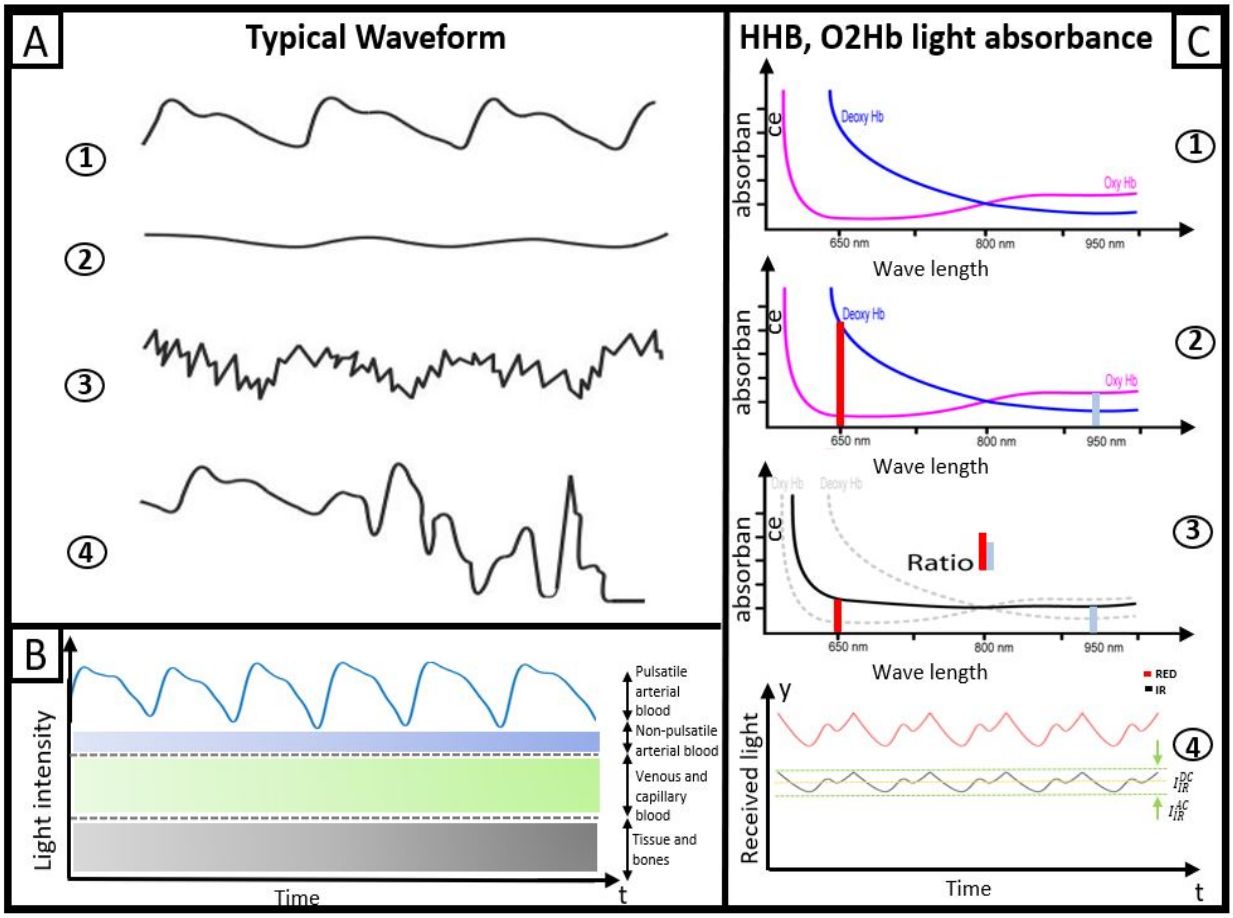

Figure 2.4: PPG signal. In A four type of PPG signals(1-Normal,2-Low perfusion,3-Noise artefacts,4-Motion artefacts). In B the component that compose the PPG signal are highlighted. In C the first three graph are related to RED and IR LEDs absorbency capabilities. In C 4, RED and IR channel typical signal and the DC and AC component of IR channel is illustrated.

The calculation of the oxygen saturation level starts with the application of the Beer-Lambert law. This law describes how light attenuation behaves depending on the properties of the material though which the light is travelling. The law is expressed in equation 2.1:

$$I = I_0 e^{-\varepsilon(\lambda)Cd} \quad or \quad A = ln(\frac{I_0}{I}) = \varepsilon(\lambda)Cd \tag{2.1}$$

I and I_0 represents respectively the received light intensity and the incident light (transmitted or received). C is the concentration of material and d is the optical path, while $\varepsilon(\lambda)$ is the molar extinction coefficient. Considering that the light travelling through the body will pass different material, the equation 2.1 can be written as follows:

$$A = d[\varepsilon_{HbO_2}(\lambda)C[HbO_2] + \varepsilon_{RHb}(\lambda)C[RHb] + \varepsilon_{others}(\lambda)C[others]] \quad (2.2)$$

With this equation it is possible to evaluate SpO_2 by using the molar extinction coefficients of HbO_2 and RHb.

First, we have to consider that the collected light will have two principal components, one is the pulsatile part, composed by arterial blood that changes periodically (cardiac cycle), the other is a constant non-pulsatile part, composed of skin, fat, venous blood, bones etc. This is clearly showed in Figure 2.4 B, with the pulsatile arterial blood on the top. Since the amount of oxygen has to be evaluated, a relevant quantity to define is the RED, IR modulation ratio R (Figure 2.4 C3):

$$R = \frac{A_{RED}}{A_{IR}} \rightarrow R = \frac{\partial(A_{RED})}{\partial t}[\frac{\partial(A_{IR})}{\partial t}]^{-1} = \frac{I_{RED}^{AC}}{I_{RED}^{DC}}[\frac{I_{IR}^{AC}}{I_{IR}^{DC}}]^{-1} \quad (2.3)$$

A_{RED} and A_{IR} represent respectively the absorbed RED and IR light, while AC and DC are the pulsatile and non-pulsatile components. The variables called I are the received light of the LEDs. To have a clear idea, in Figure 2.4 C 4, the variables I related with the IR LED is highlighted in the graph, while the red and black signals represents respectively the light detected from the RED LED and the light collected from the IR LED.

Once the calculation of R is computed as indicated in the last part of the equation 2.3, one more step is required to evaluate the SpO_2 value. Indeed, if we put together the second part of equation 2.1 with equation 2.3 and

rearrange the expression, SpO_2 calculation can be defined as:

$$S = aR^2 + bR + c \quad or \quad S = \frac{k_1 + k_2 R}{k_3 + k_4 R} \tag{2.4}$$

In equation 2.4, S represents the true value of SpO_2. In the first formula [8] a, b, c are calibration coefficients, usually provided by the company that fabricate the optical sensor. In the second formula [9], k_i represent some constants and, one or more constants are usually zero.
The pulse oximetry signal is strongly dependent on the portion of the body chosen, on the perfusion of the light and artefacts. Indeed, in Figure 2.4 A, four type of pulse oximetry signals are showed. The first one represents a normal PPG signal and the second one is a pulsatile signal during low perfusion, it shows a typical sine wave. The last two are signals affected by artefacts: the first one due to noise and the second one due to motions.
From a normal pulse oximetry signal, the estimation of the HR is simply a matter of computing the frequency component of a signal channel. Choosing the wavelength at which perform this calculation depends on the Signal-To-Noise Ratio (SNR) of the two channels. The SNR value is correlated with the quality of light source and the accuracy of the photodetector.

2.4.2 Pulse Oximetry: Anatomic body location

The body location in which the sensor will be placed is very important to obtain a reliable result. It affects considerably the quality of the pulse oximeter signal and the accuracy of the SpO_2 and HR estimation.
An interesting test, regarding the body location of the sensor, was carried out by Sally K. Longmore et al. [10]. They used the same sensor's model in eight different body locations to find the best anatomical position for photoplethysmography. The data was collected from a single microprocessor in two different conditions: at rest and while walking. The comparison of all the locations is then performed using commercial pulse oximeter

ground truth. The eight different body locations chosen for this test was: forehead, temple, rib cage, finger, wrist, tibia, lower back, back of neck.
Their results show that the most accurate body locations are the finger and the forehead with a percentage of median error, in resting situation, slightly above 2 %.
The experiment demonstrate that the best spots are the ones that are more well blood-perfused such as the fingers and the forehead.

2.5 Detection Algorithms

Different techniques are used to detect this kind of disease and more will come in the future thanks to new studies.
It is possible to detect OSA through different type vital signs measurements, using algorithms based on the analysis of sounds, pulse oximetry, respiration rate, ECG, etc. .
Many researches are focused on prototyping a device able to perform continuous measuring of a one or more vital sign; others have the scope to process data with multiple algorithms that have not been implemented on a developed system. Indeed, some algorithms have a high degree of complexity that makes its execution hard in system that have limited resources.
A brief overview of some algorithms for OSA detection, based on different vital signal measurements, will be carried out below. These algorithms differ not only for the kind of vital signals considered, but also on the sensor used to collect the samples.

- **Based on respiration**

  The data obtained from the oronasal airflow is typically used to detect a breathing disorder and here will be used for OSA detection. The data are processed using a low pass Butterworth filter, to remove artefacts that affect this type of measurement, and then the signal is normalized in order to avoid variations that depend on the subject.

The resulting signal is then segmented, and three time-domain features are extracted. These features were then used by a three binary Support Vector Machine (SVM) to classify the data. An SVM algorithm aims to find a hyperline in a N-dimensional space that distinctly classifies the data points. The variable N represents the number of features.

- **Based on pulse oximetry**

  Common oximetry analysis consists of cumulative time spent below a certain threshold (like 90%), Oxygen Desaturation Index (ODI), number of falls in SpO_2 below a defined value, etc.

  An example can be an algorithm based on threshold approach. Three point are taken in consideration and generated based on the oxygen levels waveform behaviour. The central tendency measure (CTM) approach has been then performed to improve OSA detection capability. The CTM is a statistic outline that represents the central point or the typical value of a dataset. These measures specify where most values in a distribution fall. In statistics typical measure of central tendency are for example mean, median and mode.

- **Based on sound**

  When we breath, we produce a characteristic sound that can be used to detect the existence of possible disorders. It is possible, for example, to detect breathing disorder by looking at the result obtained subtracting the average value of the audio signal to its energy envelope. A disorder period is detected if the value is negative. Then, six different features were calculated and used as input to a classifier. The produced output was classified by an adaptive threshold formed for each subject's score.

- **Based on combined approaches**

  Different vital signs can be used together for detection. For exam-

ple, heart rate and SpO_2, combined together, can be used for OSA detection. The frequency spectrum of the signals is first interpolated and then the data are averaged. In order to detect OSA, the algorithm looks for peaks on the apnea frequency band of both the signals. The apnea-related frequency band is obtained comparing different OSA positive subjects data together.

These are only some of the algorithms used for OSA detection; more of them are discussed in [11]. This aspect will not be discussed in deep because it is out of the scope of this dissertation. More details on detection algorithms can be founded in literature.

CHAPTER 3

STATE OF ART

There are in the market different types of wearable devices able to perform health monitoring of one or more vital signals. In this Chapter, a review will be presented of some WHDs available and the positive and negative aspects of each device will be discussed.
Furthermore, a review of some devices under research and development will be presented.

As explained in Chapter 2, choosing the body location of the WHD is a crucial element during the design procedure both for measurement quality and for patient comfort. The available WHDs on the market can be divided into different categories based on vital signal measuring, body location, system performance etc. Here these devices will be divided into four categories based on their body position:

- Ear sensor
- Wristwatches
- Finger clips
- Chest Strap and Adhesive Patches

3.1 Ear sensor

Ear devices are rapidly emerging in the market and are able to evaluate several physiological parameters like heart rate, oxygen saturation level and temperature[12].
Once connected to the ear the device is able to start measuring vital signal and due to the absence of muscle interfering, the presence of arteries close to the surface and its composition of mainly cartilage, it is considered a viable sensor. Indeed Valencell, a US-based biometric technology company that develops biometric sensors for wearable and hearable, argues that an ear signal is 100 times clearer than one at the wrist.
At the moment the number of this type of devices available in the market is low but they will represent a new trend in the coming years [13].

3.1.1 Cosinuss ° One

An example of an In-Ear WHD comes from COSINUSS [14]; it is the COSINUSS ONE (Figure 3.1). It is able to monitor the heart rate precisely and additionally it measures HRV and body temperature. The company is also promoting a future Blood Oxygen level calculation.
The ONE is composed of three main sensors: optical HR, temperature and 3-axis accelerometer. The device has a sample rate of 100 Hz and the communication is done with a Bluetooth Low Energy (BLE) version 4.0 to enable wireless communication with iOS such as Android, Apple or any device that supports the Bluetooth 4.0.
In the ear the blood vessels are continuously well supplied with blood and since the head is kept in rest from our body, the temperature is close to the core of the body.
It weighs 6.5 grams considering also the Lithium-ion Polymer (LiPo) battery and it has an overall size of about 4 x 4 centimetres. This device will last up to eight hours and can be recharged in an hour.

Figure 3.1: COSINUSS -Cosinuss One, Wireless In-ear

Furthermore, its current price is 129 €, making the ONE a cheap and accurate solution.

3.2 Wristwatches

Wristwatches, known also as smartwatches, have been developed for a few years and one of the first devices of this type was AMON, presented in 2002. It is capable of monitoring skin temperature, HR, blood oxygen saturation with a wireless data communication module [15]. More recently, a new generation of smartwatches is emerging with wireless (and/or mobile communication) able to provide more than 24 h of vital signal monitoring [16]. Smartwatches have a comfortable design, like a normal watch, and for this reason these devices are being developed in different areas such as activity and fitness trackers (like burned calories and distance travelled), HR, and recently sleep monitoring, like PEAKTM which was the first wristwatch able to track sleeping cycles [17].

In the market it is possible to find a big variety of wristwatches but here only two examples will be illustrated.

3.2.1 Caretaker Medical

Caretaker4 (Figure 3.2) comes from the Caretaker Medical company (USA) and it is an innovative patient monitoring that uses only a simple finger cuff to measure continuous beat-by-beat blood pressure, HR, blood oxygen saturation (SpO_2), respiration rate and core body temperature [18]. The company is also waiting an approval from the FDA for early warning of score and blood volume levels.
This device uses a Bluetooth Low Energy (BLE) in order to communicate and thanks to the capability of 2000 mAh from a rechargeable Lithium Polymer battery it has a lifetime of eight hours while the recharging takes five or more hours depending on the charging method. The system size is 58x80x28 mm and it weigh 12 grams.

Figure 3.2: Caretaker4 and the related App

3.2.2 WristOx2

WristOx2 model 3150 (Figure 3.3) is a WHD coming from Nonin Medical incorporation based on the proven PureSAT SpO_2 technology made by Nonin Medical. This device is used for accurate oxygen saturation and pulse rate monitoring and data recording, making it good for studies at home, in the hospital and in sleep laboratories.
This device is able to perform discrete and continuous measurements with the possibility of being turned off, in order to save battery life, the display and the Bluetooth for the communication. It works with two AAA batteries (rechargeable or not) and considering a continuous measuring mode, display and Bluetooth turned on, the battery life is 44 hours. The device size is small $51x73x19$ mm and it weigh 70 grams; both these factors increase the overall comfortability [19].
The WristOx2 is available with a relatively high price of 798$ compared to other similar devices on the market.

Figure 3.3: WristOx2 model 3150

3.3 Finger clips

Finger clips are another type of WHDs that are becoming popular in recent years. They are usually used to measure HR (and/or heart rate variability) and the oxygen level in the blood. These devices have been used in hospitals for many years and the first test on a patient was reported in 1975 [20] and thanks to the progress in technology they are available in the market with a relatively low price for home monitoring. High-resolution pulse oximetry and heart rate has been developed for in-home sleep apnea screening and testing in patients for whom it is impractical to perform polysomnography [21].

3.3.1 CorSense® by Elite HRV

The CorSense (Figure 3.4) heart rate variability monitor is an accurate device comparable to a grade 5-lead EKG/ECG for HRV (it represents a holistic approach to stress on the body).

Figure 3.4: CorSense® by Elite HRV

The CorSense starts measuring as a finger is inserted and detected in the device and uses a Bluetooth Low Energy (BLE) in order to communicate the data to a smartphone. The optical sensor uses 3 multi-wavelength LED and 5 large visible spectrum photo detectors and 1 infrared detector enhancing accuracy for different skin colours and circulation strengths. The system is based on an ARM processor with a sample rate of 500 Hz. It has a lithium Ion cell battery that provides more than 4 hours of continuous operating time while it can be charged through a micro USB port. Even if this device is really accurate its cost is low (149$) [22].

3.3.2 Motiv Ring

This device (Figure 3.5) comes from MOTIV and it is not a proper finger clip since it is a ring shape heart rate monitoring system. It is able to measure heart rate, track steps and distance and has the possibility of tracking sleeping activities.
Inside the ring there are a flexible circuit with the photodetector and a flexible battery. Once inserted in the finger, the device sends data to the smartphone through a Bluetooth module and the vital signs are displayed. It has a magnetic USB charger and its battery life is up to 3 days with a single charge. It is made of lightweight titanium that reduces the overall weight and improves its strength. It is available in the market at a cost of 199$ [23].

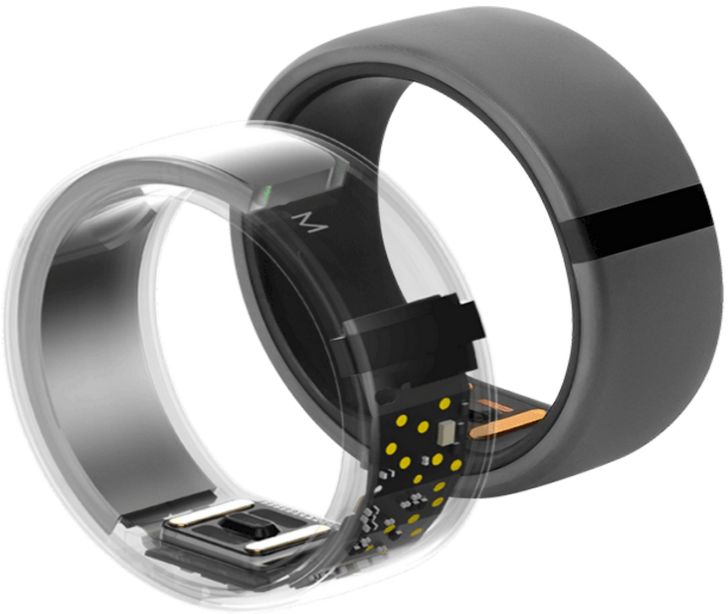

Figure 3.5: Motiv ring by MOTIV

3.4 Chest Strap and Adhesive Patches

Chest straps and adhesive patches are devices capable of measuring different vital signs such as ECG, heart rate, heart rate variability, oxygen saturation, respiration rate, body posture and movement, blood pressure and body temperature. There are many devices on the market able to measure one or more of these vital signs.

3.4.1 VitalPatch

The first example of a chest strap is the adhesive patch coming from VitalConnect called VitalPatch (Figure 3.6). It is a powerful device since it can measure 8 different vital signs including live ECG at a sample rate of 125 Hz and it transmits data with a Bluetooth low energy (BT4.1) with a tablet sold by VitalConnect [24].
VitalPatch has ECG electrodes to detect heart rate and a 3-axis MEMS accelerometer to detect posture and a fall detection system for ambulatory patients. It can also measure skin, body and ambient temperature.Thank to

its disposable zinc air battery the device can operates for about 120 hours.

VitalPatch weights 13 grams and its dimensions are small indeed it is a

Figure 3.6: VitalPatch attached on the chest

120x40x9.5 mm device while its application on the chest it's based on a hydrocolloid adhesive [25].

3.4.2 QardioCore

QardioCore (Figure 3.7) from QARDIO [26] is able to perform continuous single lead EKG signal measurements and monitor heart rate (and heart rate variability), respiratory rate, skin temperature and activity.

The device is equipped with Bluetooth Low Energy (BLE) version 4.0 to enable wireless communication with smartphones (iOS but not Android) and an app is available to display EKG waveforms and interact with the device.

For what regards the hardware, QardioCore performs 16-bit acquisitions at a sample rate of 600 Hz and the Lithium-Ion Polymer (LiPo) battery can power the device for more than one day without recharging.

Figure 3.7: QardioCore and its own application on mobile phone

QardioCore weighs 130 g including the battery and it is dust and water jets resistant. Due to its positioning on the body, QardioCore can't monitor arterial blood oxygen level. Its current price is 499€ which is quite higher than for similar devices in the market.

In this chapter different devices were illustrated in order to have an idea of the current state of the art. The number of devices available on the market is much greater that what was presented here.
For simplicity some of them are listed in the table in Figure 3.8 while a deep discussion on each of them will not be performed.
The devices presented in this table are not all medical, some of them are used for fitness porpoise. The accuracy of the heart activity is based on the capability to acquire a QRS complex that is the combination of 3 of the graphical deflections seen on a typical ECG waveform [27]: HR devices can estimate the heart rate based on R-peaks, but they are not able to acquire all the R-peaks of an ECG signal. R-R interval devices can determine

the timing of each R-peaks but unable to obtain the ECG waveform. Finally, ECG devices can be used to extract from ECG waveform peaks and valleys and perform different medical analysis.

More devices are available in the market, but they are not discussed here since they are outside the scope of this dissertation.

		HR	R-R Interval	ECG
T-Shirt	BioMan	X		
	HexoSkin		X	
	Smartex WWS			X
	D-Shirt	X		
	Nuubo -nECG TEXTILE			X
Chest Strap	HRM-Tri	X		
	BioHarness 3		X	
	Zephyr HxM		X	
Adhesive Patches	ePatch			X
	ZIO XT PATCH			X
	CaedioLeaf FIT		X	
	CardioLeaf ULTA			X
	HealPatch MD			X

Figure 3.8: Heart Activity trackers divided by type of WHD on the body. The characteristics chosen for this separation were based on [17-31] with an increase of heart activity accuracy (HR/R-R intervals/ECG)

3.5 Device under development

This section is devoted to the description of an innovative sensor and a device associated with it. This is not the unique device, more can be found in scientific papers but will not be treated here.

The sensor development made by Duun S. et al. starts in 2007[43]. The aim of this work was to create and use a ring-shaped backside photodiode to use in a wearable reflective pulse oximeter. The use of a ring-shaped photodiode increases the amount of light that can be collected. The sensor is supposed to be used in low-power homecare applications.
The chip has a hole in the middle in order to fit the dual LEDs (660 nm and 940 nm) while the photodiode, concentrically around the hole, has an inner radius of 3.29 mm and an outer one of 4.07 mm that results in a 18 mm^2 active area. The diode is fabricated with a p-type silicon (higher diffusivity compared to n-type).
The chip front side has an anti-reflection filter that permits a 98% transmission efficiency for both the LEDs. The photodiode, as shown in the paper, has a characteristic close to an ideal diode while the quantum efficiency is around 70%. This value it is assumed to be reasonable for the authors. The fabrication method is carried out in detail in [43] but, will not be treated here.
Once the sensor was tested, Duun S. et al. [44] decided to make a design of an electronic patch based on the ring-shaped photodiode. The power of this sensor consists on collecting backscattered light all around the light source, increasing the collected light more that normally single square photodiode positioned on one side with respect to the light source. As pointed out in the article: “This enables very low LED driving current which will lower the power consumption of the device”.
The electronic patch is composed of sensor, electronic, radio communica-

tion and battery, all integrated into an adhesive patch made of hydrocolloid polymer.

This is only the fist prototype and the results presented by the authors shown that the device can be used for pulse oximetry measurements.

The next prototype patch designed and developed by Duun S. et al. [45] has a size of 80 mm by 60 mm and a thickness of 5 mm. The electronic patch is composed of a disposable part and a reusable part: the first one is composed of a hydrocolloid polymer and contains the battery while the second one contains the sensor (ring-shaped pulse oximeter) and the microelectronics encapsulated in a hard polylaurinlactam.

The electronic patch is composed of a central PCB housing the ring-shaped photodiode (to have low-power consumption) with the two LEDs. The aim of this work is the development of the firmware and hardware of the electronic patch while has limited focus on the power consumption.

The main components of the system are: the CC2430 (Texas Instruments) System-on-Chip (SoC) with a build-in 12-bit analog-to-digital converter, MAX6947 (Maxim Integrated Products) current controller, a 32 MHz crystal oscillator and a 64 *Kbit* electrically erasable programmable read-only memory. The communication used are two: I2C for the memory and a RS232 serial interface for sending the data on a host pc.

The firmware has two main tasks to accomplish: 1) read the data from the sensor and save them and 2) send the data to a host pc. The first task needs an accurate timing while the second one can be completed when the serial is available.

The authors performed also a current consumption of the device, obtaining values from 0.2 mA (SoC idle) to 18.2 mA (SoC active, LEDs @ 10 mA and 200 Hz while sending data on serial wire).

The obtained results are satisfactory for a pulse oximeter wearable healthcare device while an improvement on the power consumption can be implemented and in particular, in the article, the authors say that the power

consumption could be lowered using a newer generation of microelectronic systems and optimizing the firmware and the analog-to-digital conversion.

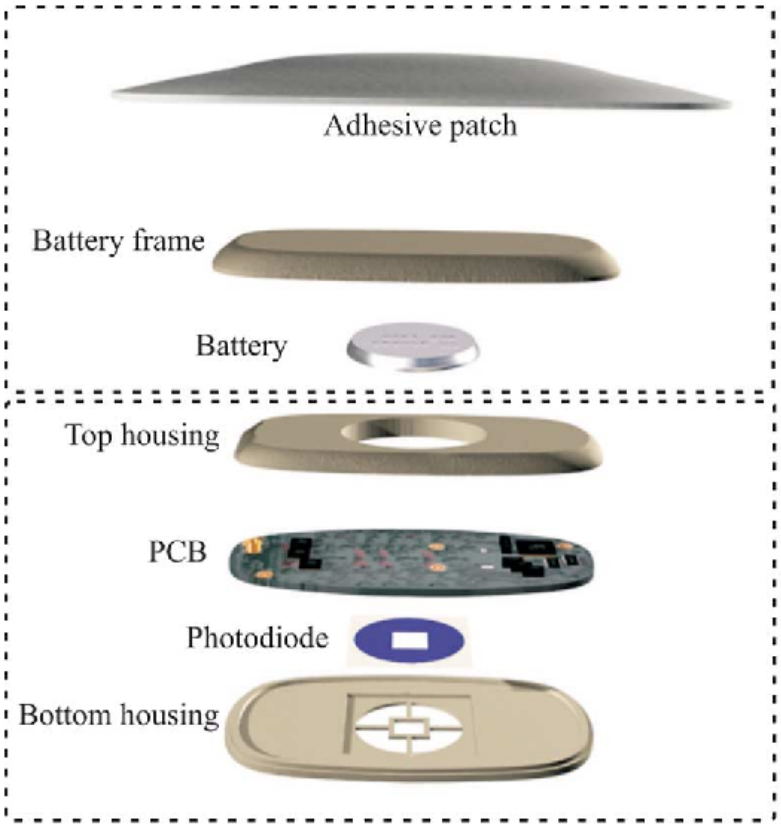

Figure 3.9: mechanical assembly of the Electronic patch [35]. Upper, disposable, contains: adhesive patch, the battery and the battery frame. Lower, reusable, contains: electronics, reflectance pulse oximetry sensor and a hard-plastic housing.

In figure 3.9 it is possible to see how the device is composed. All the parts can be divided in two groups: the top side that represents the disposable part of the device and the bottom side that is the reusable part of the all device.

The photodiode and the PCB are contained inside the top and bottom housing. On the top, an adhesive patch is used to apply the device on the chosen body location and the battery will be attached to the reusable part through a battery frame.

All the components together give as a result a compact and reusable device

able to accomplish interesting tasks, but as the author highlighted further improvements and tests on power consumptions are necessary to increase battery life. For more detailed information, the reader is invited to go to [33][34][35].

As we saw, the device under development is small, comfortable and gives good results. The disadvantage of such a device, with respect the market, is that it still requires improvements and deeper analysis.
Furthermore, the advantages are the new photodiode (it is more sensitive with respect other sensors due to its geometric shape) and the small dimensions.

CHAPTER 4

MEASURING DEVICE

This Chapter will provide detailed information of the device from the hardware point of view.
At first, requirements and specifications of the system will be described and then, the hardware design and the methods used will be highlighted and explained in detail.
Finally, the hardware design of the whole system will be illustrated.

4.1 Device Requirements

The measuring device should be compliant with different requirements, typical of wearable health devices (WHD) also called Personal Health System (PHS), when it regards home monitoring. Some of the requirements are reported below while an in-depth analysis is carried out in [46]:

1 Low-power capability

2 Continuous monitoring

3 Comfortability with the daily routine of the patient/user

4 Reliability

5 Low-cost, cost-effective

6 No effort for the availability of the information

To these requirements some others have been added to guide the design and development of the measuring device:

a Body temperature

b Oxygen saturation (SpO_2 meter)

c Heart rate meter

d On-board storage of data

e Measuring period of 48-72 hours

f Rechargeable battery

g Compact form factor

Next it is important to make some necessary remarks. Requirements **1**, **2**, **a**, **b**, **c**, **e** cannot be satisfied at the same time with the available technology. For this reason, requirement **2** will be neglected and replaced: discrete sampling with a certain frequency in order to respect requirement **1**, **e** and use the sensors mentioned before.
Moreover, requirement **f**, **g** is related to **3** because the battery represents one of the biggest components of the system. The most convenient choice is then a coin cell battery. In the market a great variety is available but, considering also the necessary capability (in mAh), the battery dimension is crucial and for this reason **f** will be replaced with a non-rechargeable battery in order to respect **g**, **3** and make possible the required measuring period (**e**).
For these reasons, the proposed solution is different and becomes, for the first design:

i Body temperature every 3 minutes

While for the second design:

ii Body temperature every 3 minutes

iii Oxygen saturation (SpO_2 meter) and Heart rate every 40 minutes

These periods have been chosen based on calculations regarding: the power consumption of the sensors, the memory, the step-down DC-DC converters and the battery capability.
In the next section the reasons of the chosen periods will be explained.

4.2 Device Specifications

In this section a summary of hardware and software of the measuring device is provided while a detailed discussion is presented respectively in section 4.3 and chapter 5.
A theoretical calculation of power consumption has been performed considering the sensors active current and quiescent current.
Powering the Microcontroller ST Microelectronics STM32L072KB, the temperature sensor Texas Instruments (TI) TMP117, the photoplethysmogram (PPG) sensor Maxim Integrated MAX30102, the step-down DC-DC MAX1921 and the EEPROM CAT24C128, the current draw is less than 6 mA with a power supply voltage for most of the components of 1.8 V. In shutdown mode the consumption is less than 1 mA with a quiescent current for MAX1921 of about 220 µA. The component that consume more is the PPG sensor MAX30102 that in continuous measuring mode can consume 18 mA each minute: considering 30 seconds of measuring period and 30 seconds of shut-down current. The reason of this high consumption are the red and infrared LEDs. The battery is the Panasonic CR2354 with a nominal voltage of about 3 V and 560 mA capability. Using a measuring period of one value of SpO_2 and HR every 40 minutes and one value of Temperature every 3 minutes the device can achieve a measuring period of 48-72 hours (requirements **e**).
The reason of this choice is a comparison of different possible periods

based on the information coming from the datasheets [47] [48] and tacking into account the battery capability. A clear summary of technical specification is provided in Table 4.1

Table 4.1: Technical specification of measuring device.

Item	**Value**
Case size	32 mm x 42 mm x 8 mm
PCB size	31.5 mm x 41.15 mm x 6 mm
Operating Voltage	$1.8V$
LED Operating Voltage	$3V$ to $5V$
Battery Voltage Range	$3V$ to $3.7V$
Battery Capability	$560mAh$
Working period	$48-72$ hours
MAX30102 acquisition time	10 s
MAX30102 Sampling Frequency	50 Sa s^{-1}
MAX30102 Sample Size	18-bit, 2-channels
MAX30102 ADC Integration time	215 μs
HR sample size	32-bit
SpO_2 sample size	32-bit
TMP117 Sampling frequency	8 Sa s^{-1}
TMP117 Sample Size	16-bit

4.3 Hardware

In this section detailed information about hardware design and Printed Circuit Board (PCB) layout are provided.
In the fast prototyping design, a modular approach has been used and each part is presented and explained separately.

4.3.1 Circuit Design

- **Microcontroller**

  The Microcontroller (MCU) represents a curtail component for every system for different aspects. The measuring device needs many peripherals, for some of which Digital Signal Processing (DSP) is necessary and for these reasons a 32-bit MCU is mandatory. In order to satisfy requirement 1 a high power MCU must be discarded also because it is important to keep under control requirement **5**.
  For these reasons an ultra-low-power MCU must be chosen. Starting from the 2017 Embedded Markets Study [49] where different companies have been compared, the ST Microelectronics (STM) results as one of the most used company for microcontrollers. The STM32L072KB coming from the STM32L0 ultra-low-power STM family has been chosen for the measuring device and a block diagram is shown in Figure 4.1.
  The STM32L072KB has different characteristics, some of them are listed below [50]:

  - Arm® 32-bit Cortex®-M0+ with MPU
  - From 32 kHz up to 32 MHz max
  - 32 kHz oscillator for RTC with calibration

- Internal low-power 37 kHz RC
- Internal multispeed low-power 65 kHz to 4.2 MHz RC
- Up to 192 KB Flash memory with ECC (2 banks with read-while-write capability)
- 20 KB RAM

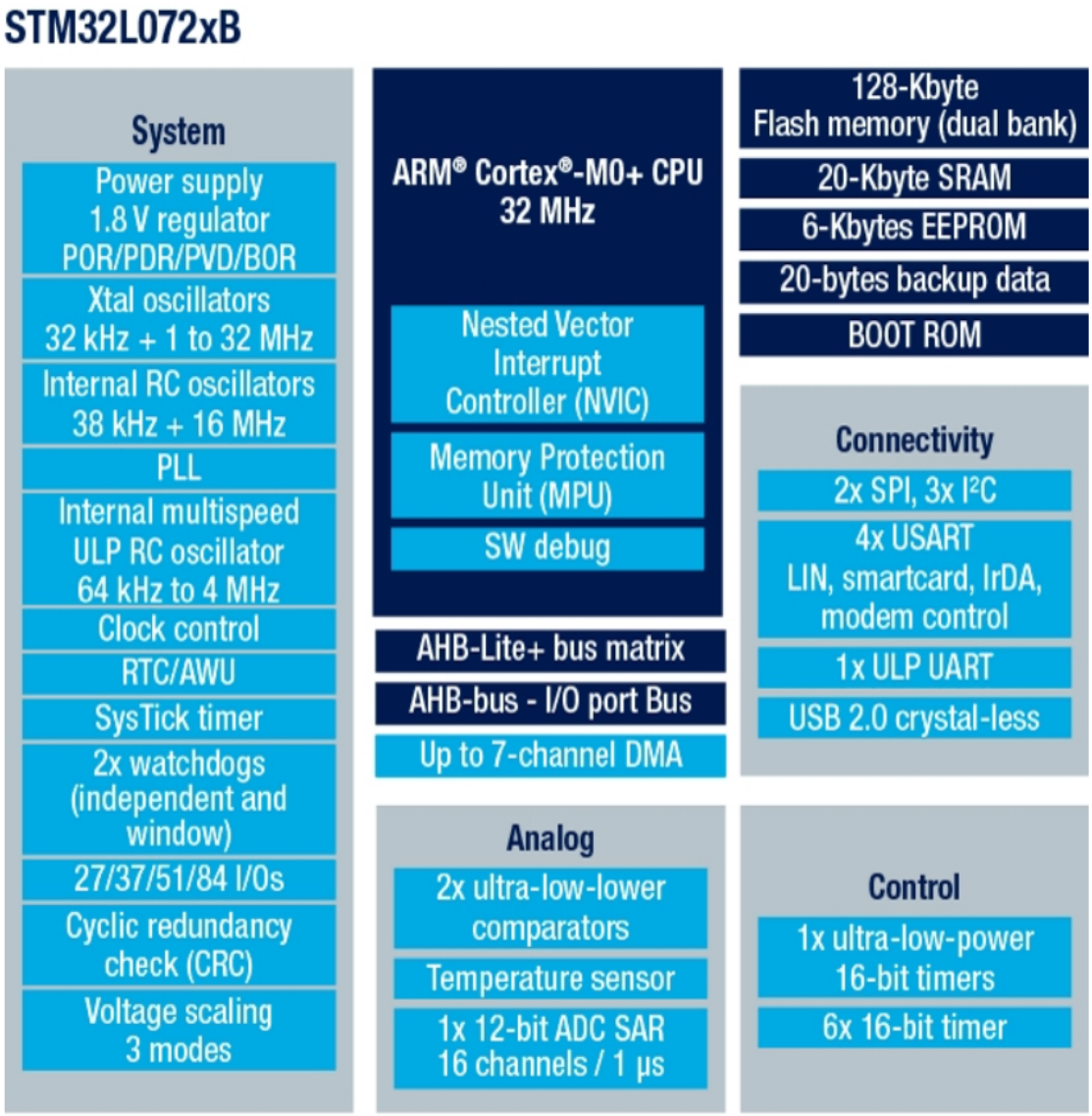

Figure 4.1: STM32L072xB block diagram (Copyright © 2017, ST Microelectronics)

- 6 KB of data EEPROM with ECC
- Sector protection against R/W operation

- 32 General Purpose Input Output (GPIO)
- 1.65 V to 3.6 V power supply
- 0.29 µA Standby mode (3 wakeup pins)
- Down to 93 µA/MHz in Run mode
- 41 µA 12-bit ADC conversion at 10 ksps
- Pre-programmed bootloader – USB, USART supported
- Development support – Serial wire debug supported
- 12-bit ADC 1.14 Msps up to 16 channels (down to 1.65 V)
- 2x ultra-low-power comparators (window mode and wake up capability, down to 1.65 V)
- 2x Inter Integrated Circuit (I2C) driver with System Management Bus (SMBus)
- 1x Serial Peripheral Interface (SPI) driver
- 4x Universal Synchronous/Asynchronous Receiver/Transmitter (USART) and one low-power UART

STM32L072KB requires only some decoupling capacitors for the power supply. No external crystal oscillator has been used and as a clock source will be managed the internal Multi-speed internal RC oscillator (MSI), trimmable by software, able to generate 7 frequencies (65 kHz, 131 kHz, 262 kHz, 524 kHz, 1.05 MHz, 2.1 MHz, 4.2 MHz).
The MCU will be used in low-power run mode and, in this mode, the MSI is not turned off, providing a constant clock to the system.
A PCB view of the MCU is shown in Figure 4.2 and since a modular approach has been used for fast prototyping, it is a stand-alone view of only the MCU. The complete schematic with interconnections will be illustrated in section 4.5.

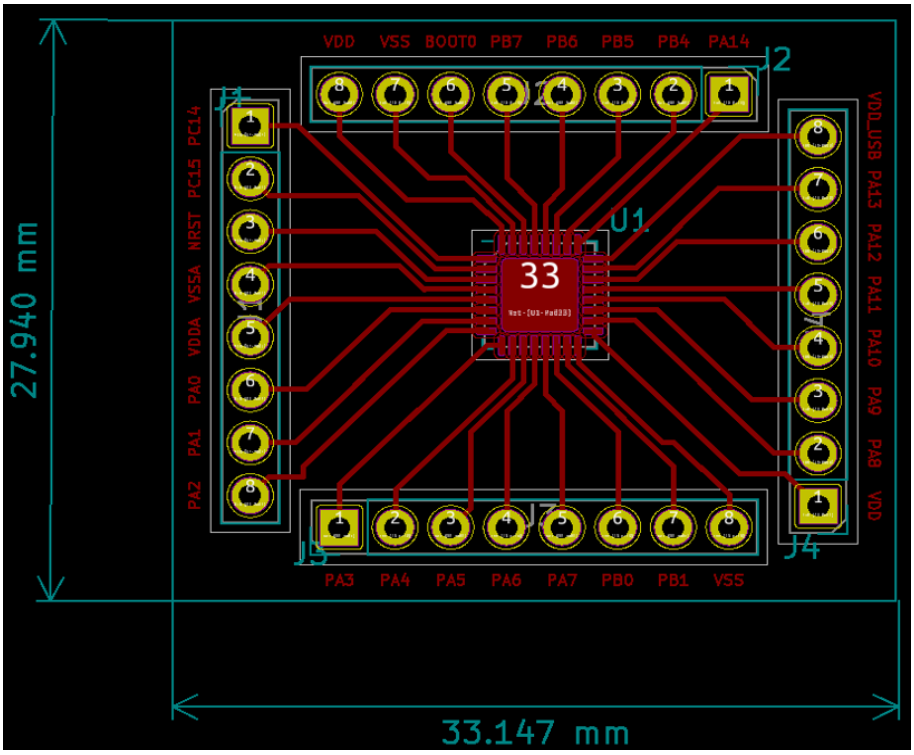

Figure 4.2: Microcontroller PCB view

- **Step down DC-DC** In order to decrease the operating voltage from 3 V (battery) to 1.8 V, the step-down DC-DC MAX1921 has been chosen. The 1.8 V is used to supply all the components except for the MAX30102 that requires two supply voltages: 1.8 V for the internal logic of the component itself and a minimum of 3 V for the two LEDs. In the modular approach used for fast prototyping, the PCB housing the DC-DC contains also the pull-up resistors used for the I2C communication for both the Maxim Integrated MAX30102 and the TI TMP117. The calculation for the capacitors, resistors and Inductor are carried in the MAX1921 datasheet [51] and are explained below. At first, a calculation of a duty-cycle is required as expressed in formula 4.1:

$$DutyCycle_{MAX} = \frac{V_{OUT}}{V_{IN_{MIN}}} * 100 \tag{4.1}$$

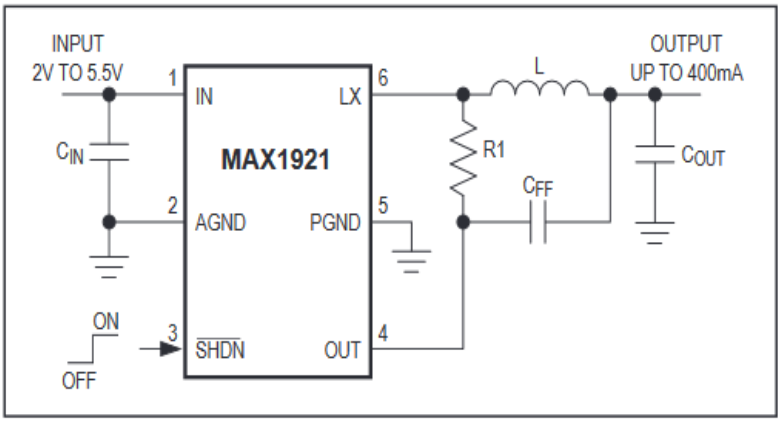

Figure 4.3: MAX1921 typical application (Copyright © 2016, Maxim Integrated)

And then, if $DutyCycle_{MAX} < 50\%$, $V_{CRITICAL}$ is defined as:

$$V_{CRITICAL} = V_{IN_{MIN}} - V_{OUT} \tag{4.2}$$

else

$$V_{CRITICAL} = V_{OUT} \tag{4.3}$$

Now, the calculation of a minimum Inductance L_{MIN} is performed with the following formula:

$$L_{MIN} = 2.5e^{-6} * V_{CRITICAL} \tag{4.4}$$

At this point it is possible to calculate C_{OUT},C_{FF} and R_1 as follow:

$$C_{OUT} = 2.5 * 10^{-6} * V_{CRITICAL} \tag{4.5}$$

$$C_{FF} = \frac{2.5 * 10^{-6}}{R_1} \quad with \quad R_1 = 5 * 10^{-4} * R_{L_{MAX}} \tag{4.6}$$

For simplicity, the obtained values are summarized and listed in Table 4.2 while the DC-DC schema is shown in Figure 4.3. In the datasheet the minimum value of C_{IN} is suggested to be $2.2\,\mu F$.

These components, the pull-up resistors for the I2C communication and the pull-up resistors for the interrupts pin for both the Maxim In-

Table 4.2: MAX1921 Chosen Values

Component	Value
L	$4.7\mu H$
C_{IN}	$4.7\mu F$
C_{OUT}	$4.7\mu F$
C_{FF}	$5600\,pF$
R_1	$4.75K\Omega$

tegrated MAX30102 and the TI TMP117 are added to the PCB housing the step-down DC-DC.

Besides these components two connectors are added to make possible the communication and energy flow to and from the step-down DC-DC.

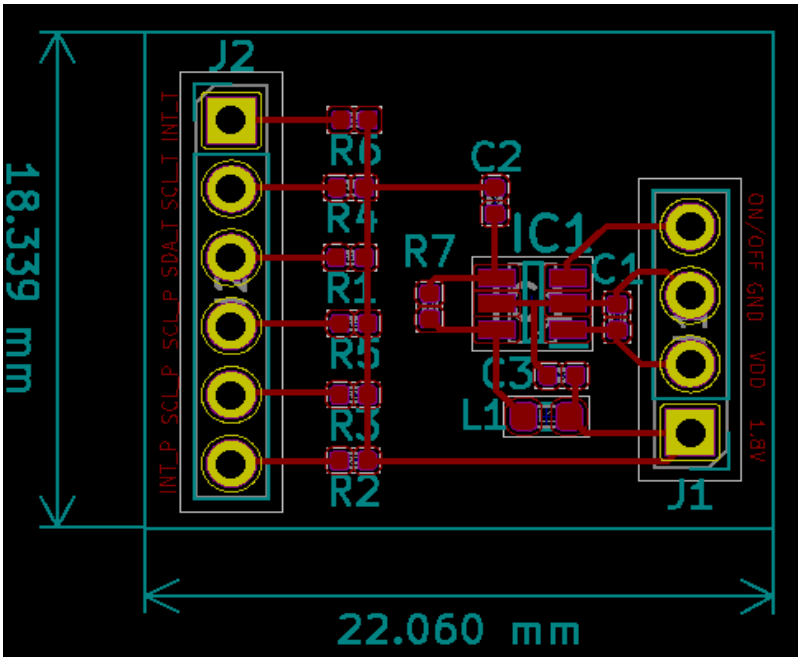

Figure 4.4: Step down DC-DC PCB view

The PCB view is then shown if Figure 4.4 and the relative dimensions are highlighted.

- **Sensors**

  For the two sensors, an ad hoc PCB has been designed and except for the pinheads it contains only bypass capacitors: two are used for

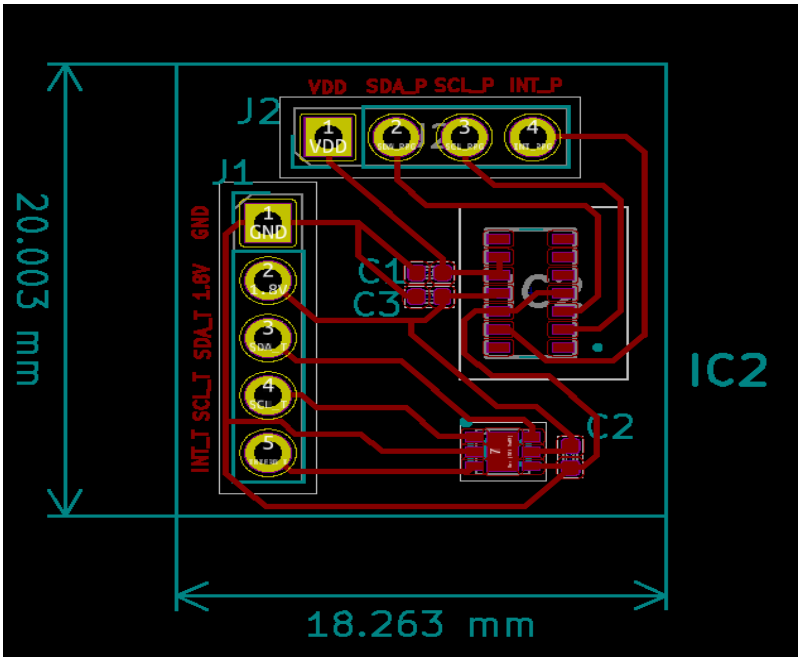

Figure 4.5: MAX30102 and TMP117 PCB view

the MAX30102 with a value of 1µF and one for the TMP117 with a value of 0.1 µF. These values are taken from the datasheet of the components [47][48] that suggest a minimum value of bypass capacitance.

In Figure 4.5 the PCB view of the sensor is shown and the dimensions are highlighted.

- **Memory**

  For what regards the memory unit, a 128 kb Serial CMOS EEPROM, the CAT24C128, has been chosen. It is internally organized as 16,384 words of 8 bits each and powered with 1.8 V through the step-down DC-DC converter. The communication protocol used is the I2C and the power consumption for a write operation is 3 mA, with a clock frequency of 400 kHz and a voltage supply of 5.5 V [52].
  In order to reduce as much as possible the power consumption, the component will be turned off for most of the time and used to store data only when 32 values of temperature are collected or when 64 value of HR and SpO_2 respectively are collected. In this way the

amount of write operation is reduced and the power consumption decrease.

For the temperature data, a write operation will occur every 1 hours and 10 minutes while for the PPG data, a write operation will occur every 10 hours and half. In this way, approximately 35 write operation will be executed with a total power consumption of about 90 mA for 48 hours of measuring period. This value is an overestimation because the clock frequency for the I2C communication will not be 400 kHz but 131 kHz and the supply voltage will be 1.8 V, less then 5.5 V.

For this component will not be illustrated the PCB view.

4.4 PCB Manufacturing

Starting from the design presented in the previews section two methods were considered to make a fast prototyping of the device. These methods have been implemented using the instruments present in the laboratory without the necessity of a supporting external company.
In order to have an easier debugging of the hardware, a modular approach was used, and more than one PCB have been developed. In this section the two methods will be explained in detail and compared.

4.4.1 First Method

The aim of the first method, as also the second one, is to obtain a one-layer PCB. The materials used for such a goal are the following:

- Copper clad Laminate (CCL): 75 x 125 mm
- Photo-sensitive ink: POSITIV 20 from KONTACT CHEMIE
- Transparent sheet

- Printer: Xerox ColorQube 8580
- CNC machine: Wegstr 3-axis
- UV oven: Sterilizer ultraviolet model VS-208
- Ferro chloride
- Acetone

At first, the copper clad board (CCL, Figure 4.6A) is covered with a thin layer of non-corrosive photo-sensitive ink (Figure 4.6B) in such a way that, if exposed to a UV light, the ink dissolves. The ink comes in a spray can and makes the covering procedure of the copper clad board easier with respect other types of products. For what regards the colour, the POSITIV 20 is transparent.
Using the designs developed and described in the previews section, the Gerber file, containing the footprints of the components and the tracks, it is printed on a transparent sheet using the Xerox ColorQube 8580. After that, the transparent sheet is placed on the top of the copper clad board with photosensitive ink with a bit of tape, in such a way that the transparent sheet does not move and then it is exposed to an UV light through a UV oven for about 30 minutes, or more, till all the undesired ink is removed. At this point we obtained a copper clad board where all the copper is exposed except the tracks and footprints that are still covered with the photosensitive ink (Figure 4.6C).
The Gerber file obtained from the design is translated in G-code through an online software called Carbide3D (http://carbide3D.com). The translation is done in order to load this information to the Wegstr software of the CNC machine. From the obtained G-code file, two information are extrapolated and two different files are created: drilling file and routing file. The drilling and routing files obtained are loaded on the Wegstr's software of CNC machine to make the drills and cut the desired shape of the copper

clad board respectively.
At first the drilling file is loaded and then a manual calibration is executed to the CNC machine: the drill is left free and the z-axis is set to 0, tightening the drill, as soon as the drill touches the board.
Moreover, the x and y axis are calibrated manually in order to fit the board shape and make the holes in the desired positions.
At the end, the routing file is loaded in the Wegstr's software and after that the drill bit is changed, a manual z-axis calibration is done again. Since the board is not removed or moved, is not necessary to make a x-y axis recalibration.
At the end of this step the result is a cutted and drilled board (Figure 4.6C).
The obtained board is then introduced into a vessel containing ferric chloride for about one hour to etch the area where no track or copper is required.
At the end of this process all the unnecessary copper is removed while the desired tracks and footprints are not etched since the ink protected them.
Finally, acetone is used to clean the board from the remaining ink and the result is the PCB with all the necessary tracks and footprints (Figure 4.6D) while the remaining work consist on soldering the components.

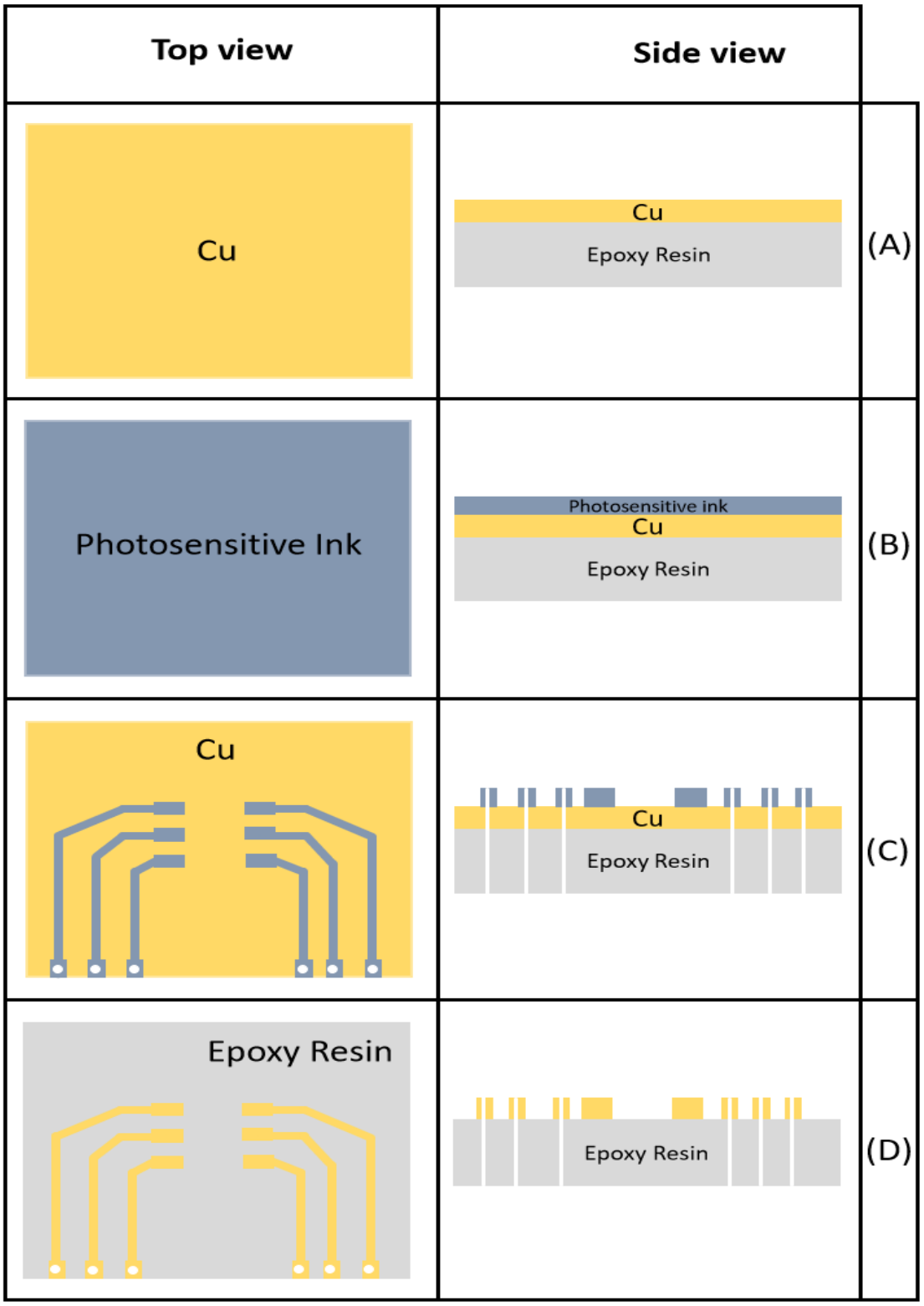

Figure 4.6: Top and side view of the fabrication methods. (A)Copper Clad Laminate (CCL). (B) CCL with photosensitive ink. (C)CCL with unnecessary ink removed and drills. (D) Final board with tracks and footprints.

4.4.2 Second Method

The objective of the second one is the same as for the first method, but the material and instruments used are different. Indeed, they are listed below:

- Copper Clad Laminate (CCL): 75 x 125 mm
- Photosensitive ink: POSITIV 20 from KONTACT CHEMIE
- Laser machine: JPT MOPA Fiber-Laser, Power = 20W, λ = 1064 nm
- CNC machine: Wegstr 3-axis
- Hydrogen peroxide and Muriatic acid
- Acetone

At first the copper clad board (CCL, Figure 4.6A) is covered with a thin layer of non-corrosive photosensitive ink as in the first method (Figure 4.6B). The ink is the same and a spray can procedure as described before is performed. Ones the ink is dry, the copper clad board is put under the laser machine.
Here, as before, the Carbide3D online software is used to convert the Gerber file into G-code. The G-code generated is used both for the Ezcad laser software and the Wegstr's software of the CNC machine. The complete generated G-code is loaded in the Ezcad laser software. From the Ezcad software a great number of parameters can be set. For my objective current pen is set to 0 and only few parameters are adjusted, in particular:

- Loop count is set to 1
- Speed is set to 1000 mm/s
- Power is set to 100 %

- Frequency is set to 50 kHz

At this point, with this parameters set, a Hatch procedure is executed between the tracks/footprints and the background in order to obtain a negative of the Gerber file in such a way to expose only the undesired part to the laser and therefore remove the ink everywhere except from tracks and footprints.

As in the first method in the Wegstr's software the drill and routing files are loaded one at a time to drill and cut the board as desired. As described in the previous method, a manual calibration is performed: first for the drilling procedure and then for the routing one.

The obtained cutted board (Figure 4.7A and Figure 4.6C) is then submerged into a vessel containing Hydrogen peroxide ($H_2\ O_2$) and Muriatic acid (HCl) with a 2:1 ratio respectively for about 3-4 minutes.

Finally, once all the undesired copper is removed the board is washed with water and then cleaned with acetone in order to remove the remaining ink from the tracks and footprints (Figure 4.7B and 4.6D).

As before, the remaining step is soldering the components.

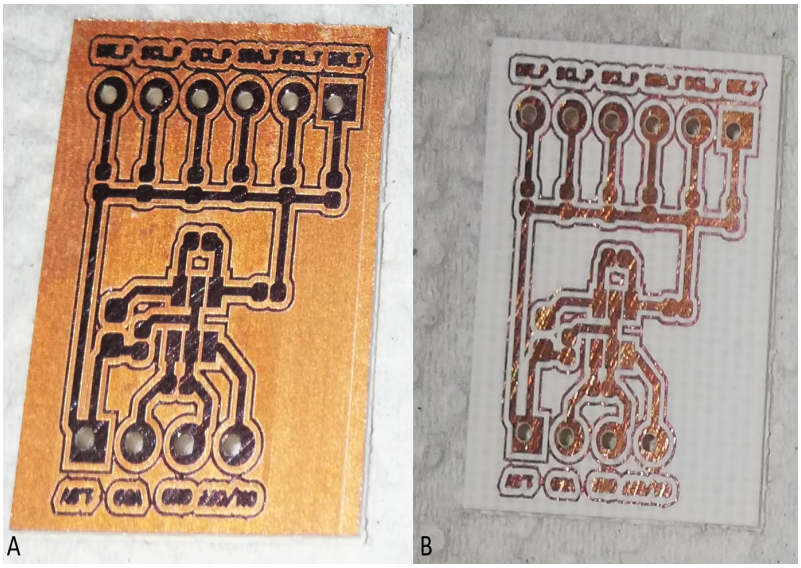

Figure 4.7: Board after laser and CNC procedure (A); Board after etching and cleaning procedure (B).

Before comparing the two methods, a top and side view of the fabrica-

tion methods is shown in Figure 4.6.
This picture illustrates the result after the main steps. In particular, the starting point (Figure 4.6A) represents the Copper Clad Laminate (CCL) composed with a thin laminate of copper (0.5 mm) and a rigid substrate made of epoxy resin with a total thickness of 1.6 mm typically. The next step (Figure 4.6B) illustrate the outcome after the application of a non-corrosive photosensitive ink. The thickness was not measured but it is supposed less than 1 mm. In Figure 4.6C is illustrated the board after the UV oven for the first method or after the laser for the second one. Besides that, it represents also the result after the procedure with the CNC machine: holes and cutting procedure.
Finally, the last step (Figure 4.6D) illustrates the result after the etching and cleaning process: the tracks and footprint remain while the unnecessary copper is removed.

4.4.3 Results and Comparison

As for the PCB results, the two methods are similar to each other but, in my experience, I notice that the second method gives more precise PCBs due probably to the accuracy of the laser. Indeed, the possibility to set a great number of parameters gives more degree of freedoms.
An important role in this comparison is the time: the process of the first method require approximatively less than 2 hours to achieve the PCB layout while for the second one, the process necessity less than half an hour. For these reasons I decided to make the prototype PCBs using the second method as it is less time consuming and more precise.
At the end of both methods two mains error could occur:

1. Ruins of the tracks/footprints due to an excess in time during the etching procedure

2 Short circuits (SCs)

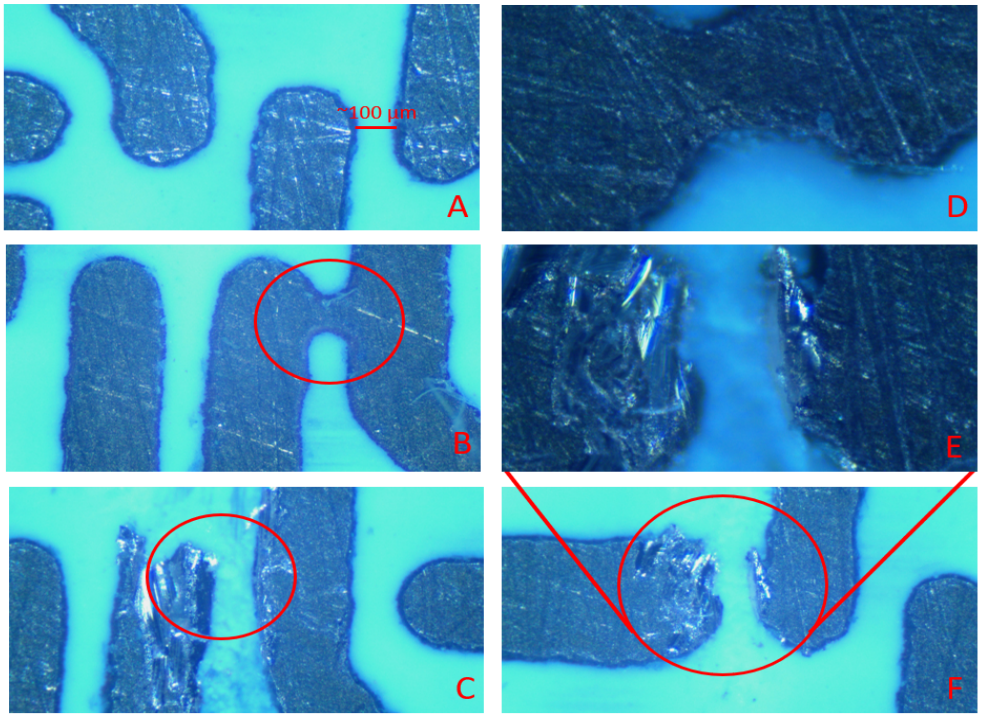

Figure 4.8: (A) Typical distance between two tracks. (B, D) A short circuit. (C and F) Result after scalpel procedure. (E) A zoom of F.

In case of the first error all the process must be repeated.
If the error is the second one, a manual procedure can save the PCB and make it usable. Indeed, a scalpel can be used to cut the possible SC. To have clear idea of the situation, some pictures are taken from a microscope and, two short circuits are illustrated in Figure 4.8 (Fig. 4.8B and 4.8D). The SCs appearing between tracks are removed scraping them off with the scalpel (Fig 4.8C and 4.8F): this step is repeated until no SC is detected with a multimeter.
A zoom has been performed (Fig. 4.8E) to see clearly the obtained tracks and using a multimeter the absence of SCs in the PCB has been verified.
This simple but efficient procedure save time avoiding to repeat all the method from the starting point.

4.4.4 Soldering

For what regard the soldering procedure only the pin headers are soldered manually. The other components are soldered using the Voltera V-One coming from the Voltera company.
At first the Gerber file obtained from the PCB design illustrated in section 4.3 are loaded into the Voltera software. This software is really user friendly and guide you step by step.
Once the PCB is in the Voltera plane and the Gerber file is loaded, the machine starts the automatic calibration. After that, a manual calibration is necessary: it ask the user to manually moves the calibration tip to at least three soldering pad of the loaded footprints and automatically interprets the orientation of the board. The solder paste dispenser is manually pushed through a gear: rotate clockwise till the paste comes out of the nuzzle and then a counterclockwise rotation is performed. This is done in order to not have to many pressures on the syringe. The syringe, containing the paste, is then mounted instead of the calibration tip. The software performs then a z-axis calibration and allows the user to put soldering paste in a desired position in order to be sure that the calibration procedure is performed correctly and the syringe dispense the right amount of solder paste.
At this point it is possible to put the soldering paste in all the necessary pads automatically and, if a pad is missed or not correctly covered with solder paste, the software allows to remake the procedure only in the necessary pads. The solder paste used with Voltera V-ONE is the HushedHeron.
This machine is not only able to dispense the solder paste but also to make a reflow soldering. Indeed, once that the components are manually placed in the PCB and the syringe containing soldering paste is removed from his housing, the Voltera can heats its plane and solder the components.

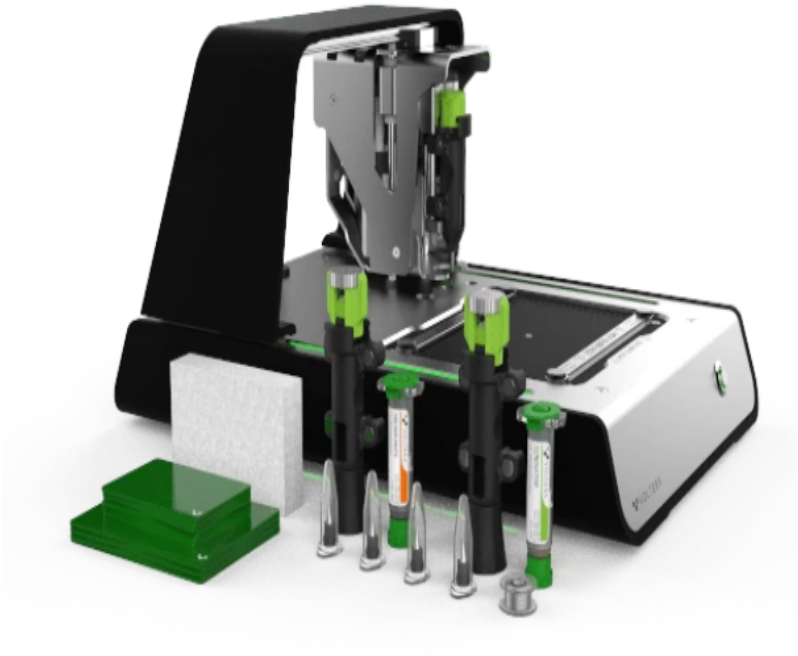

Figure 4.9: Voltera V-One

These procedure takes about 30 minutes: the Voltera achieves 147° Celsius in 15 minutes and then starts cooling down in the remaining 15 minutes.
Finally, after this procedure, the PCB is almost complete and only the pin headers remain to be solder but, this time it is done manually.
Once the reflow soldering is complete, it is important to make some tests on the obtained PCB in order to be sure that no SCs or mismatch occurs. This is important mainly because the pick and place procedure it's performed manually, and it is the most probable source of error.
In same cases if a mismatch occurs, it is possible to desolder and solder again the component, saving the PCB.
In Figure 4.9 a picture of the Voltera V-One is shown, and it is possible to see also syringe dispensers, tips and solder paste/liquid metal containers that are usually used with this machine.
More information of this machine can be founded in the user manual [53].

4.5 Final Designs

After testing the PCB and the firmware, two designs were developed for the final devices: one with the temperature sensor only 4.10 and the other with both temperature sensor and PPG sensor 4.11.
Both the systems contain a sliding switch used to turn ON or OFF all the system and a push button used to perform a reset when the device is applied to a patient/user in order to boot the device and make the firmware starts running.
In the top layer of the PCB there are also two connectors, one for the USART communication and one for the SW debugging. The connectors are plane pads and will be used for communication purposes.
The electronic connection is obtained through bended springs, in the docking station, pushed against the pads of the device.
These will be used to communicate with a host PC for collecting, processing and plotting the stored data (top-bottom side) or for debug porpoise (top-right side).

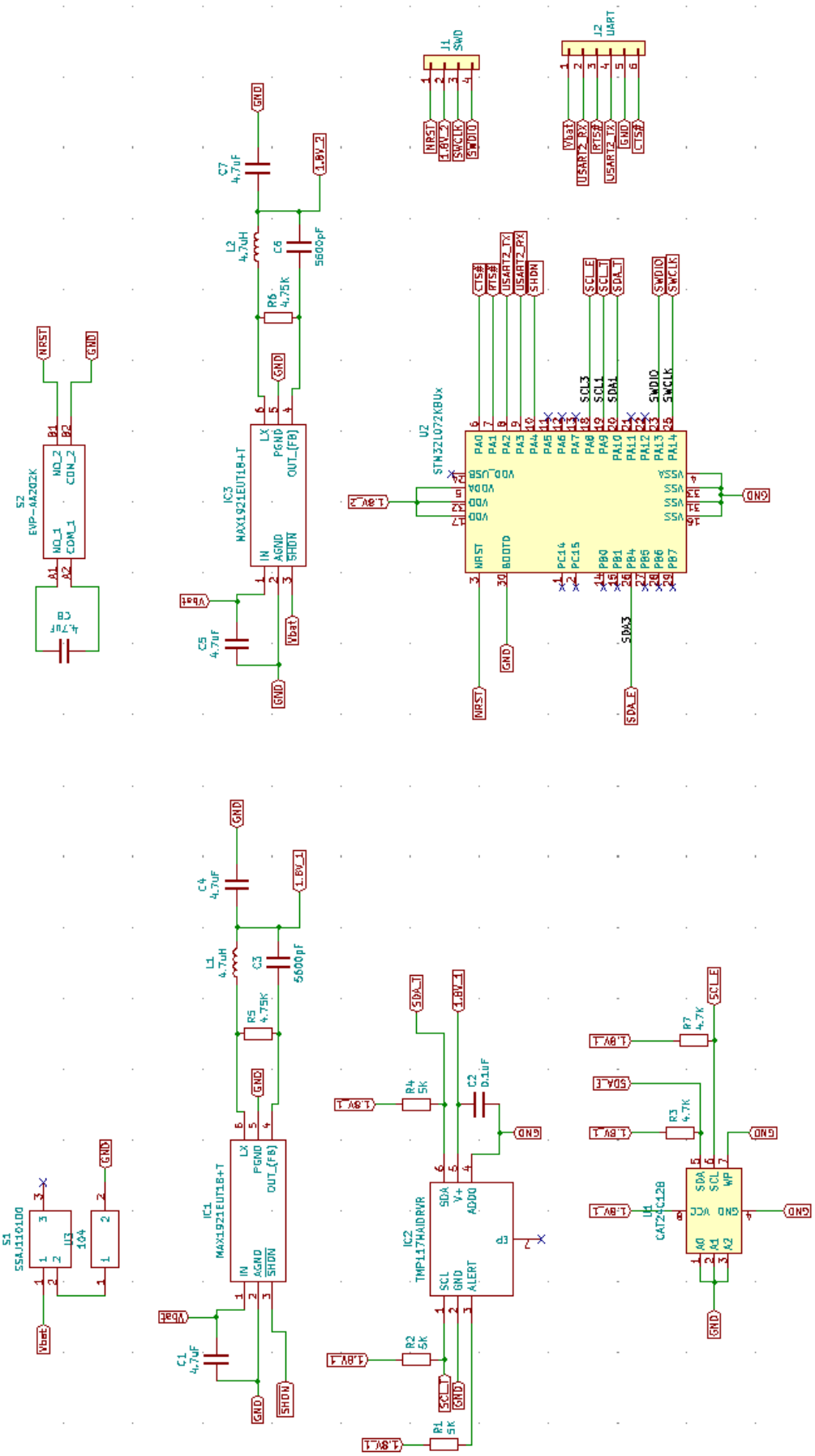

Figure 4.10: Measuring device electronic schema - First design

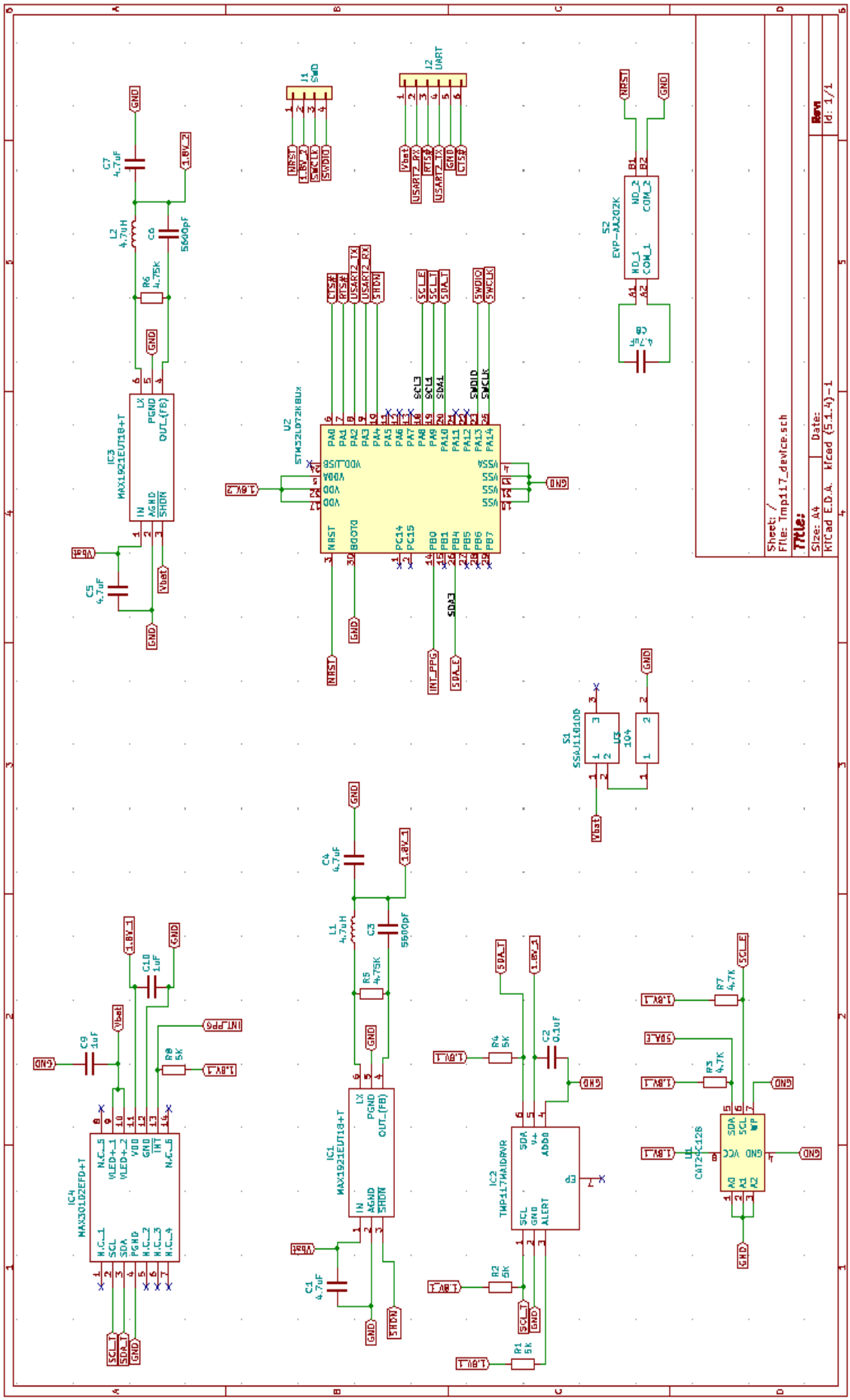

Figure 4.11: Measuring device electronic schema - Second design

For more details, the reader is invited to see Appendix A where the Gerber files are included.

CHAPTER 5

FIRMWARE

This Chapter has the aim to introduce, explain and document the firmware and algorithms that make the measuring device works properly.
This section can be also used as a user manual document or a starting point for future developments on the B-HOT prototype device.
At first, a general view on the architecture will be provided while, in the next sections, a in deep analysis will be performed.

5.1 General Architecture

As mentioned in chapter 4, the STM32L072KBU6 microcontroller is based on ARM® Cortex® M0. It allows the possibility to have low-power real-time clock at different velocities [54].
Different Integrated Development Environment (IDE) can be used while programming the MCU. ST Microelectronic, for example, suggests its own software, the STM32Cube IDE.
For this project, μVision®5 coming from Keil, property of ARM, has been used.
To simplify and better manage the time, the STM32CubeMX software has been used. This software is a graphical tool that allows the user to easily configure a STM32 microcontroller and/or microprocessors. Cube MX generate the initialization C code for the selected ARM® Cortex® -M core based on the chosen setup. In this case the clock is set to 131 KHz and

two timers (TIM) are set to trigger an interrupt at a certain rate. Different General-Purpose Input/Output (GPIO) pins are set for USART communication, for sensors' interrupt pins, for I2C communication, for debugging and one is set as switch for turning on and off a DC-DC.
The firmware is organized in different entities. Each entity manages a specific component but in same case entities use functions of other entities. The MAX30102 manager is an open source library coming from Maxim Integrated [55] and has been modified to work with the developed device. The other libraries have been written specifically for the prototype.
The architecture of the B-HOT device can be divided in two main blocks: one regarding the device itself and one developed in order to allow communication between the device and a host computer.
In Figure 5.1 there is the representation of the two blocks: communication, power source and main signals are highlighted.
The blocks labelled as manager are tasks that deals with a specific component. For example, MAX30102 Manager is a task that takes care about the MAX30102 configuration, data processing, communication protocols and others.
Figure 5.1 A represents the communication between the B-HOT device and a host pc. The communication is serial and the Universal Synchronous-Asynchronous Receiver-Transmitter (USART) peripheral will be used. The Serial communication is the process of sending data in a sequential fashion. In order to link the USART peripherals of the B-HOT device with a computer's serial port, the TLL-232RG-VREG1V8-WE cable has been adopted. It is a TTL (Transistor-Transistor Logic) serial UART to USB (Universal Serial Bus) Serial Converter. Inside the cable there is the IC FT232RQ USB to Serial UART interface device that is responsible of all the USB protocols and signalling. For more detail read [56].
USART in Asynchronous mode can also perform full duplex operations; these mean that transmission and reception can occur at the same time [57].

In section 5.2 we will see how the data are treated and how the communication between device and host pc is performed and processed. The diagram in Figure 5.1 B shows the architecture of the B-HOT device itself. Here we can see how the communication between components is performed while an in deep explanation will be completed in the next subsections.
All the components inside the device communicate with each other using the Inter-Integrated- Circuit (I^2C or I2C) protocol while some other operations are triggered using the MCU's pins.

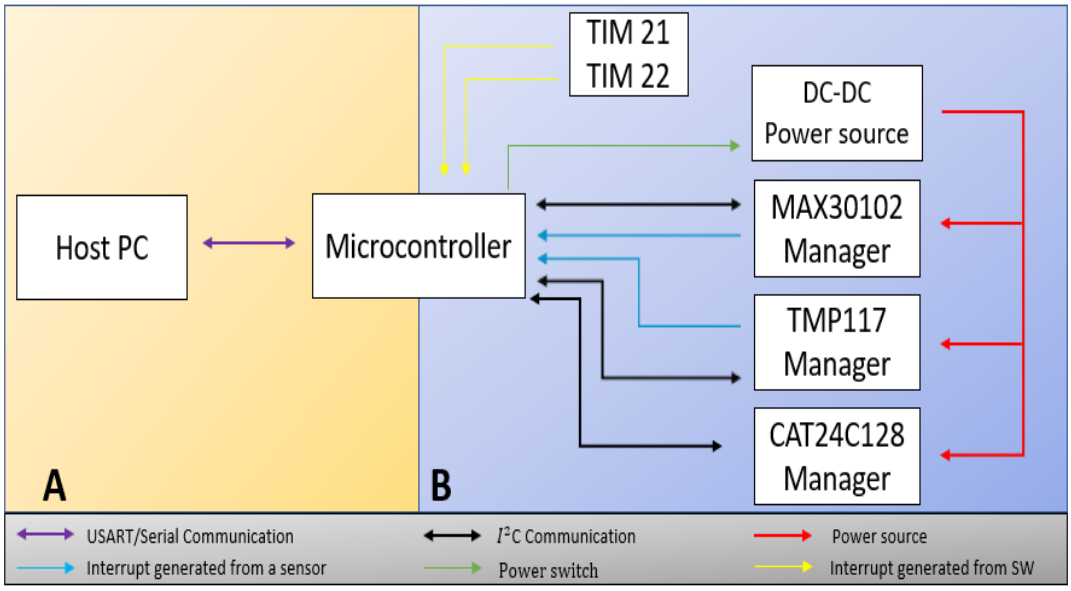

Figure 5.1: B-HOT communication and main signals diagram. The legend in the bottom side clarify the role of each arrow present in the graph.

The I2C communication protocol was invented in 1982 by Philips Semiconductors (NXP Semiconductors). This protocol allows the communication between two or more I2C devices using two wires (plus one for the common reference). One wire is used to send data through serial (SDA) while the second one is used to send the clock (SCL).
This protocol offers the possibility of multi-master and/or multi-slave communication. In this architecture the multi-slave configuration has been implemented: the MCU represents the master while the other components

are the slaves. This configuration is allowed because each component has its own identification number, or address, composed of 7 bits plus one to choose the mode (read or write).
The master sends the start condition (SCL = 1, SDA = 1 to 0) and the slave's address. After the Acknowledge bit the data are transmitted between MCU and the called component till the stop condition is not asserted (SCL = 1, SDA = 0 to 1) [58].
In Figure 5.1 it is not mentioned the four pads available in the top side of the device that allows SW debugging. The debug can be performed using the Serial Wire Debug (SWD) or Serial Wire Debug Port (SW-DP). It is composed of 2-pins: one is associated with the clock (SWCLK) and the other, single wire bi-directional, is used for the data (SWDIO). Other two pin are used to have same reference for ground and source power.
SWD uses an ARM CPU standard bi-directional wire protocol, defined in the ARM Debug Interface v5 [59]. Using the ST-Link V2 from ST and the mentioned μVision 5 IDE, it is possible to perform debug actions on the device.
Now, a in deep view will be performed on the different managers cited in the block diagram in Figure 5.1B.

5.1.1 TMP117 Manager

This manager is a task containing different functions that handle the TI TMP117 temperature sensor. In the file tmp117.h there is the register map of the component and, all the registers are saved with their respective address. Beyond that, in the file there are different function prototypes to control the sensor.
In tmp117.c there is the implementation of all the function present in tmp117.h. Among all these functions, these are the most important:

- void TMP117_set_Configuration();

- void TMP117_set_Temperature_Offset();
- void TMP117_Initialization();
- uint16_t TMP117_get_Temperature();

The first one is used to set the configuration of the sensor. This function set the bits contained in the Configuration register. This allows to choose and change among a different type of configuration. For the project, the used configuration is shown in Table 5.1.1:

Table 5.1: Configuration of the TI TMP117

Configuration	Set	Details
MOD	One-shot conversion	Set conversion mode: OS takes only one value of temperature.
CONV	15.5 ms	Conversion cycle bit: time required to have a conversion of a sample.
AVG	No averaging	Conversion averaging modes: It is possible to make multiple conversions (8 to 64) obtaining as a result one sample that is the average of the multiple conversions.
POL	Active high	Alert pin polarity bit: It is possible to set the polarity of the pin that asserts the interrupt flag.
DR/Alert	Alert pin reflects the status of the data ready flag	Alert pin select bit: the alert pin can be set for data ready flag or alert flag.

With TI TMP117 it is possible to choose an offset for the temperature and this is the reason of the second function. For the project the offset is set to 0 since it is sufficiently accurate and precise.
The third function is used to initialize the sensor in order to start collecting samples. It calls the first and second functions plus other two function that can be used to set high and/or low limits of temperature in case the alert

pin is used for the alert flag.
The last function is the most used of this manager and its scope is to read the temperature of the sensor. The function returns a 16-bit value that can be translated into a floating-point value of temperature by multiplying it for its resolution (7.8125° mC).
We will see in the next section that for protocol reasons, the 16-bit sample will be first divided into two 8-bit values and then stored.

5.1.2 MAX30102 Manager

This manager is a task containing different functions, some of them handle the Maxim Integrated MAX30102 optical sensor, others are used to process the data. Indeed, the script named MAX30102 contains the register map and the functions responsible for example of reading the data. The most important function here is the Max30102_Calc(). This function initializes the sensor , turn on the red and infrared LEDs and starts to read samples from both channels: every 17 samples the sensor assert the interrupt pin and the data are red. When 527 samples are collected the LEDs are turn off and the first 27 sample are discarded because are usually random values coming from the sensor's data register. It calls then the function contained in the script algorithm, maxim_heart_rate_and_oxygen_saturation(). It process the collected data and calculates the value of heart rate (in beat per minute, *bpm*) and blood oxygen levels (the percentage obtained from the red/infrared signal ratio).
The mean value is calculated and removed from the IR signal. Then, the four points moving average is performed as shown in formula 5.1, considering $0 < k < 500$.

$$IR_{buff}[k] = \frac{IR_{buff}[k] + IR_{buff}[k+1] + IR_{buff}[k+2] + IR_{buff}[k+3]}{4} \tag{5.1}$$

The samples are then used to calculate the difference of smoothed IR signal by subtracting two consecutive sample ($IR_{diff}[k] = IR_{buff}[k+1] - IR_{buff}[k]$). A two points moving average is then performed and a filter, the Hamming window, is applied for tapering porpoise. The number of peaks contained in the PPG signal is calculated and the heart rate in beat per minutes is evaluated as shown in formula 5.2. The value 10 (seconds) represents the measuring period: it is the time required to collect 527 samples from the sensor. The multiplication (60 seconds) is performed to have the value in beats per minutes.

$$HR = num_{peak} * \frac{60}{10} \tag{5.2}$$

In order to evaluate the blood oxygen levels, it is important at first to calculate the variable R. It is the ratio of AC/DC signals of both the LEDs and it is showed in formula 5.3.

$$R = \frac{\frac{AC_{red}}{DC_{red}}}{\frac{AC_{infrared}}{DC_{infrared}}} \tag{5.3}$$

Once R is determinate, a lookup table is used to find out the estimated value of blood oxygen level. To obtain the lookup table, typically, an empirical method is used to collect data from different subjects. Maxim Integrated has performed this method and provides a lookup table for this scope.
Skin tone, age and overall health can affect the accuracy of the calculation.

5.1.3 CAT24C128 Manager

This manager is a task that hand the read/write operations for the CAT24C128 EEPROM. It is a 128 Kbit memory organized in 256 pages, each of 64 bytes organized in 64 rows of 8-bit.
Four function are written to accomplish different tasks. The first two of them are used to read or write a specific page. The other two functions are

used to erase the data from a specific page or erase completely the EEPROM.
No function has been written to write to a specific position of a certain page. This because a high number of reading/writing operations affect the power consumption, decreasing the life of the battery. For this reason, the writing operation is executed only when 64 bytes of data are collected while the reading operation is performed only when the data are required from the user.

For more detail, the reader is invited to see Appendix B and [55].

5.1.4 Algorithm: Main

Before talking about the main, the MCU configuration will be treated. The Software Debugging(SW) Interface is enabled and set for debugging porpoise while the UART interface is enabled for communication at 4800 bit/s.
Two timers (TIM) are set in interrupt mode in order to have an interrupt event at a certain rate in such a way to start a new measurement. In order to set the prescaler (PSC) and the period (PR) of both TIMs, formula 5.4 has been used.

$$UE = \frac{f_{CLK}}{(PSC+1)*(PR+1)} \tag{5.4}$$

The Update Event (UE) represents the frequency rate at which the interrupt will be asserted. In order to have a temperature value every three minutes, TIM22 has been configured with $PSC = 9215$ and $PR = 2599$. This, with $f_{CLK} = 131.072KHz$, brings to a value of $UE \approx 5.35mHz$, lower then the desired one ($5.55mHz$). This mismatch lead to an interrupt every 184 seconds (3 minutes and 4 seconds). Due to this, a measuring period of 72 hours lasts instead 73 hours and 36 minutes.

TIM21, related with the PPG sensor, has been set to assert an interrupt every 40 minutes ($\approx 4.16mHz$). Selecting $PSC = 32767$ and $PR = 9599$, the obtained result is equal to the desired one.
In figure 5.2 a pseudo-code of the main is presented. It contains the main steps and the most important variables. At first the program read the UART and based on its content, the *WhatToDo()* function decides the next step choosing between three different options:

1 Nothing: No input comes from UART and the code continue the measurements

2 READ: The input from USART is "READ" and the code sends on USART all the samples

3 DELETE: The input from USART is "DELETE" and program erases all the values in the EEPROM

Once the *WhatToDo()* function ends its job, the program looks if an interrupt has been asserted. In case TIM21 asserts an interrupt(FLAG_T = 0), a new sample of temperature is red from the sensor, and only if we have collected 32 samples a write operation is executed. The reason of this is to reduce at minimum the total number of write operation in order to save power. In any case, when a new sample is collected, the value of the FLAG_T is set to 1.
In case TIM22 asserts an interrupt(FLAG_PPG = 0), the PPG sensor is turned on and the calculation for a new sample of HR and SpO_2 is performed. Also here, for the same reason as before, a write operation is executed only when 64 samples are collected. At the and, the value of FLAG_PPG is set to 1.
The variables Flag_W and Flag_PPG_W are used to avoid oversampling. Indeed, when the number of desired/required samples are collected, we want to avoid errors due to overwritten measures. When one of these variables is set to 1, a message is sent to UART saying to the user: "Sample

collection completed".

Now, the user can read the data using the Graphical User Interface (GUI) written on python.

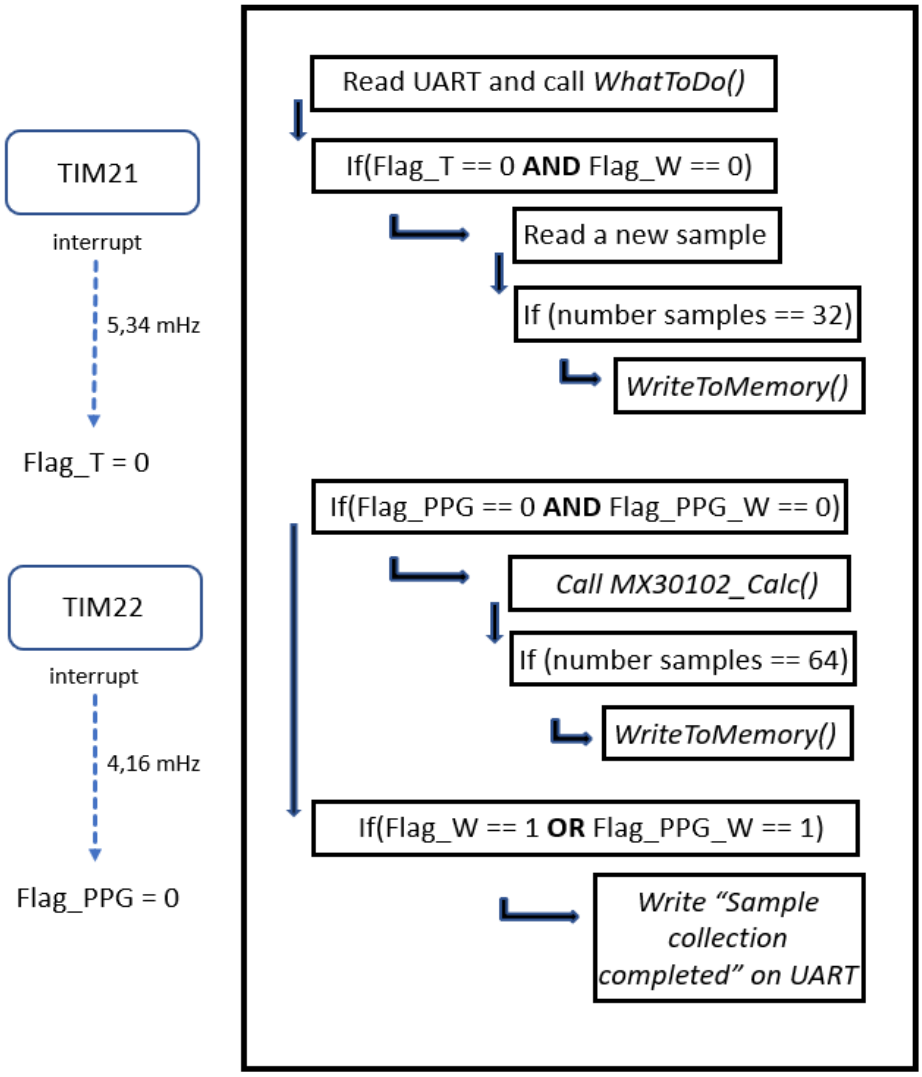

Figure 5.2: Pseudo-code contained in the while(1) loop of the main

For more detail, the reader is invited to see Appendix B.

5.2 Python Graphical User Interface

In this section the Graphical User Interface (GUI), written on python, will be discussed and the main functions analyzed. A GUI is a program that includes graphical controls, which the user can select using a mouse or keyboard.
The desktop application is written in Python 2.7 using Tkinter [60], a GUI Programming toolkit for Python. The software allows the user to communicate with the device, through USART, using a four buttons command window that appears as soon as the application is executed.
When the program is running, the user can choose between four different options by simply clicking with their mouse.
The four possible choices are explained and discussed below:

- **Read Temperature**

  This button must be used with the first design (see section 4.5). When the measuring period (72 hours) has been completed, the user can use this button to collect data.
  First, the user is asked to enter a name for the file where the data will be saved. Some rules guide the user during the name selection such as no special characters are admitted.
  After the selection, the program sends through USART the word 'READ' and waits for data coming from the device. The user will see the message 'waiting for data...' till the device starts sending the data.
  When the data collection process is finished, the user can select again between all the possible buttons.

- **Read PPG**

  This button must be used with the second design (see section 4.5).
  After 48 hours of measuring period, the user can select this button and starts collecting data from the device.
  First, the user is asked to enter a name for the file where the data will be saved. Also here, some rules guide the user during the name selection such as: no special characters are admitted.
  After the selection, the program sends through USART the word 'READ' and waits for data coming from the device.
  When the data collection process is finished, the user can select again between all the possible buttons.

- **Plot**

  This button allows the user to plot the data obtained from the device. This program, based on the number of collected data, understands which is the design case to be considered (section 4.5) and performs the adequate plot. For design 1, the 72 hours measuring period is showed. For design 2 the measuring period is 48 hours and heart rate and oxygen saturation levels must be plotted too.
  When the user selects this button, a file dialog box appears. This is graphical control element that allows the user to choose a file in the file system. When the selected file is correctly formatted a plot operation is performed.

- **Delete memory**

  This button allows the user to perform an erase operation on the EEPROM mounted in the device. These is a sensitive process because we are cleaning all the memory and the stored data will be lost. For this reason, if the user selects this option, a message appears on the screen warning the user about the consequences of deleting all the stored data. At this point, the user must type specifically 'y' to continue, any other keyboard keys cancel the process.
  The memory erase operation is performed by writing on USART the word 'DELETE' while the physical operation will be executed by the MCU.

Different error situations such as typing a name outside the rules, device connection not founded and other have been considered. These are handled by printing, for the user, an error message specifying the type of error and the possible causes.
The data are first collected and then saved in a file with .txt extension, where the file name is chosen by the user following some name limitations. These data can be imported to other software like excel or MATLAB that allows the possibility to perform other kinds of data processing or simply different plots.
A picture of the GUI is showed in Figure 5.3: on the left there is the window menu and on the right a shall window.
On the top side of the menu, there is written: "Choose what you want to do:". This window allows the user to select one of the four possible options described before.
On the right of Figure 5.3, a shall window is used to allow communication between user and desktop application. The user can use it for different reasons such as type the file name, choose a sub-option, read error or in-

formation messages etc..

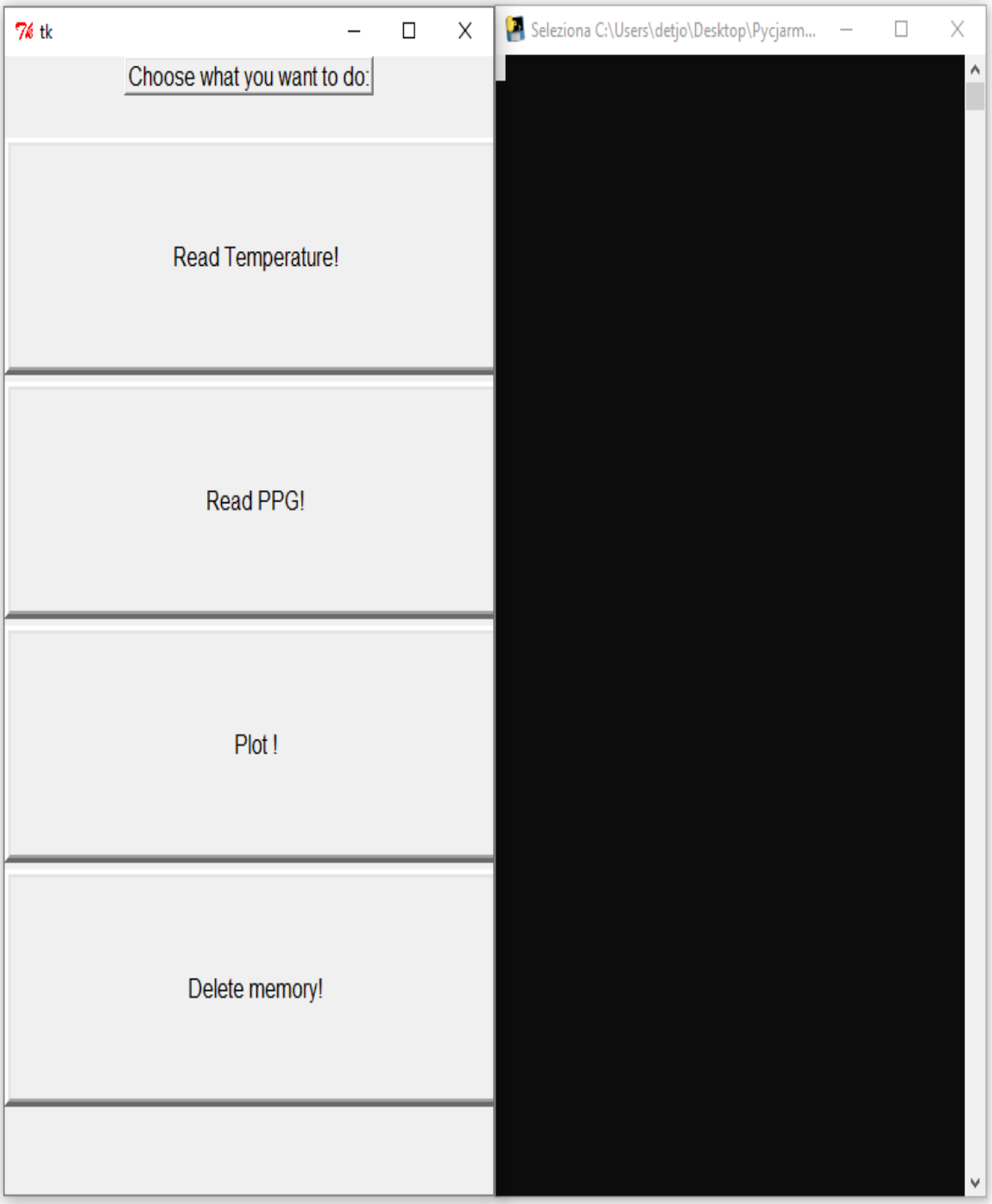

Figure 5.3: Desktop Application. On the left the GUI menu with the four buttons. On the right a shell window used to send or receive messages to the user

For more details, the reader is invited to see Appendix C.

CHAPTER 6

TESTING

This Chapter will be used to present the different tests performed on the device and can be divided in two parts. The first one will discuss a test performed when the hardware of the device was assembled in a modular approach and regards the PPG sensor.
In the second part, the final device will be presented, and some errors discussed. Then, some tests on the device will be performed and discussed.
The algorithms described in chapter 5 are specifically coded for the prototype device and the code will be used in both parts, modular and compact.

6.1 MAX30102 Anatomic body position

As described in section 2.4.2, the anatomic body location in which the PPG sensor will be placed is very important. For this reason, four anatomic positions will be compared to see which one gives better results in term of quality of the signal, not accuracy.
The chosen location are wrist, chest leg and finger. The tests are performed with a 10 seconds window of measuring period and the sampling frequency is 50 Hz. The data are collected on a host pc using the serial communication protocol and then saved on a txt output file.
A script, written in MATLAB R2018b, was used to import the PPG data and then perform data processing and analysis. The data are first loaded from the txt file and then divided according to the channel they belong to

(RED, IR).

On one side, the data are first normalized, the signal is scaled around one, and then plotted together as we can see on the left side of figure 6.1. The red and blue signals are respectively the RED and IR channel and the same holds for the graphs on the right side.

The data coming from the sensor are also passed through a filter to remove the DC component and then, a low pass filter with a cut-off frequency around 3 Hz was applied. This removes high frequency noise and "clean" the signal.

The output signals, after this process, are illustrated on the graphs on the right side of figure 6.1.

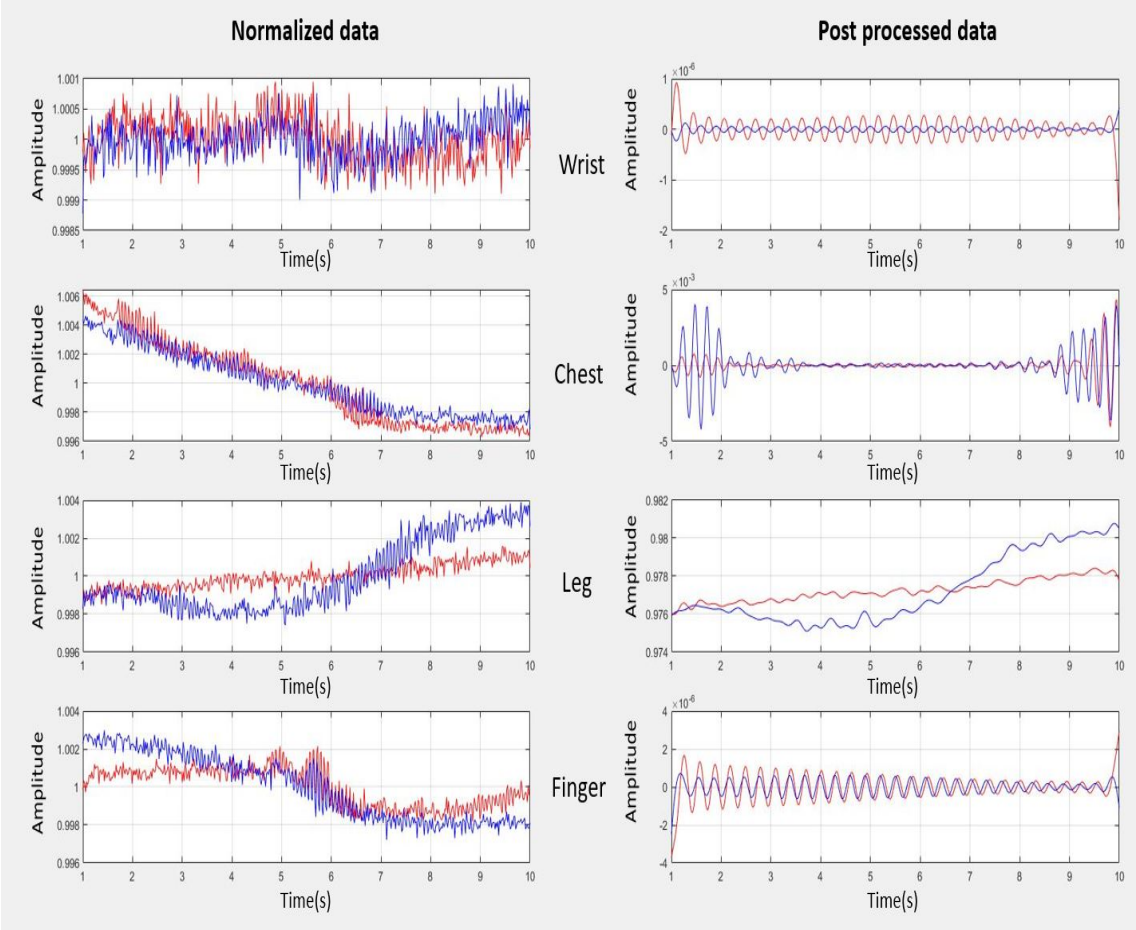

Figure 6.1: The data collected from the sensor are first normalized (the signal is scaled around 1) as showed on the left and then filtered (on the right). The data of each measurement are two signals; the red is referred to the RED channel and the blue one to the IR channel

The calculation of the HR can be easily calculated by counting the number of pecks in the signal while SpO_2 calculation needs the computation of the AC/DC ration of both channel and the use of the SpO_2 table provided by the sensor's company.
These signals were compared with [10], the same sensor was used in both works, and similar signal quality was founded. Among these four anatomic body locations, the finger is the best spot for PPG measurements.
The wrist has better results with respect the chest while the leg is the worst between these four anatomic locations. In the next section, an accuracy test on both sensors will be performed.

For what regards the temperature sensor, as discussed in section 2.2, the best anatomic body location for BT measurements are those that are less exposed to the temperature of the environment.
Since the motivation of this work aims to monitor body temperature, the device will be placed in an anatomic location that gives more accurate measures for temperature with respect HR or SpO_2.
The chosen place is the chest and the device will be in the chest strap category (see chapter 3).

6.2 Device Testing

When the ordered PCB arrived and the component was soldered, hardware failure test was performed on each component. At first, was verified that possible short circuits did not occur. This was performed using a multimeter in the continuity mode. This mode measures the resistance between two points of a circuit and if the value is low (a few Ohms), there is a direct connection between the two points and i.e. a short circuit occurs. The test demonstrates that no short circuits was found in the boards.
The desired connections between components was correct except for one

pin, the Boot0, that was unconnected instead of connected to ground. Boot0 pin is used to start the bootloader and consequently the firmware of the device. Indeed, from the documentation of STM [61] and from the written firmware, if boot 0 is connected to ground and a reset event is asserted, trough the push button, the written algorithms will start.
Even if this seems to be an important problem, it was discovered that, pushing the reset button around 30 seconds after the device it is powered, in some way overcome the problem and the device starts properly.
As highlighted from STM, all the free pins of the microcontroller were set as analog in order to optimize the power consumption.

6.2.1 Device presentation

As described in Chapter 4, the device dimensions are 31.5 x 41.15 x 6 mm. A picture of the prototype device is showed in figure 6.2 (1, 2, 3). The dimensions are highlighted and in picture labelled 3, the device in illustrated inside its own case. The case is 3D-printed with a flexible 3D filament. This allows to use it as a clover of the device that will protect it from possible shocks. On the bottom side, the one that will stay in contact with the body (6.2 3), a hole is made to allows the sensors to stay in contact with the skin. For the same reason, the bottom side is thinner with respect the other parts of the case. Indeed, it measure a bit less than one millimetre, while the other sides are thicker and measure around two millimetres. In figure 6.2 5, it is possible to see the device applied on the anatomic body location, the chest, of a patient. From this picture, it is possible to compare the body dimensions of the patient with the device dimensions. In figure 6.2 4, the device is applied to a patient with a commercial adhesive plaster that does not irritate the skin. This picture illustrates also the starting point of a typical monitoring test on a patient.
In figure 6.3 the device and the 3D-printed docking station are illustrated. The picture labelled A is a render image of both the case and the docking

station in the same position of the real picture labelled B. In figure C, the device is plugged into the docking station that allows serial communication between the device and a host pc using the proper cable, as described in Chapter 5.
The docking station dimensions are highlighted in figure 6.3 D and are 35 x 23 x 23 mm.

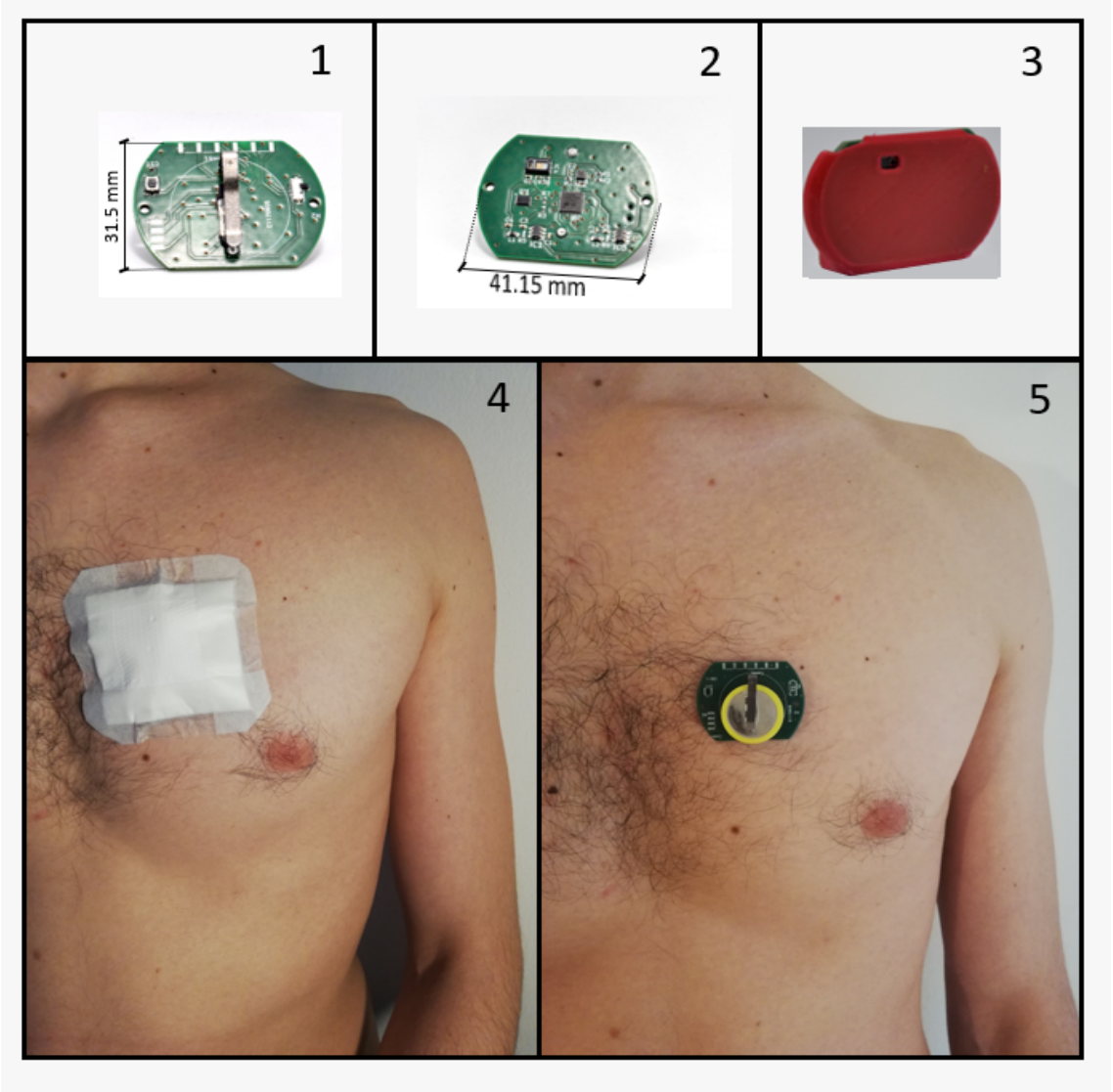

Figure 6.2: The first two pictures are respectively the top and bottom side of the device while the third one is the device inserted in its case. The last two pictures illustrate the device applied on the body of a patient; first with an adhesive plaster and in the last one the device is only leaning on the patient body.

The circular part of the case illustrated in figure 6.3 (A, B, and C), where there is the drawing of a beat, is used to maintain stable the battery and avoids power disconnections due to misconnection of the battery with the device.
Looking A and C, it is possible to notice the holes made for the switch and the push button, but also the holes that can be used to attach the device on the body (like a belt).
These holes exist but will not be used. Indeed, as illustrate in figure 6.2 4, an adhesive plaster will be used for this scope.

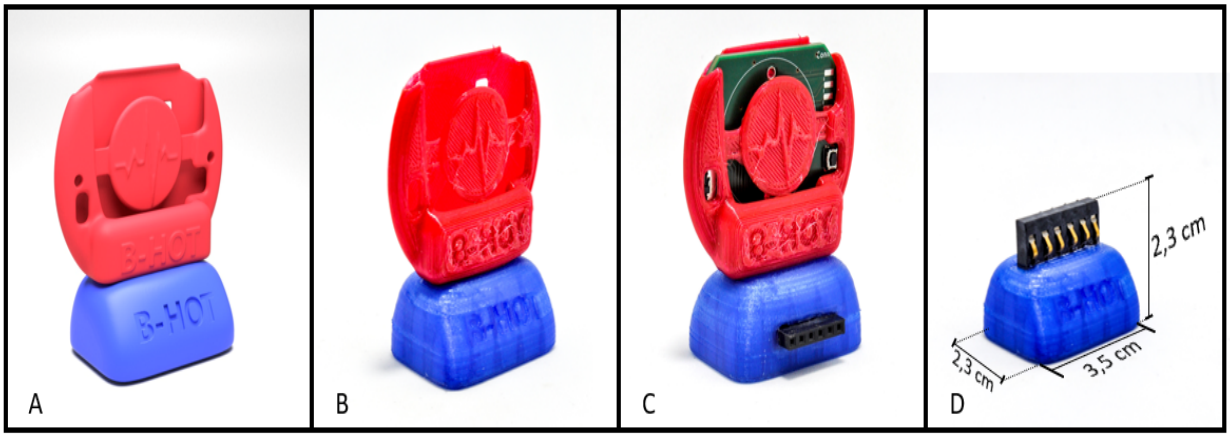

Figure 6.3: The first image is a render picture of the second one. In C the device, inside the case, is plugged in the docking station. The last picture illustrated and highlight the docking station dimensions.

6.2.2 Sensors Accuracy

In this section, the accuracy of the sensor will be discussed. The tests are performed as follows:

1. The device in applied on the chest of the patient and Temperature and Heart rate values are collected for 1 hour of measuring period.

2. One value of temperature will be collected and stored each minute and one value of HR will be calculated and stored every 30 seconds.

3. The temperature will be measured also every five minutes by hand using the Fluke 62 mini IR Thermometer.

4. HR values will be compared with the Movesense (Maxim Integrated ECG monitor) that calculates the HR every second.

For simplicity, the two tests are divided and discussed separately.

- **Temperature**

  After collecting the data from the device and the Fluke 62, the results were compared. In figure 6.4, in the first graph, we can see in red the temperature measured by the device while, in the second graph, the temperature values measured by the Fluke 62 (blue) are plotted. In the last graph the error between the ground truth and the prototype device is shown. From the graph, it is clear that the error is always lower or equal to 0,073 and greater or equal to -0.15, as expressed in equation 6.1.

$$|error| <= 0.15^{\circ}C \tag{6.1}$$

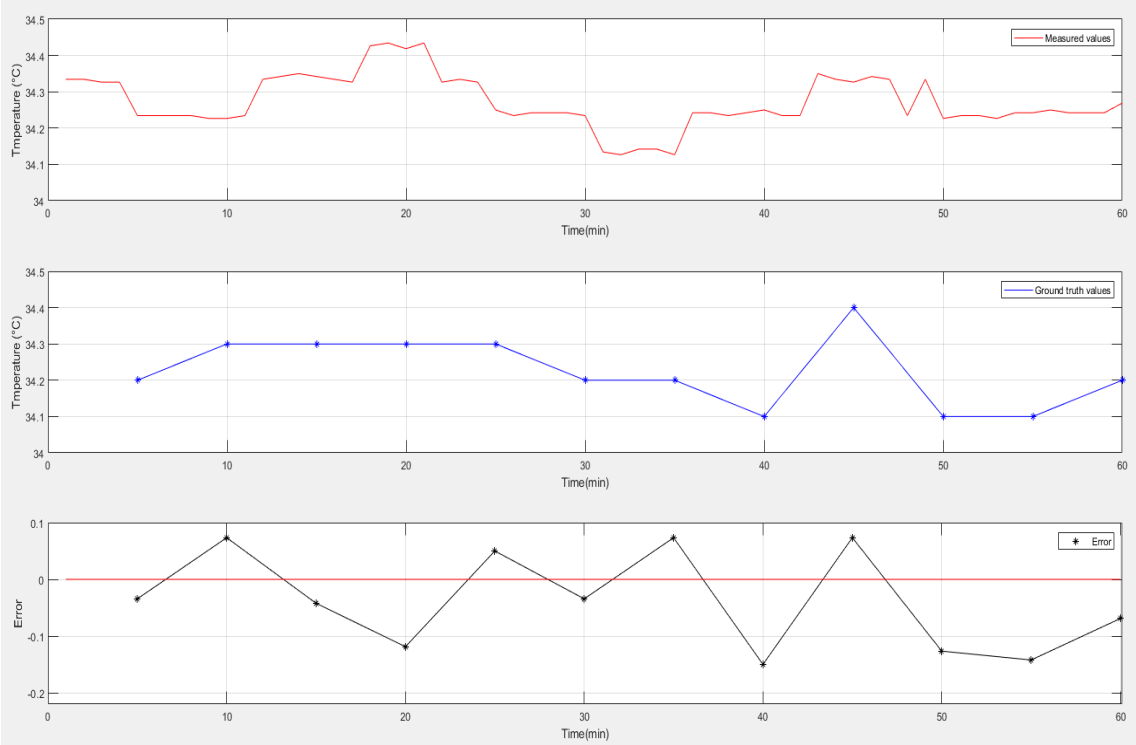

Figure 6.4: One-hour accuracy test for temperature. Starting from the top there are the samples obtained from the device(red). In blue there are the values measured with Fluke 62. In black there is the error computed for each Fluke sample.

This is quite a low error, but a further consideration can be performed on the decimal digits of the samples.

The temperature values obtained from the device are numbers with 6 decimal digits while the Fluke 62 have only one decimal digit. In order to have a better comparison, the data obtained from the sensor was rounded with one decimal number.

In figure 6.5 we have both device samples (red) and Fluke samples (black *). The green signal can be seen as the rounded signal because it is computed by rounding each device sample to one decimal number. In this case, the error is expressed as :

$$|error| <= 0.1°C \tag{6.2}$$

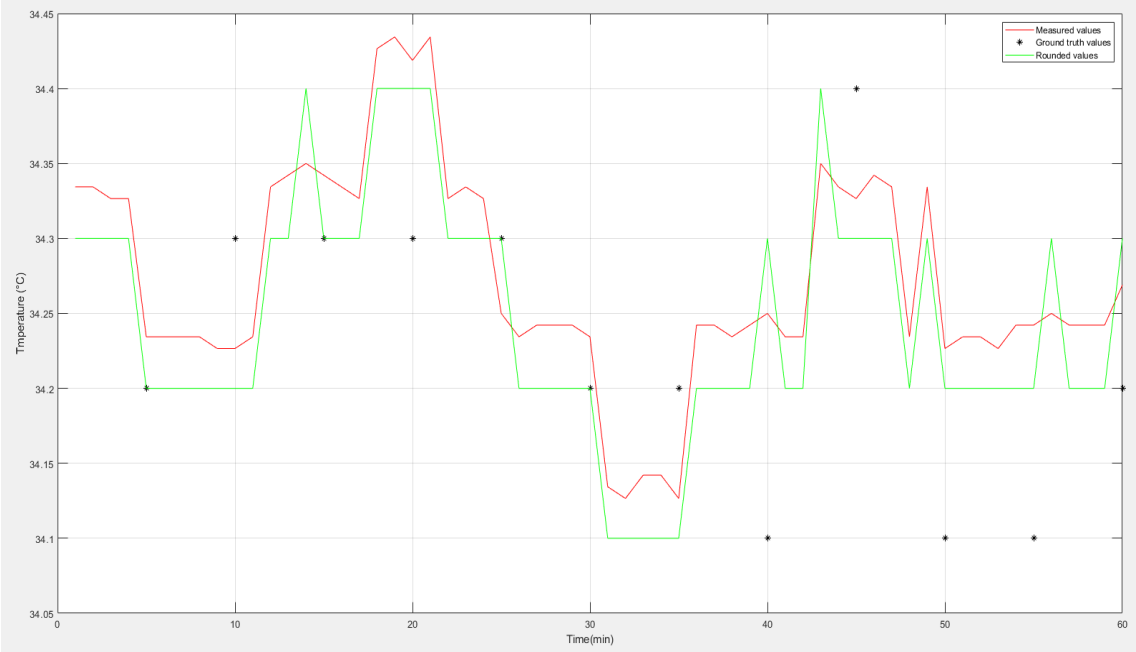

Figure 6.5: In red the signal obtained from the device. The black * are the value obtained from Fluke 62. The green signal is the red one after the rounding procedure.

The statement expressed in eq. 6.2 is true for all the point except for $t = 40$ minutes, where the error is 0.2. The rounding procedure has increased this sample, increasing consequently the error associated with it.

- **Heart rate**

  As for Temperature, after collecting the samples from the prototype device and the Maxim Movesense, the results were compared.
  In figure 6.6 there are the samples obtained from the prototype (blue), the sample obtained from Movesense (red) and the error of the device with respect to the ground (magenta).

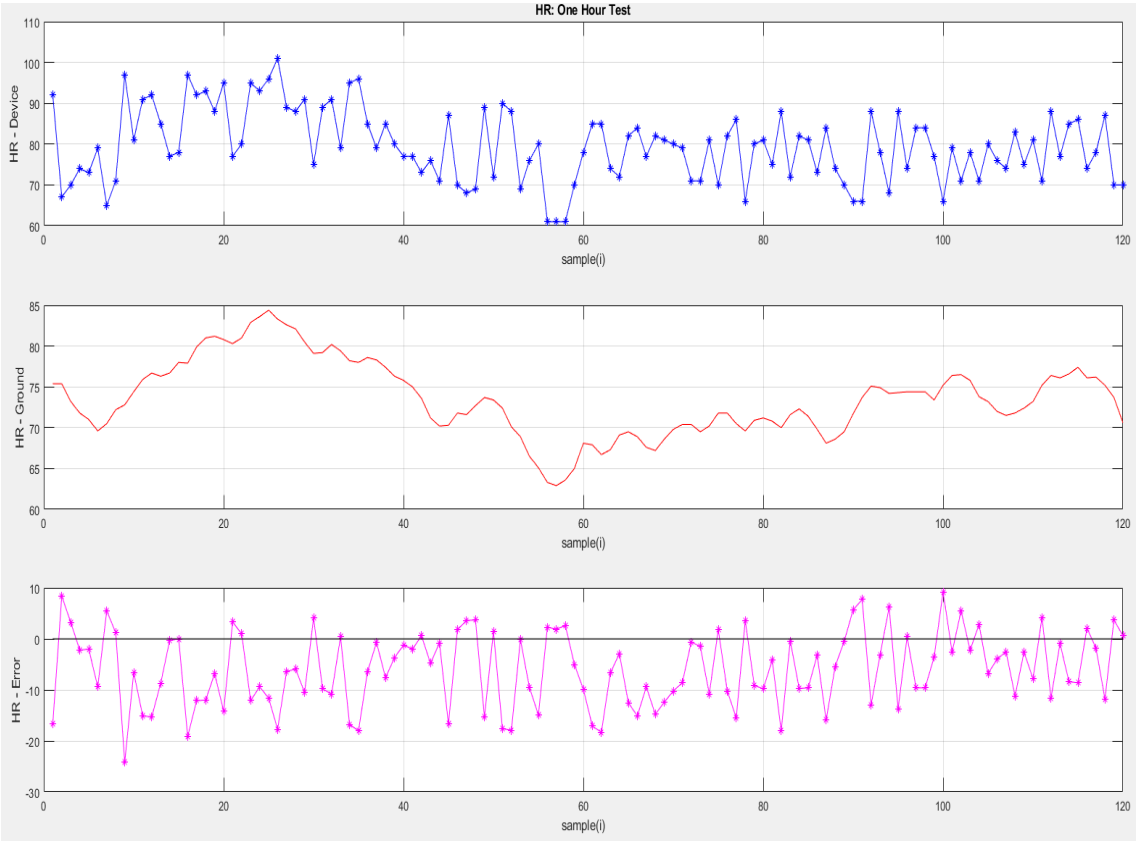

Figure 6.6: One-hour accuracy test for temperature. Starting from the top there are the samples obtained from the device(blue). In red there are the values measured with Movesense. In magenta there is the error computed for each Movesense sample

The error for HR measurements is high and can be defined as described in equation 6.3:

$$|error| \leq 23bpm \tag{6.3}$$

This error was expected to be high because as we say in the previews section, the best place for HR measurements are finger and forehead. The wrist can give better results with respect the chest, but the anatomic body location is then exposed to the environments that compromise the accuracy of the temperature sensor.

6.3 Testing on patient

In order to verify the proper performance of the prototype, some tests were performed. In particular, the aim of this test is to ensure the capability of the device to perform a 72-hour measuring period.
The subject was asked to:

1. Do not wash until the test is completed

2. Take care to not remove the adhesive plaster while dressing or undressing the upper side of the body

The subject was asked then to come in the laboratory for the device application and removal. During the test, the subject was free to do its daily routine; going to do shopping, meet friends etc. When the 72-hour period was passed, the device was removed from the patient and the data were collected on a host pc.
In order to understand how long the battery life would last, different read procedure was performed. The device was able to perform ten complete readings and one incomplete.
After this procedure, it was no more possible to communicate with the device and a new battery is required.

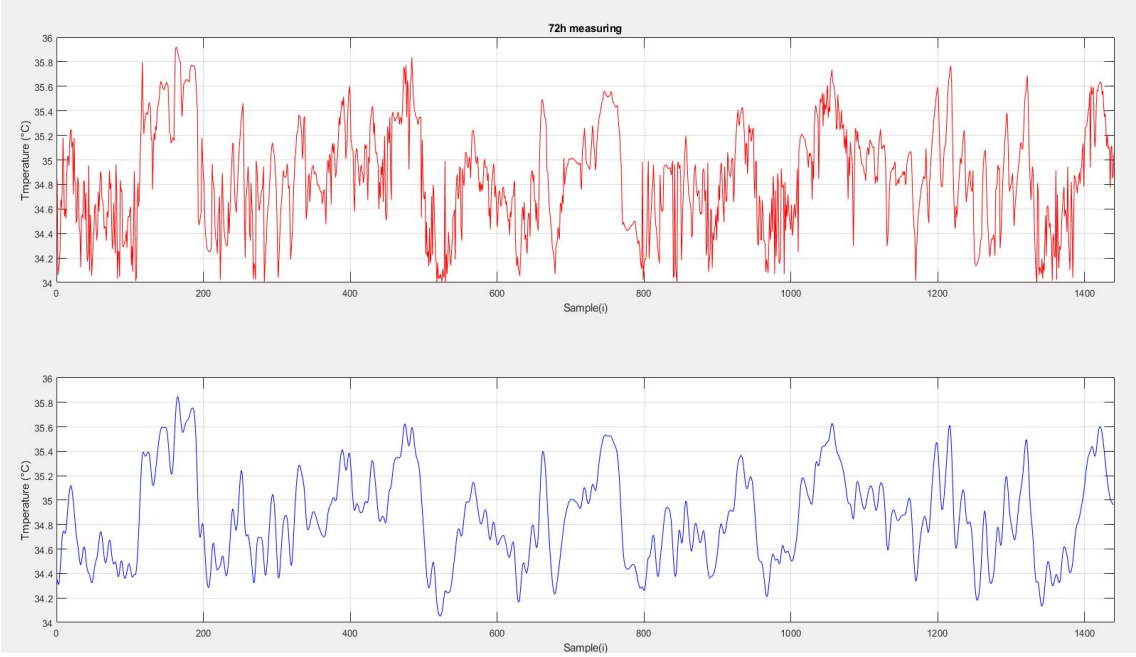

Figure 6.7: A 72-hour Temperature measuring period. In red the sensor collected form the device. In blue the signal obtained after the smoothing procedure.

The collected data are illustrated in figure 6.7. The red signal represents the raw data collected directly from the sensor. These data have been then processed using the MATLAB 2018b software Signal Analyzer. The result is a smoothed signal (blue one in figure 6.7) smoothed with a Gaussian. This technique, in general, consist in creating an approximating function that attempts to modify individual point, higher or lower than adjacent points, reducing or increasing them leading to a smother signal.

In figure 6.8, a "zoom" has been performed. The 72-hour test has been splitted in 3. In particular, this is done in order to highlight each single day separately and see the signal clearly. As for figure 6.7, the red and infrared signals are respectively the sensor data and the smoothed signal.

Taking into consideration that the test starts and ends at 17:30 each day, is possible to divide data between daytime and night time or perform further data processing.

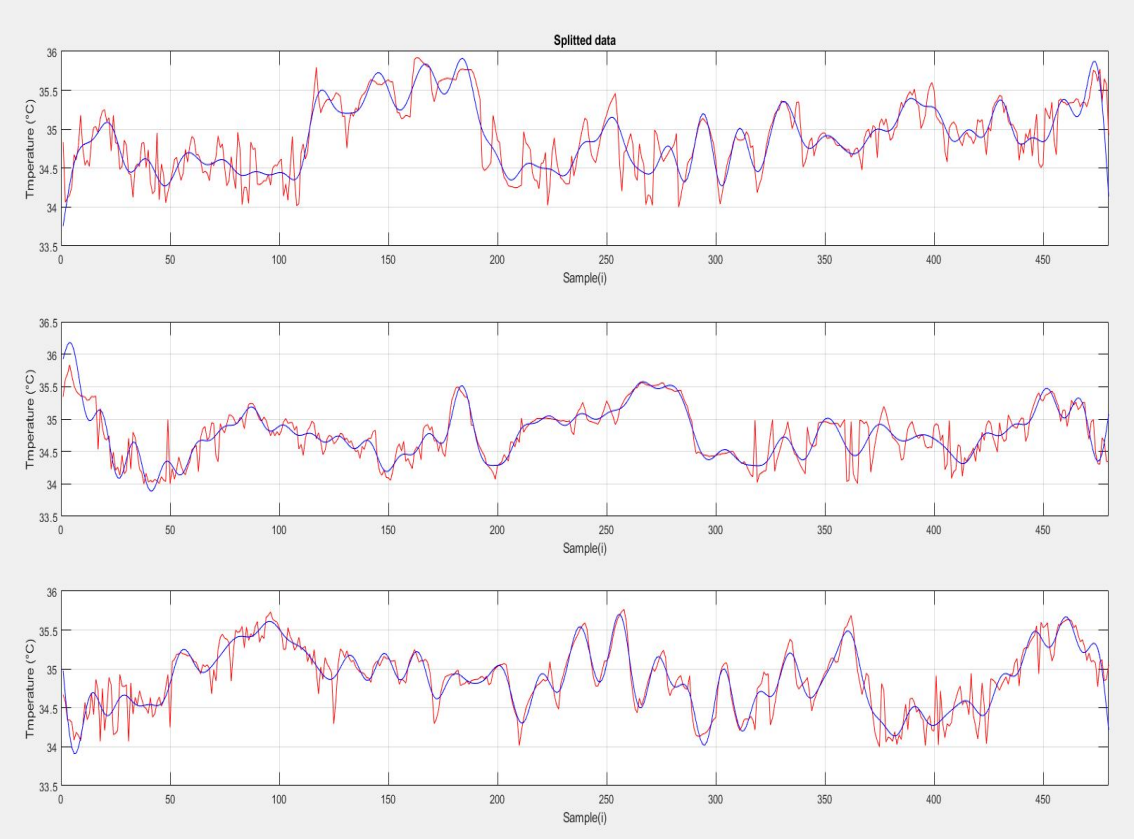

Figure 6.8: The 72-hour test splitted in 3 days. Starting from the top the first graph represents the first day while the last on is the last day.

With the obtained data it is possible to process them with different detection algorithm analysing, for example, the time or frequency domain signal.

As mentioned in section 2.5, detection is not the aim of this work and for this reason only a brief overview was performed. It is out of the scope of the dissertation and no detection algorithms has been developed or used.

In the accuracy test, the HR values obtained were poor and demonstrate a great error with respect the ground. This is certainly due to the chosen anatomic body location, the chest. The error is big because of technology limitations, due to low photodetector sensibility/light intensity.For these reasons, the sensory accuracy must be increased in

order to have useful data to process and use for detection.

CHAPTER 7

CONCLUSIONS

This Chapter will be used to discuss the results of this work. It contains possible future improvement for the prototype, including new features and some remarks regarding this work.

7.1 Future work

With the obtained results, discussed in chapter 6, it is possible at first to develop a machine learning algorithm to perform diseases detection.
For what regard the hardware, a deeper power consumptions analysis can be performed and better PPG signal accuracy is required.
The firmware can be improved and more performant algorithms can be used.
Moreover, It is possible to empower the desktop application, adding more features such as basic data analysis even if good data analysis can be performed by simply importing the data to another software such as MATLAB.
In the Institute of Systems and Robotics (ISR), were this work take place, a new researcher is working on the B-HOT prototype device, trying to power harvesting it. Power harvesting is the process of capture and conversion of a small amount of available energy in the environment into usable electrical energy.
If this experiment gives successful results, this opens the possibility of

making the device waterproof in order to remove the restriction related with taking a shower while performing the test.
Another interesting implementation that will be performed in the ISR regards stretchable electronics environment. The idea is to make a stretchable printed circuit board that will holds all the components. The power of such a PCB is that no case is necessary and that it can be a self-adhesive prototype device.

7.2 Remarks

Wearable health devices are rapidly increasing in scientific papers and in the markets. This work explores some of them and on the basis of that knowledge, a prototype device was designed and implemented.
The requirements for the device, written in chapter 4, were mostly achieved. In particular, the device is able to acquire accurate temperature measurement for the required amount of time, 72 hours.
As described in chapter 6, the subjects feel the device comfortable, thanks also to its compact format. The data are stored in the prototype device and are easy to be collected thanks to the desktop application and the docking station.
For what regard HR and SpO_2, the accuracy of the signal obtained from the chest was low and improvements on the signal quality or data processing is required. This can be performed increasing the average current of LEDs through software, but consequently the battery capability will be compromised and the 72 hours measuring period is no more guaranteed.
Another possibility is to change the sensor with an alternative one that has a more sensible photodiode but that still has low power capabilities.

A first proof of concept of a wearable device for health monitoring has been designed. It fulfils most of the requirements while some are not achieved due to technology limitations of the PPG sensor related with the anatomic body location chosen.

This work is composed of different libraries used to communicate with sensors and that allows communication through the device and the desktop application.

The documentation is organized to guide and simplify future developer, providing hardware and software files and encourage the beauty of discovery.

Appendices

APPENDIX A

PCB MATERIALS

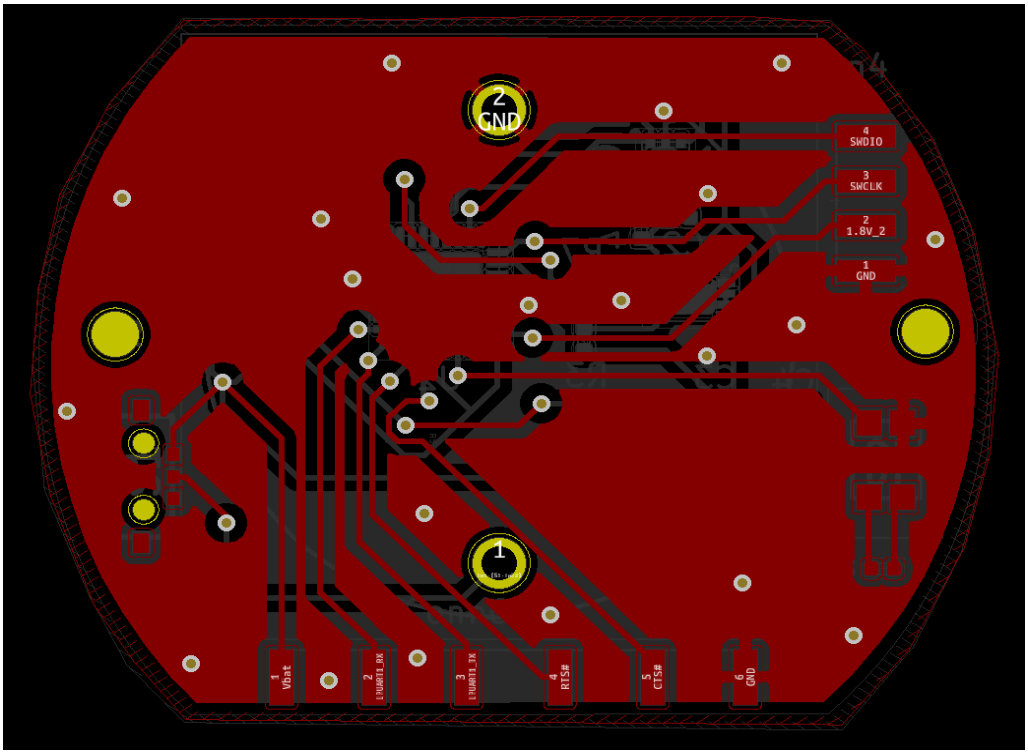

Figure A.1: PCB Top Layer. It includes tracks, holes, vias,pads and ground plane.

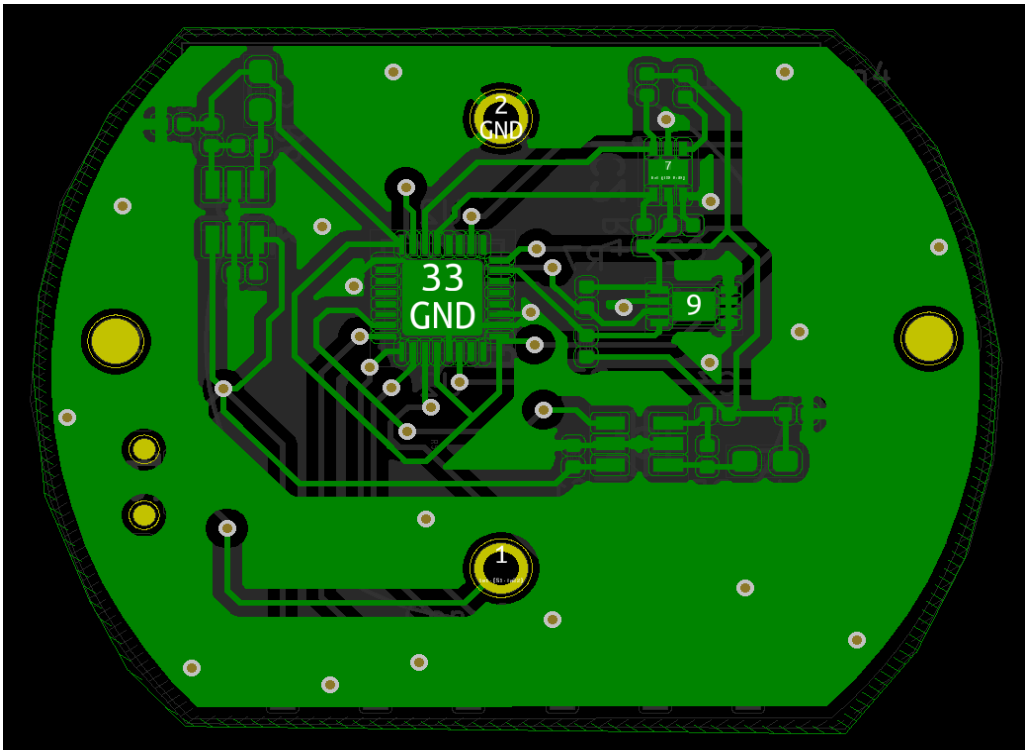

Figure A.2: PCB Bottom Layer. It includes tracks, holes, vias, pads and ground plane.

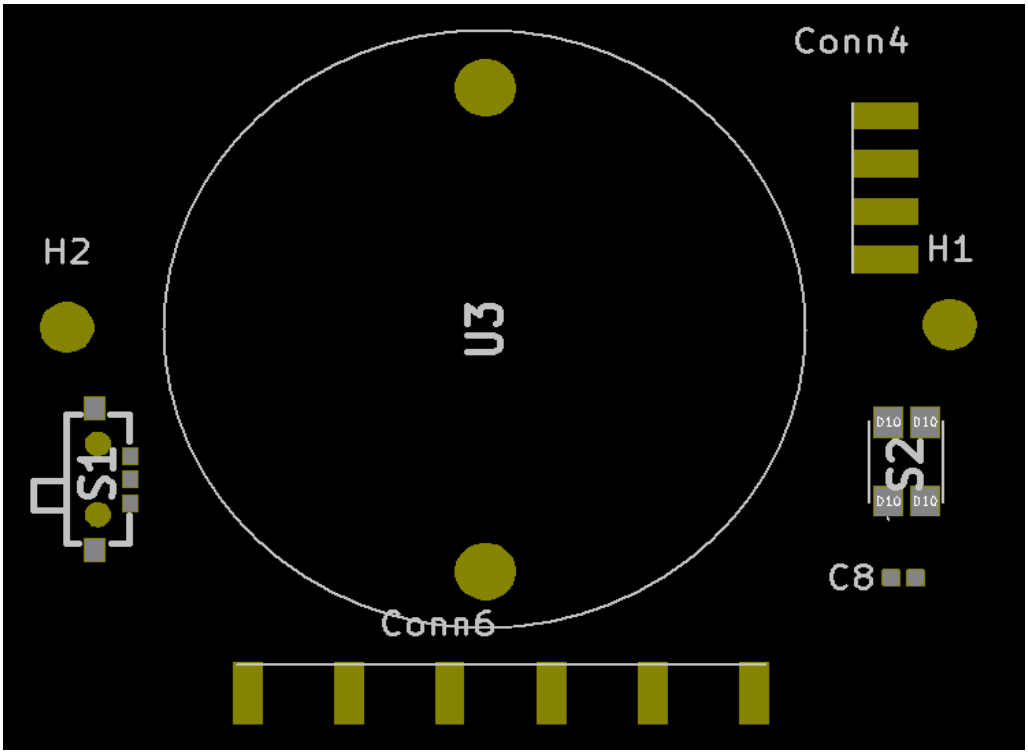

Figure A.3: PCB Top Layer - Gerber view

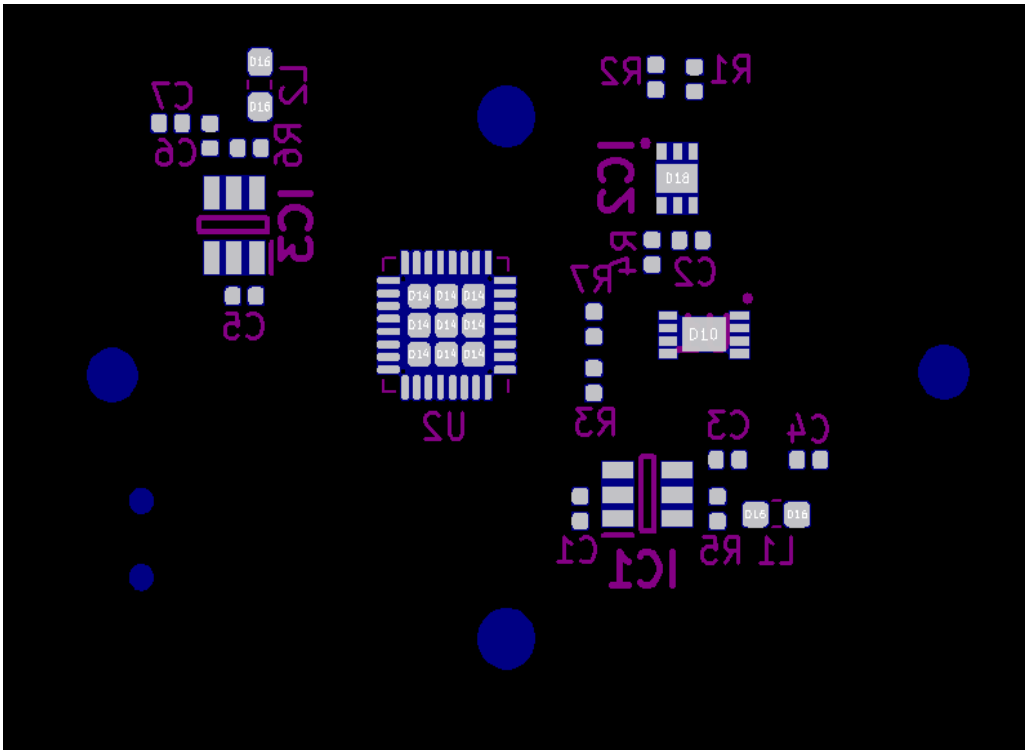

Figure A.4: PCB Bottom Layer - Gerber view

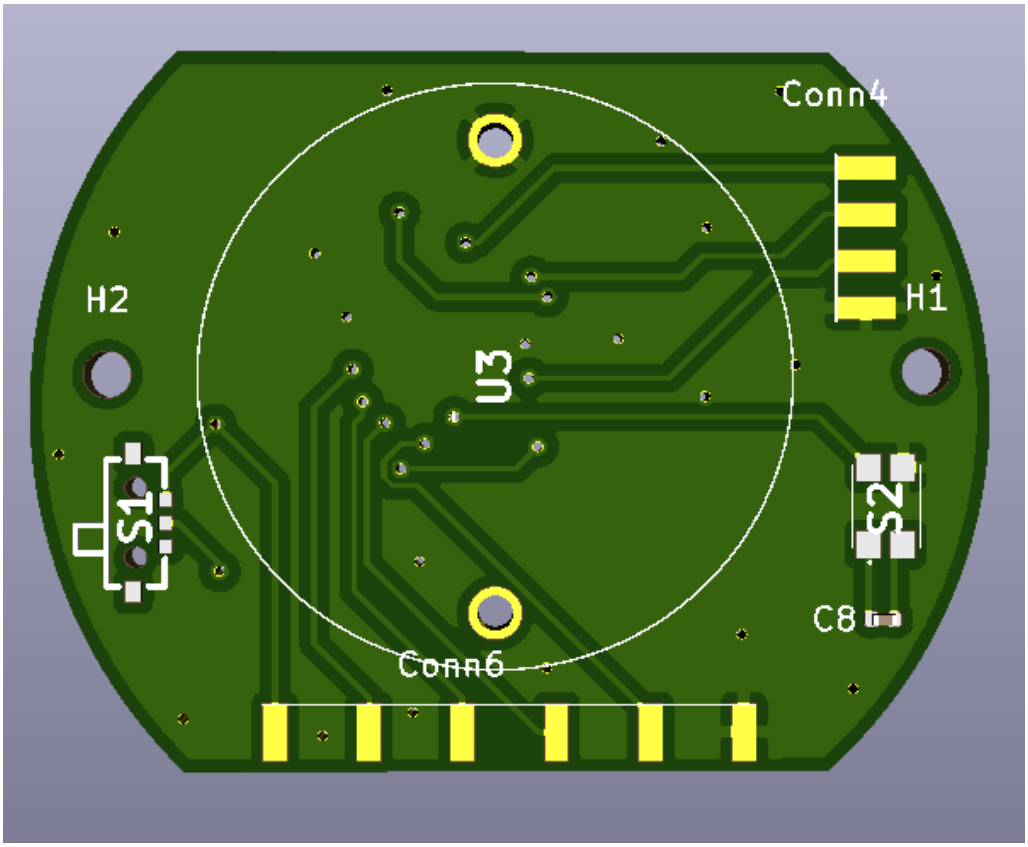

Figure A.5: PCB Top Layer-3D view

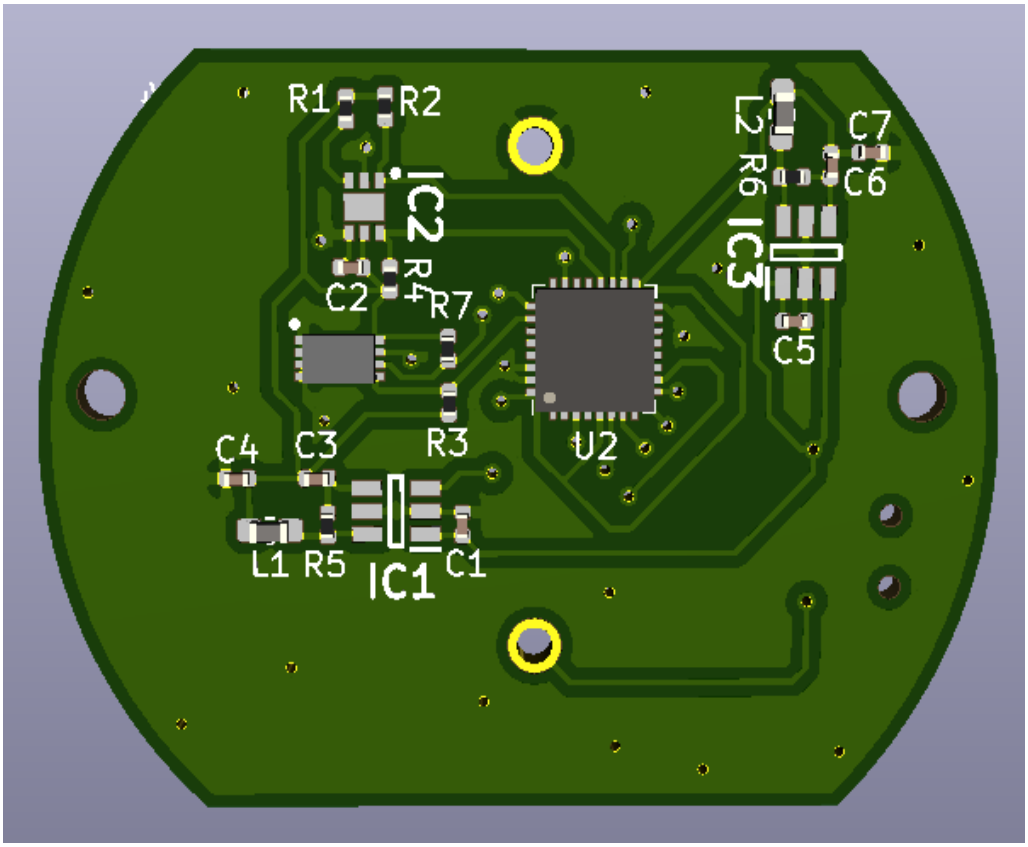

Figure A.6: PCB Bottom Layer-3D view

APPENDIX B

MCU SCRIPTS

B.1 Main

```
/* USER CODE BEGIN Header */
/**
******************************************************************************
* @file           : main.c
* @brief          : Main program body
******************************************************************************
* @attention
*
* <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
* All rights reserved.</center></h2>
*
* This software component is licensed by ST under BSD 3-Clause license,
* the "License"; You may not use this file except in compliance with the
* License. You may obtain a copy of the License at:
*                        opensource.org/licenses/BSD-3-Clause
*
******************************************************************************
*/
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "i2c.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
```

```
27
28  /* Private includes
     ----------------------------------------------------------------------*/
29  /* USER CODE BEGIN Includes */
30  #include "CAT24C128.h"
31  #include "tmp117.h"
32  #include "stdio.h"
33  #include "string.h"
34  /* USER CODE END Includes */
35
36  /* Private typedef
     -----------------------------------------------------------------------*/
37  /* USER CODE BEGIN PTD */
38
39  /* USER CODE END PTD */
40
41  /* Private define
     ------------------------------------------------------------------------*/
42  /* USER CODE BEGIN PD */
43  /* USER CODE END PD */
44
45  /* Private macro
     -------------------------------------------------------------------------*/
46  /* USER CODE BEGIN PM */
47
48  /* USER CODE END PM */
49
50  /* Private variables
     ---------------------------------------------------------------------*/
51
52  /* USER CODE BEGIN PV */
53
54  volatile int Flag_T;
55  volatile int Flag_W;
56  volatile uint8_t Page;
57
58  /* USER CODE END PV */
59
60  /* Private function prototypes
     -----------------------------------------------------------------*/
61  void SystemClock_Config(void);
62  /* USER CODE BEGIN PFP */
63
64  void WhatToDo(char * my_string);
65  void UART2_SendString(char * s);
```

```
66  int compare(char * astr);
67  void WriteToMemory(uint8_t * vector);
68
69  /* USER CODE END PFP */
70
71  /* Private user code
     ---------------------------------------------------------------------*/
72  /* USER CODE BEGIN 0 */
73
74  /* USER CODE END 0 */
75
76  /**
77  * @brief  The application entry point.
78  * @retval int
79  */
80  int main(void)
81  {
82  /* USER CODE BEGIN 1 */
83
84  Flag_T = 0;
85  Flag_W = 0;
86  Page = 0x00;
87
88  uint16_t Temp = 0;
89  uint8_t Vtemp[64];
90  int i = 0;
91  char input[8] = "READ";
92  float T;
93  char Buffer[25];
94
95  /* USER CODE END 1 */
96
97
98  /* MCU Configuration
     ---------------------------------------------------------------------*/
99
100 /* Reset of all peripherals, Initializes the Flash interface and the
     Systick. */
101 HAL_Init();
102
103 /* USER CODE BEGIN Init */
104
105 /* USER CODE END Init */
106
107 /* Configure the system clock */
```

```
108  SystemClock_Config();
109
110  /* USER CODE BEGIN SysInit */
111
112  /* USER CODE END SysInit */
113
114  /* Initialize all configured peripherals */
115  MX_GPIO_Init();
116  MX_I2C1_Init();
117  MX_I2C3_Init();
118  MX_TIM22_Init();
119  MX_USART2_UART_Init();
120  /* USER CODE BEGIN 2 */
121
122  HAL_TIM_Base_Start_IT(&htim22);
123  // Turn on DC-DC in order to read the first value of Temperature
124  HAL_GPIO_WritePin(ON_OFF_GPIO_Port,ON_OFF_Pin,GPIO_PIN_SET);
125  sprintf(Buffer, "Connected!\n\r");
126  UART2_SendString(Buffer);
127  /* USER CODE END 2 */
128
129  /* Infinite loop */
130  /* USER CODE BEGIN WHILE */
131  while (1)
132  {
133  HAL_UART_Receive(&huart2,(uint8_t*)input, strlen(input), 5000);
134  WhatToDo(input);
135  if ((Flag_T == 0)&&(Flag_W == 0))
136  {
137  HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_SET);
138  HAL_Delay(50);
139  TMP117_Initialization(hi2c1);
140  HAL_Delay(30);
141  Temp = TMP117_get_Temperature(hi2c1);
142  T = (float)Temp * 0.0078125;
143  while(T < 10 || T > 45 )
144  {
145  Temp = TMP117_get_Temperature(hi2c1);
146  T = (float)Temp * 0.0078125;
147
148  }
149  Vtemp[i] = (Temp>>8); // high
150  Vtemp[i+1] = (Temp&0x00ff); // low;
151  // Turn-off DC-DC
152  HAL_GPIO_WritePin(ON_OFF_GPIO_Port,ON_OFF_Pin,GPIO_PIN_RESET);
```

```
153   i = i+2;
154   Flag_T = 1;
155   sprintf(Buffer,"Reading sample %d\r\n",(i/2 + (int)Page*32));
156   UART2_SendString(Buffer);
157   if (i == 64)
158   {
159   WriteToMemory(Vtemp);
160   i = 0;
161   }
162   }
163   else if(( Flag_W == 1)&&(Flag_T == 0))
164   {
165   sprintf(Buffer,"Sample collection complete\r\n");
166   UART2_SendString(Buffer);
167   }
168
169
170   /* USER CODE END WHILE */
171
172   /* USER CODE BEGIN 3 */
173   }
174   /* USER CODE END 3 */
175   }
176
177   /**
178   * @brief System Clock Configuration
179   * @retval None
180   */
181   void SystemClock_Config(void)
182   {
183   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
184   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
185   RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
186
187   /** Configure the main internal regulator output voltage
188   */
189   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
190   /** Initializes the CPU, AHB and APB busses clocks
191   */
192   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
193   RCC_OscInitStruct.MSIState = RCC_MSI_ON;
194   RCC_OscInitStruct.MSICalibrationValue = 0;
195   RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_1;
196   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
197   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
```

```
198   {
199   Error_Handler();
200   }
201   /** Initializes the CPU, AHB and APB busses clocks
202   */
203   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
204   |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
205   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
206   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
207   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
208   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
209
210   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) !=
       HAL_OK)
211   {
212   Error_Handler();
213   }
214   PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2|
       RCC_PERIPHCLK_I2C1
215   |RCC_PERIPHCLK_I2C3;
216   PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
217   PeriphClkInit.I2c1ClockSelection = RCC_I2C1CLKSOURCE_PCLK1;
218   PeriphClkInit.I2c3ClockSelection = RCC_I2C3CLKSOURCE_PCLK1;
219   if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
220   {
221   Error_Handler();
222   }
223   }
224
225   /* USER CODE BEGIN 4 */
226
227   void WhatToDo(char * my_string)
228   {
229   int val;
230   uint8_t num;
231   uint8_t Data[64];
232   char Buffer[10];
233   float TempCels = 0;
234   //   my_string = "READ";
235   val = compare(my_string);
236   switch(val)
237   {
238   case 1:
239
240   break;
```

```
241
242 case 10:   //READ
243 num = 0x00;
244 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_SET);
245 HAL_Delay(50);
246 while(num <0x2D)
247 {
248 cat24_HAL_ReadPage(&hi2c3,num,Data,64);
249
250 for(int i=0;i<64;i=i+2)
251 {
252 TempCels = ((float)(((Data[i]<<8)|Data[i+1]))*0.0078125);
253 sprintf(Buffer,"%f\r\n", TempCels);
254 UART2_SendString(Buffer);
255 }
256 num = num + 0x01;
257 }
258 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_RESET);
259 break;
260
261 case 100: //DELETE
262 num = 0x00;
263 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_SET);
264 HAL_Delay(50);
265 while( (num < 0x2D) )
266 {
267 cat24_HAL_ErasePage(&hi2c3, num);
268 num = num + 0x01;
269 }
270 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_RESET);
271 break;
272
273 default:
274
275 break;
276
277 }
278 if (Page == 0x2D)
279 {
280 Flag_W = 1;
281 }
282
283
284 }
285
```

```
286 void WriteToMemory(uint8_t * vector)
287 {
288 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_SET);
289 HAL_Delay(50);
290 cat24_HAL_WritePage(&hi2c3,Page,vector,64);
291 Page = Page + 0x01;
292 HAL_GPIO_WritePin(GPIOA,ON_OFF_Pin,GPIO_PIN_RESET);
293 }
294
295 void UART2_SendString(char * s)
296 {
297 HAL_UART_Transmit(&huart2, (uint8_t*)s,strlen(s), 100);
298 }
299
300 int compare(char * astr)
301 {
302 char copy[8]="";
303 int first = 0;
304 if (strcmp(astr,"DELETE") == 0)
305 {
306 first = 100;
307 for(int i=0;i<8; i++)
308 {
309 astr[i] = copy[i];
310 }
311 }
312 if (strcmp(astr,"READ") == 0)
313 {
314 first = 10;
315 for(int i=0;i<8; i++)
316 {
317 astr[i] = copy[i];
318 }
319 }
320 else
321 first = 1;
322
323 return first;
324 }
325 /* USER CODE END 4 */
326
327 /**
328 * @brief  This function is executed in case of error occurrence.
329 * @retval None
330 */
```

```
331 void Error_Handler(void)
332 {
333 /* USER CODE BEGIN Error_Handler_Debug */
334 /* User can add his own implementation to report the HAL error return
      state */
335
336 /* USER CODE END Error_Handler_Debug */
337 }
338
339 #ifdef  USE_FULL_ASSERT
340 /**
341 * @brief  Reports the name of the source file and the source line
     number
342 *         where the assert_param error has occurred.
343 * @param  file: pointer to the source file name
344 * @param  line: assert_param error line source number
345 * @retval None
346 */
347 void assert_failed(uint8_t *file, uint32_t line)
348 {
349 /* USER CODE BEGIN 6 */
350 /* User can add his own implementation to report the file name and
     line number,
351 tex: printf("Wrong parameters value: file %s on line %d\r\n", file,
     line) */
352 /* USER CODE END 6 */
353 }
354 #endif /* USE_FULL_ASSERT */
355
356 /************************END OF FILE************************/
357
```

Listing B.1: MCU main loop algorithm

B.2 MAX30102 Manager

B.2.1 MAX30102.c

```
1   /** \file max30102.cpp
    ******************************************************
2   *
3   * Project: MAXREFDES117#
4   * Filename: max30102.cpp
5   * Description: This module is an embedded controller driver for the
    MAX30102
```

```
 *
 *
 *
-------------------------------------------------------------------------


 *
 * This code follows the following naming conventions:
 *
 * char              ch_pmod_value
 * char (array)      s_pmod_s_string[16]
 * float             f_pmod_value
 * int32_t           n_pmod_value
 * int32_t (array)   an_pmod_value[16]
 * int16_t           w_pmod_value
 * int16_t (array)   aw_pmod_value[16]
 * uint16_t          uw_pmod_value
 * uint16_t (array)  auw_pmod_value[16]
 * uint8_t           uch_pmod_value
 * uint8_t (array)   auch_pmod_buffer[16]
 * uint32_t          un_pmod_value
 * int32_t *         pn_pmod_value
 *
 *
-------------------------------------------------------------------------
 */
/*
*****************************************************************************

 * Copyright (C) 2016 Maxim Integrated Products, Inc., All Rights
Reserved.
 *
 * Permission is hereby granted, free of charge, to any person
obtaining a
 * copy of this software and associated documentation files (the "
Software"),
 * to deal in the Software without restriction, including without
limitation
 * the rights to use, copy, modify, merge, publish, distribute,
sublicense,
 * and/or sell copies of the Software, and to permit persons to whom
 the
 * Software is furnished to do so, subject to the following
conditions:
 *
 * The above copyright notice and this permission notice shall be
```

```
included
38  * in all copies or substantial portions of the Software.
39  *
40  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS
41  * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
42  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT.
43  * IN NO EVENT SHALL MAXIM INTEGRATED BE LIABLE FOR ANY CLAIM,
    DAMAGES
44  * OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
    OTHERWISE,
45  * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
    USE OR
46  * OTHER DEALINGS IN THE SOFTWARE.
47  *
48  * Except as contained in this notice, the name of Maxim Integrated
49  * Products, Inc. shall not be used except as stated in the Maxim
    Integrated
50  * Products, Inc. Branding Policy.
51  *
52  * The mere transfer of this software does not imply any licenses
53  * of trade secrets, proprietary technology, copyrights, patents,
54  * trademarks, maskwork rights, or any other form of intellectual
55  * property whatsoever. Maxim Integrated Products, Inc. retains all
56  * ownership rights.
57

    ********************************************************************************

58  *
59  * Modified original MAXIM source code on: 13.11.2019
60  * Author: Detjon Brahimaj
61  * Code is modified to work with STM32 HAL libraries.
62  *
63  *
64  */
65 #include "main.h"
66 #include "i2c.h"
67 #include "usart.h"
68 #include "string.h"
69 #include "stdio.h"
70
71 #include "MAX30102.h"
72 #include "algorithm.h"
73
```

```
74  #define I2C_TIMEOUT 150
75
76  I2C_HandleTypeDef *i2c_max30102;
77
78  volatile uint32_t IrBuffer[MAX30102_BUFFER_LENGTH+27]; //IR LED
    sensor data
79  volatile uint32_t RedBuffer[MAX30102_BUFFER_LENGTH+27];     //Red
    LED sensor data
80  volatile uint32_t IrBuffer1[MAX30102_BUFFER_LENGTH]; //IR LED
    sensor data
81  volatile uint32_t RedBuffer1[MAX30102_BUFFER_LENGTH];     //Red LED
    sensor data
82  volatile uint32_t CollectedSamples;
83  volatile uint8_t IsFingerOnScreen;
84  int32_t Sp02Value;
85  int8_t Sp02IsValid;
86  int32_t HeartRate;
87  int8_t IsHrValid;
88  uint8_t uch_dummy;
89
90  typedef enum
91  {
92  MAX30102_STATE_BEGIN,
93  MAX30102_STATE_CALIBRATE,
94  MAX30102_STATE_CALCULATE_HR,
95  MAX30102_STATE_COLLECT_NEXT_PORTION
96  }MAX30102_STATE;
97
98  MAX30102_STATE StateMachine;
99
100
101 void Max30102_Calc(I2C_HandleTypeDef *i2c)
102 {
103 int i = 0;
104 uint8_t puch;
105
106
107 Max30102_Reset() ; // Reset
108 Max30102_Init(i2c); // Initialization
109
110
111 int n_ir_buffer_length=500; //buffer length of 50 stores 10 seconds
    of samples running at 50sps
112
113 //read 500 samples
```

```
114     Max30102_Led1PulseAmplitude(MAX30102_RED_LED_CURRENT_HIGH);
115     Max30102_Led2PulseAmplitude(MAX30102_IR_LED_CURRENT_HIGH);
116     while(i < (n_ir_buffer_length+27))
117     {
118
119     while(HAL_GPIO_ReadPin(GPIOA,INT_PPG_Pin) == 1)
120     {
121     //wait until the interrupt pin asserts
122     }
123     for(uint8_t k = 0; k < MAX30102_FIFO_ALMOST_FULL_SAMPLES; k++)
124     {
125     Max30102_ReadFifo((RedBuffer+i+k), (IrBuffer+i+k)); //read from
    MAX30102 FIFO
126     }
127     Max30102_ReadReg(REG_INTR_STATUS_1,&puch);
128
129     i = i +17;
130
131     }
132
133
134     Max30102_Led1PulseAmplitude(MAX30102_IR_LED_CURRENT_LOW);
135     Max30102_Led2PulseAmplitude(MAX30102_RED_LED_CURRENT_LOW);
136
137     for(uint16_t k = 0; k < MAX30102_BUFFER_LENGTH; k++)
138     {
139     IrBuffer1[k] = IrBuffer[k+27];
140     RedBuffer1[k] = RedBuffer[k + 27];
141     }
142     maxim_heart_rate_and_oxygen_saturation(IrBuffer1, RedBuffer1,
    MAX30102_BUFFER_LENGTH-1, MAX30102_BUFFER_LENGTH-2, &Sp02Value, &
    Sp02IsValid, &HeartRate, &IsHrValid);
143
144
145
146     }
147
148
149
150     MAX30102_STATUS Max30102_WriteReg(uint8_t uch_addr, uint8_t
    uch_data)
151     {
152     if(HAL_I2C_Mem_Write(i2c_max30102, MAX30102_ADDRESS, uch_addr, 1, &
    uch_data, 1, I2C_TIMEOUT) == HAL_OK)
153     return MAX30102_OK;
```

```
154     return MAX30102_ERROR;
155     }
156 
157     MAX30102_STATUS Max30102_ReadReg(uint8_t uch_addr, uint8_t *
        puch_data)
158     {
159     if(HAL_I2C_Mem_Read(i2c_max30102, MAX30102_ADDRESS, uch_addr, 1,
        puch_data, 1, I2C_TIMEOUT) == HAL_OK)
160     return MAX30102_OK;
161     return MAX30102_ERROR;
162     }
163 
164     MAX30102_STATUS Max30102_WriteRegisterBit(uint8_t Register, uint8_t
         Bit, uint8_t Value)
165     {
166     uint8_t tmp;
167     if(MAX30102_OK != Max30102_ReadReg(Register, &tmp))
168     return MAX30102_ERROR;
169     tmp &= ~(1<<Bit);
170     tmp |= (Value&0x01)<<Bit;
171     if(MAX30102_OK != Max30102_WriteReg(Register, tmp))
172     return MAX30102_ERROR;
173 
174     return MAX30102_OK;
175     }
176 
177     MAX30102_STATUS Max30102_ReadFifo(volatile uint32_t *pun_red_led,
        volatile uint32_t *pun_ir_led)
178     {
179     uint32_t un_temp;
180     *pun_red_led=0;
181     *pun_ir_led=0;
182     uint8_t ach_i2c_data[6];
183 
184     if(HAL_I2C_Mem_Read(i2c_max30102, MAX30102_ADDRESS, REG_FIFO_DATA,
        1, ach_i2c_data, 6, I2C_TIMEOUT) != HAL_OK)
185     {
186     return MAX30102_ERROR;
187     }
188     un_temp=(unsigned char) ach_i2c_data[0];
189     un_temp<<=16;
190     *pun_red_led+=un_temp;
191     un_temp=(unsigned char) ach_i2c_data[1];
192     un_temp<<=8;
193     *pun_red_led+=un_temp;
```

```
194     un_temp=(unsigned char) ach_i2c_data[2];
195     *pun_red_led+=un_temp;
196
197     un_temp=(unsigned char) ach_i2c_data[3];
198     un_temp<<=16;
199     *pun_ir_led+=un_temp;
200     un_temp=(unsigned char) ach_i2c_data[4];
201     un_temp<<=8;
202     *pun_ir_led+=un_temp;
203     un_temp=(unsigned char) ach_i2c_data[5];
204     *pun_ir_led+=un_temp;
205     *pun_red_led&=0x03FFFF;  //Mask MSB [23:18]
206     *pun_ir_led&=0x03FFFF;  //Mask MSB [23:18]
207
208     return MAX30102_OK;
209     }
210
211
212     //
213     //  LEDs Pulse Amplitute Configuration
214     //  LED Current = Value * 0.2 mA
215     //
216     MAX30102_STATUS Max30102_Led1PulseAmplitude(uint8_t Value)
217     {
218     if(MAX30102_OK != Max30102_WriteReg(REG_LED1_PA, Value))
219     return MAX30102_ERROR;
220     return MAX30102_OK;
221     }
222
223     MAX30102_STATUS Max30102_Led2PulseAmplitude(uint8_t Value)
224     {
225     if(MAX30102_OK != Max30102_WriteReg(REG_LED2_PA, Value))
226     return MAX30102_ERROR;
227     return MAX30102_OK;
228     }
229
230     // Get values
231
232     uint8_t Max30102_GetHeartRate(void)
233     {
234     return (uint8_t)HeartRate;
235     }
236
237     uint8_t Max30102_GetSpO2Value(void)
238     {
```

```
239     return (uint8_t)Sp02Value;
240     }
241 
242     // Reset
243     MAX30102_STATUS Max30102_Reset(void)
244     {
245     uint8_t uch_dummy;
246     Max30102_ReadReg(0x00,&uch_dummy);
247     Max30102_ReadReg(0x01,&uch_dummy);
248     if(MAX30102_OK != Max30102_WriteReg(REG_MODE_CONFIG,0x40))
249     return MAX30102_ERROR;
250     else
251     return MAX30102_OK;
252     }
253 
254     //  Initialization
255     MAX30102_STATUS Max30102_Init(I2C_HandleTypeDef *i2c)
256     {
257     uint8_t uch_dummy;
258     i2c_max30102 = i2c;
259 
260     Max30102_ReadReg(0,&uch_dummy);
261     Max30102_ReadReg(0x00,&uch_dummy);
262     Max30102_ReadReg(0x01,&uch_dummy);
263     if(MAX30102_OK != Max30102_WriteReg(REG_INTR_ENABLE_1,0x80))// INTR
      setting
264     return MAX30102_ERROR;
265     if(MAX30102_OK != Max30102_WriteReg(REG_INTR_ENABLE_2,0x00))
266     return MAX30102_ERROR;
267     if(MAX30102_OK != Max30102_WriteReg(REG_FIFO_WR_PTR,0x00))//
    FIFO_WR_PTR[4:0]
268     return MAX30102_ERROR;
269     if(MAX30102_OK != Max30102_WriteReg(REG_OVF_COUNTER,0x00))//
    OVF_COUNTER[4:0]
270     return MAX30102_ERROR;
271     if(MAX30102_OK != Max30102_WriteReg(REG_FIFO_RD_PTR,0x00))//
    FIFO_RD_PTR[4:0]
272     return MAX30102_ERROR;
273     if(MAX30102_OK != Max30102_WriteReg(REG_FIFO_CONFIG,0x0f))//sample
    avg = 1, fifo rollover=false, fifo almost full = 17
274     return MAX30102_ERROR;
275     if(MAX30102_OK != Max30102_WriteReg(REG_MODE_CONFIG,0x03))//0x02
    for Red only, 0x03 for SpO2 mode 0x07 multimode LED
276     return MAX30102_ERROR;
277     if(MAX30102_OK != Max30102_WriteReg(REG_SPO2_CONFIG,0x61))//
```

```
    SPO2_ADC range = 16384nA, SPO2 sample rate (50 Hz), LED pulseWidth
    (118uS)
278  return MAX30102_ERROR;
279  if(MAX30102_OK != Max30102_WriteReg(REG_LED1_PA,0x18)) //  //Choose
    value for ~ 4.8 mA for LED1
280  return MAX30102_ERROR;
281  if(MAX30102_OK != Max30102_WriteReg(REG_LED2_PA,0x18)) //  //Choose
    value for ~ 4.8 mA for LED2
282  return MAX30102_ERROR;
283
284
285  return MAX30102_OK;
286  }
287
```

Listing B.2: MCU-MAX30102 sensor manager

B.2.2 Algorithm.c

```
1   /** \file algorithm.cpp
    ******************************************************************
2   *
3   * Project: MAXREFDES117#
4   * Filename: algorithm.cpp
5   * Description: This module calculates the heart rate/SpO2 level
6   *
7   *
8   * --------------------------------------------------------------------
9   *
10  * This code follows the following naming conventions:
11  *
12  * char             ch_pmod_value
13  * char (array)     s_pmod_s_string[16]
14  * float            f_pmod_value
15  * int32_t          n_pmod_value
16  * int32_t (array)  an_pmod_value[16]
17  * int16_t          w_pmod_value
18  * int16_t (array)  aw_pmod_value[16]
19  * uint16_t         uw_pmod_value
20  * uint16_t (array) auw_pmod_value[16]
21  * uint8_t          uch_pmod_value
22  * uint8_t (array)  auch_pmod_buffer[16]
23  * uint32_t         un_pmod_value
24  * int32_t *        pn_pmod_value
25  *
26  * -------------------------------------------------------------------- */
```

```
27  /*******************************************************************
28   * Copyright (C) 2016 Maxim Integrated Products, Inc., All Rights
    Reserved.
29   *
30   * Permission is hereby granted, free of charge, to any person
    obtaining a
31   * copy of this software and associated documentation files (the "
    Software"),
32   * to deal in the Software without restriction, including without
    limitation
33   * the rights to use, copy, modify, merge, publish, distribute,
    sublicense,
34   * and/or sell copies of the Software, and to permit persons to whom
     the
35   * Software is furnished to do so, subject to the following
    conditions:
36   *
37   * The above copyright notice and this permission notice shall be
    included
38   * in all copies or substantial portions of the Software.
39   *
40   * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS
41   * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
42   * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT.
43   * IN NO EVENT SHALL MAXIM INTEGRATED BE LIABLE FOR ANY CLAIM,
    DAMAGES
44   * OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
    OTHERWISE,
45   * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
    USE OR
46   * OTHER DEALINGS IN THE SOFTWARE.
47   *
48   * Except as contained in this notice, the name of Maxim Integrated
49   * Products, Inc. shall not be used except as stated in the Maxim
    Integrated
50   * Products, Inc. Branding Policy.
51   *
52   * The mere transfer of this software does not imply any licenses
53   * of trade secrets, proprietary technology, copyrights, patents,
54   * trademarks, maskwork rights, or any other form of intellectual
55   * property whatsoever. Maxim Integrated Products, Inc. retains all
56   * ownership rights.
57   *******************************************************************
```

```
58  *
59  * Modified original MAXIM source code on: 13.11.2019
60  * Author: Detjon Brahimaj
61  * Code is modified to work with STM32 HAL libraries.
62  *
63  *
64  */
65 #include "main.h"
66 #include <algorithm.h>
67
68 const uint16_t auw_hamm[31]={ 41,    276,    512,    276,    41 };
    //Hamm=  long16(512* hamming(5)');
69 //uch_spo2_table is computed as  -45.060*ratioAverage* ratioAverage
    + 30.354 *ratioAverage + 94.845 ;
70 const uint8_t uch_spo2_table[184]={ 95, 95, 95, 96, 96, 96, 97, 97,
    97, 97, 97, 98, 98, 98, 98, 98, 99, 99, 99, 99,
71 99, 99, 99, 99, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
   100, 100, 100, 100, 100, 100,
72 100, 100, 100, 100, 99, 99, 99, 99, 99, 99, 99, 99, 98, 98, 98, 98,
    98, 98, 97, 97,
73 97, 97, 96, 96, 96, 96, 95, 95, 95, 94, 94, 94, 93, 93, 93, 92, 92,
    92, 91, 91,
74 90, 90, 89, 89, 89, 88, 88, 87, 87, 86, 86, 85, 85, 84, 84, 83, 82,
    82, 81, 81,
75 80, 80, 79, 78, 78, 77, 76, 76, 75, 74, 74, 73, 72, 72, 71, 70, 69,
    69, 68, 67,
76 66, 66, 65, 64, 63, 62, 62, 61, 60, 59, 58, 57, 56, 56, 55, 54, 53,
    52, 51, 50,
77 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33,
    31, 30, 29,
78 28, 27, 26, 25, 23, 22, 21, 20, 19, 17, 16, 15, 14, 12, 11, 10, 9,
   7, 6, 5,
79 3, 2, 1 } ;
80 static int32_t an_dx[ BUFFER_SIZE-MA4_SIZE]; // delta
81 static int32_t an_x[ BUFFER_SIZE]; //ir
82 static int32_t an_y[ BUFFER_SIZE]; //red
83
84
85 void maxim_heart_rate_and_oxygen_saturation(volatile uint32_t *
   pun_ir_buffer, volatile uint32_t *pun_red_buffer, int32_t
   n_buffer_length, uint16_t un_offset, int32_t *pn_spo2, int8_t *
   pch_spo2_valid, int32_t *pn_heart_rate, int8_t  *pch_hr_valid)
86 /**
87 * \brief        Calculate the heart rate and SpO2 level
88 * \par          Details
```

```
89   *               By detecting  peaks of PPG cycle and corresponding
     AC/DC of red/infra-red signal, the ratio for the SPO2 is computed.
90   *               Since this algorithm is aiming for Arm M0/M3.
     formaula for SPO2 did not achieve the accuracy due to register
     overflow.
91   *               Thus, accurate SPO2 is precalculated and save longo
      uch_spo2_table[] per each ratio.
92   *
93   * \param[in]    *pun_ir_buffer           - IR sensor data buffer
94   * \param[in]    n_buffer_length      - IR sensor data buffer length
95   * \param[in]    *pun_red_buffer          - Red sensor data buffer
96   * \param[out]    *pn_spo2                - Calculated SpO2 value
97   * \param[out]    *pch_spo2_valid         - 1 if the calculated SpO2
      value is valid
98   * \param[out]    *pn_heart_rate          - Calculated heart rate
     value
99   * \param[out]    *pch_hr_valid           - 1 if the calculated
     heart rate value is valid
100  *
101  * \retval       None
102  */
103  {
104  uint32_t un_ir_mean ,un_only_once ;
105  int32_t k ,n_i_ratio_count;
106  int32_t i, s, m, n_exact_ir_valley_locs_count ,n_middle_idx;
107  int32_t n_th1, n_npks,n_c_min;
108  int32_t an_ir_valley_locs[15] ;
109  int32_t an_exact_ir_valley_locs[15] ;
110  int32_t an_dx_peak_locs[15] ;
111  int32_t n_peak_interval_sum;
112  int32_t n_y_ac, n_x_ac;
113  int32_t n_spo2_calc;
114  int32_t n_y_dc_max, n_x_dc_max;
115  int32_t n_y_dc_max_idx, n_x_dc_max_idx;
116  int32_t an_ratio[5],n_ratio_average;
117  int32_t n_nume,  n_denom ;
118  // remove DC of ir signal
119  un_ir_mean =0;
120  for (k=0 ; k<n_buffer_length ; k++ )
121  {
122  un_ir_mean += pun_ir_buffer[k];
123  }
124  un_ir_mean =un_ir_mean/n_buffer_length ;
125  for (k=0 ; k<n_buffer_length ; k++ )
126  {
```

```
127     an_x[k] =  pun_ir_buffer[k] - un_ir_mean;
128     }
129     // 4 pt Moving Average
130     for(k=0; k< BUFFER_SIZE-MA4_SIZE; k++)
131     {
132     n_denom= ( an_x[k]+an_x[k+1]+ an_x[k+2]+ an_x[k+3]);
133     an_x[k]=  n_denom/(int32_t)4;
134     }
135     // get difference of smoothed IR signal
136     for( k=0; k<BUFFER_SIZE-MA4_SIZE-1;  k++)
137     an_dx[k]= (an_x[k+1]- an_x[k]);
138
139     // 2-pt Moving Average to an_dx
140     for(k=0; k< BUFFER_SIZE-MA4_SIZE-2; k++){
141     an_dx[k] =  ( an_dx[k]+an_dx[k+1])/2 ;
142     }
143     // hamming window
144     // flip wave form so that we can detect valley with peak detector
145     for ( i=0 ; i<BUFFER_SIZE-HAMMING_SIZE-MA4_SIZE-2 ;i++){
146     s= 0;
147     for( k=i; k<i+ HAMMING_SIZE ;k++){
148     s -= an_dx[k] *auw_hamm[k-i] ;
149     }
150     an_dx[i]= s/ (int32_t)1146; // divide by sum of auw_hamm
151     }
152     n_th1=0; // threshold calculation
153     for ( k=0 ; k<BUFFER_SIZE-HAMMING_SIZE ;k++){
154     n_th1 += ((an_dx[k]>0)? an_dx[k] : ((int32_t)0-an_dx[k])) ;
155     }
156     n_th1= n_th1/ ( BUFFER_SIZE-HAMMING_SIZE);
157     // peak location is acutally index for sharpest location of raw
      signal since we flipped the signal
158     maxim_find_peaks( an_dx_peak_locs, &n_npks, an_dx, BUFFER_SIZE-
      HAMMING_SIZE, n_th1, 8, 5 );//peak_height, peak_distance,
      max_num_peaks
159     n_peak_interval_sum =0;
160     if (n_npks>=2){
161     for (k=1; k<n_npks; k++)
162     n_peak_interval_sum += (an_dx_peak_locs[k]-an_dx_peak_locs[k -1]);
163     n_peak_interval_sum=n_peak_interval_sum/(n_npks-1);
164     *pn_heart_rate=(int32_t)(2500/(float)n_peak_interval_sum*(float)(FS
      /100.0));// beats per minutes
165     *pch_hr_valid  = 1;
166     }
167     else  {
```

```
168     *pn_heart_rate = -1;
169     *pch_hr_valid  = 0;
170     }
171     for ( k=0 ; k<n_npks ;k++)
172     an_ir_valley_locs[k]=an_dx_peak_locs[k]+HAMMING_SIZE/2;
173     // raw value : RED(=y) and IR(=X)
174     // we need to assess DC and AC value of ir and red PPG.
175     for (k=0 ; k<n_buffer_length ; k++ )
176     {
177     an_x[k] =  pun_ir_buffer[k];
178     an_y[k] =  pun_red_buffer[k];
179     }
180     // find precise min near an_ir_valley_locs
181     n_exact_ir_valley_locs_count =0;
182     for(k=0 ; k<n_npks ;k++){
183     un_only_once =1;
184     m=an_ir_valley_locs[k];
185     n_c_min= 16777216;//2^24;
186     if (m+5 <  BUFFER_SIZE-HAMMING_SIZE  && m-5 >0){
187     for(i= m-5;i<m+5; i++)
188     if (an_x[i]<n_c_min){
189     if (un_only_once >0){
190     un_only_once =0;
191     }
192     n_c_min= an_x[i] ;
193     an_exact_ir_valley_locs[k]=i;
194     }
195     if (un_only_once ==0)
196     n_exact_ir_valley_locs_count ++ ;
197     }
198     }
199     if (n_exact_ir_valley_locs_count <2 ){
200     *pn_spo2 =  -1 ; // do not use SPO2 since signal ratio is out of
      range
201     *pch_spo2_valid  = 0;
202     return;
203     }
204     // 4 pt MA
205     for(k=0; k< BUFFER_SIZE-MA4_SIZE; k++){
206     an_x[k]=( an_x[k]+an_x[k+1]+ an_x[k+2]+ an_x[k+3])/(int32_t)4;
207     an_y[k]=( an_y[k]+an_y[k+1]+ an_y[k+2]+ an_y[k+3])/(int32_t)4;
208     }
209     //using an_exact_ir_valley_locs , find ir-red DC andir-red AC for
      SPO2 calibration ratio
210     //finding AC/DC maximum of raw ir * red between two valley
```

```
    locations
211  n_ratio_average =0;
212  n_i_ratio_count =0;
213
214  for(k=0; k< 5; k++) an_ratio[k]=0;
215  for (k=0; k< n_exact_ir_valley_locs_count; k++){
216  if (an_exact_ir_valley_locs[k] > BUFFER_SIZE ){
217  *pn_spo2 =  -1 ; // do not use SPO2 since valley loc is out of
    range
218  *pch_spo2_valid  = 0;
219  return;
220  }
221  }
222  // find max between two valley locations
223  // and use ratio betwen AC compoent of Ir & Red and DC compoent of
    Ir & Red for SPO2
224  for (k=0; k< n_exact_ir_valley_locs_count-1; k++){
225  n_y_dc_max= - 16777216 ;
226  n_x_dc_max= - 16777216;
227  if (an_exact_ir_valley_locs[k+1]-an_exact_ir_valley_locs[k] >10){
228  for (i=an_exact_ir_valley_locs[k]; i< an_exact_ir_valley_locs[k+1];
     i++)
229  {
230  if (an_x[i]> n_x_dc_max)
231  {
232  n_x_dc_max =an_x[i];n_x_dc_max_idx =i;
233  }
234  if (an_y[i]> n_y_dc_max)
235  {
236  n_y_dc_max =an_y[i];n_y_dc_max_idx=i;
237  }
238  }
239  n_y_ac= (an_y[an_exact_ir_valley_locs[k+1]] - an_y[
    an_exact_ir_valley_locs[k] ] )*(n_y_dc_max_idx -
    an_exact_ir_valley_locs[k]); //red
240  n_y_ac=  an_y[an_exact_ir_valley_locs[k]] + n_y_ac/ (
    an_exact_ir_valley_locs[k+1] - an_exact_ir_valley_locs[k])  ;
241  n_y_ac=  an_y[n_y_dc_max_idx] - n_y_ac;    // subracting linear DC
    compoenents from raw
242  n_x_ac= (an_x[an_exact_ir_valley_locs[k+1]] - an_x[
    an_exact_ir_valley_locs[k] ] )*(n_x_dc_max_idx -
    an_exact_ir_valley_locs[k]); // ir
243  n_x_ac=  an_x[an_exact_ir_valley_locs[k]] + n_x_ac/ (
    an_exact_ir_valley_locs[k+1] - an_exact_ir_valley_locs[k]);
244  n_x_ac=  an_x[n_y_dc_max_idx] - n_x_ac;      // subracting linear
```

```
DC compoenents from raw
 n_nume=( n_y_ac *n_x_dc_max)>>7 ; //prepare X100 to preserve floating value
 n_denom= ( n_x_ac *n_y_dc_max)>>7;
 if (n_denom>0  && n_i_ratio_count <5 &&  n_nume != 0)
 {
 an_ratio[n_i_ratio_count]= (n_nume*100)/n_denom ; //formular is ( n_y_ac *n_x_dc_max) / ( n_x_ac *n_y_dc_max) ;
 n_i_ratio_count++;
 }
 }
 }
 maxim_sort_ascend(an_ratio, n_i_ratio_count);
 n_middle_idx= n_i_ratio_count/2;

 if (n_middle_idx >1)
 n_ratio_average =( an_ratio[n_middle_idx-1] +an_ratio[n_middle_idx])/2; // use median
 else
 n_ratio_average = an_ratio[n_middle_idx ];

 if( n_ratio_average>2 && n_ratio_average <184)
 {
 n_spo2_calc= uch_spo2_table[n_ratio_average] ;
 *pn_spo2 = n_spo2_calc ;
 *pch_spo2_valid  = 1;//  float_SPO2 =  -45.060*n_ratio_average* n_ratio_average/10000 + 30.354 *n_ratio_average/100 + 94.845 ;  // for comparison with table
 }
 else{
 *pn_spo2 =  -1 ; // do not use SPO2 since signal ratio is out of range
 *pch_spo2_valid  = 0;
 }
 }

 void maxim_find_peaks(int32_t *pn_locs, int32_t *pn_npks, int32_t *pn_x, int32_t n_size, int32_t n_min_height, int32_t n_min_distance, int32_t n_max_num)
 /**
 * \brief        Find peaks
 * \par          Details
 *               Find at most MAX_NUM peaks above MIN_HEIGHT separated by at least MIN_DISTANCE
 *
```

```
280     * \retval         None
281     */
282     {
283     maxim_peaks_above_min_height( pn_locs, pn_npks, pn_x, n_size,
    n_min_height );
284     maxim_remove_close_peaks( pn_locs, pn_npks, pn_x, n_min_distance );
285     *pn_npks = min( *pn_npks, n_max_num );
286     }
287
288     void maxim_peaks_above_min_height(int32_t *pn_locs, int32_t *
    pn_npks, int32_t  *pn_x, int32_t n_size, int32_t n_min_height)
289     /**
290     * \brief         Find peaks above n_min_height
291     * \par           Details
292     *                Find all peaks above MIN_HEIGHT
293     *
294     * \retval         None
295     */
296     {
297     int32_t i = 1, n_width;
298     *pn_npks = 0;
299
300     while (i < n_size-1){
301     if (pn_x[i] > n_min_height && pn_x[i] > pn_x[i-1]) // find left
    edge of potential peaks
302     {
303     n_width = 1;
304     while (i+n_width < n_size && pn_x[i] == pn_x[i+n_width]) // find
    flat peaks
305     n_width++;
306     if (pn_x[i] > pn_x[i+n_width] && (*pn_npks) < 15 ) // find right
    edge of peaks
307     {
308     pn_locs[(*pn_npks)++] = i;
309     // for flat peaks, peak location is left edge
310     i += n_width+1;
311     }
312     else
313     i += n_width;
314     }
315     else
316     i++;
317     }
318     }
319
```

```
320  void maxim_remove_close_peaks(int32_t *pn_locs, int32_t *pn_npks,
     int32_t *pn_x, int32_t n_min_distance)
321  /**
322  * \brief        Remove peaks
323  * \par          Details
324  *               Remove peaks separated by less than MIN_DISTANCE
325  *
326  * \retval       None
327  */
328  {
329
330  int32_t i, j, n_old_npks, n_dist;
331
332  /* Order peaks from large to small */
333  maxim_sort_indices_descend( pn_x, pn_locs, *pn_npks );
334
335  for ( i = -1; i < *pn_npks; i++ )
336  {
337  n_old_npks = *pn_npks;
338  *pn_npks = i+1;
339  for ( j = i+1; j < n_old_npks; j++ )
340  {
341  n_dist =  pn_locs[j] - ( i == -1 ? -1 : pn_locs[i] ); // lag-zero
     peak of autocorr is at index -1
342  if ( n_dist > n_min_distance || n_dist < -n_min_distance )
343  pn_locs[(*pn_npks)++] = pn_locs[j];
344  }
345  }
346
347  // Resort indices longo ascending order
348  maxim_sort_ascend( pn_locs, *pn_npks );
349  }
350
351  void maxim_sort_ascend(int32_t *pn_x,int32_t n_size)
352  /**
353  * \brief        Sort array
354  * \par          Details
355  *               Sort array in ascending order (insertion sort
     algorithm)
356  *
357  * \retval       None
358  */
359  {
360  int32_t i, j, n_temp;
361  for (i = 1; i < n_size; i++)
```

```
362     {
363     n_temp = pn_x[i];
364     for (j = i; j > 0 && n_temp < pn_x[j-1]; j--)
365     pn_x[j] = pn_x[j-1];
366     pn_x[j] = n_temp;
367     }
368     }
369
370     void maxim_sort_indices_descend(int32_t *pn_x, int32_t *pn_indx,
        int32_t n_size)
371     /**
372     * \brief        Sort indices
373     * \par          Details
374     *               Sort indices according to descending order (
        insertion sort algorithm)
375     *
376     * \retval       None
377     */
378     {
379     int32_t i, j, n_temp;
380     for (i = 1; i < n_size; i++)
381     {
382     n_temp = pn_indx[i];
383     for (j = i; j > 0 && pn_x[n_temp] > pn_x[pn_indx[j-1]]; j--)
384     pn_indx[j] = pn_indx[j-1];
385     pn_indx[j] = n_temp;
386     }
387     }
388
```

Listing B.3: MCU-Maxim Integrated PPG algorithm

B.3 TMP117 Manager

```
1   /*
2   * tmp117.c
3   *
4   *  Created on: Nov 2, 2019
5   *      Author: detjon
6   */
7
8
9
10
11  #include "main.h"
```

```
12 #include "tmp117.h"
13
14 I2C_HandleTypeDef i2c;
15
16 uint8_t TMP117_DeviceID = TMP117_DeviceID1;
17
18 /*
19 @Brief          Get temperature basically
20 @Description    Function gives to us ambient temperature
21 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
22 @Return value   Float
23 */
24 uint16_t TMP117_get_Temperature(I2C_HandleTypeDef i2c)
25 {
26 static uint8_t buf[3];
27 uint16_t raw;
28 buf[0]=TMP117_TemperatureRegister;
29
30 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100); // send
    device ID
31 HAL_Delay(30);
32 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100); // get
    teperature from temp_register
33 raw = (((buf[0]<<8)|buf[1]));
34 return raw; // resolution = 0.0078125;
35 }
36
37 /*
38 @Brief          Get Configuration
39 @Description    Get Configuration Register Value
40 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
41 @Return value   uint16_t
42 */
43 uint16_t TMP117_get_Configuration      (I2C_HandleTypeDef i2c)
44 {
45 static uint8_t buf[3];
46 buf[0]=TMP117_ConfigurationRegister;
47
48 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
49 HAL_Delay(1);
50 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
51 return ((buf[0]<<8)|buf[1]);
52 }
53
54 /*
```

```
55  @Brief          Set Configuration
56  @Description    Set Configuration Register for Features
57  @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
58  uint8_t first      -> [15:8]
59  uint8_t second     -> [7:0]
60  @Return value   void
61  */
62  void TMP117_set_Configuration              (I2C_HandleTypeDef i2c,
     uint8_t first,uint8_t second)
63  {
64  static uint8_t buf[3];
65  buf[0]=TMP117_ConfigurationRegister;
66  buf[1]=first;
67  buf[2]=second;
68
69  HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
70  HAL_Delay(1);
71  }
72
73  /*
74  @Brief          Set HighLimit
75  @Description    Set HighLimit for Alert
76  @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
77  uint8_t first      -> [15:8]
78  uint8_t second     -> [7:0]
79  @Return value   void
80  */
81  void TMP117_set_HighLimit                  (I2C_HandleTypeDef i2c,
     uint8_t first,uint8_t second)
82  {
83  static uint8_t buf[3];
84  buf[0]=TMP117_TemperatureHighLimit;
85  buf[1]=first;
86  buf[2]=second;
87
88  HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
89  HAL_Delay(1);
90
91  }
92
93  /*
94  @Brief          Get Highlimit
95  @Description    Get Highlimit Register Value
96  @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
97  @Return value   uint16_t
```

```
98  */
99  uint16_t TMP117_get_HighLimit          (I2C_HandleTypeDef i2c)
100 {
101 static uint8_t buf[3];
102 buf[0]=TMP117_TemperatureHighLimit;
103 
104 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
105 HAL_Delay(1);
106 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
107 return ((buf[0]<<8)|buf[1]);
108 
109 }
110 
111 /*
112 @Brief          Set LowLimit
113 @Description    Set LowLimit for Alert
114 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
115 uint8_t first     -> [15:8]
116 uint8_t second    -> [7:0]
117 @Return value   void
118 */
119 void TMP117_set_LowLimit                  (I2C_HandleTypeDef i2c,
    uint8_t first,uint8_t second)
120 {
121 static uint8_t buf[3];
122 buf[0]=TMP117_TemperatureLowLimit;
123 buf[1]=first;      //Reset Value
124 buf[2]=second;     //Reset Value
125 
126 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
127 HAL_Delay(1);
128 
129 }
130 
131 /*
132 @Brief          Get LowLimit
133 @Description    Get Lowlimit Register Value
134 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
135 @Return value   uint16_t
136 */
137 uint16_t TMP117_get_LowLimit           (I2C_HandleTypeDef i2c)
138 {
139 static uint8_t buf[3];
140 buf[0]=TMP117_TemperatureLowLimit;
141 
```

```
142 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
143 HAL_Delay(1);
144 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
145 return ((buf[0]<<8)|buf[1]);
146
147 }
148
149 /*
150 @Brief          Get EEPROM Unlock Register Value
151 @Description    Check EEPROM for Unlock/Lock
152 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
153 @Return value   uint16_t
154 */
155 uint16_t TMP117_get_EEPROM_Unclock     (I2C_HandleTypeDef i2c)
156 {
157 static uint8_t buf[3];
158 buf[0]=TMP117_EEPROM_Uclock;
159
160 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
161 HAL_Delay(1);
162 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
163 return ((buf[0]<<8)|buf[1]);
164
165 }
166
167 /*
168 @Brief          Set EEPROM Unlock Register Value
169 @Description    Active/Inactive for EEPROM read/write situation
170 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
171 uint8_t first      -> [15:8]
172 uint8_t second     -> [7:0]
173 @Return value   void
174 */
175 void TMP117_set_EEPROM_Unlock             (I2C_HandleTypeDef i2c,
    uint8_t first,uint8_t second)
176 {
177 static uint8_t buf[3];
178 buf[0]=TMP117_EEPROM_Uclock;
179 buf[1]=first;
180 buf[2]=second;
181
182 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
183 HAL_Delay(1);
184
185 }
```

```
186
187 /*
188 @Brief
189 @Description
190 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
191 uint8_t first      -> [15:8]
192 uint8_t second     -> [7:0]
193 @Return value  void
194 */
195 void TMP117_set_EEPROM1                    (I2C_HandleTypeDef i2c,
    uint8_t first,uint8_t second)
196 {
197 static uint8_t buf[3];
198 buf[0]=TMP117_EEPROM1;
199 buf[1]=first;      //Reset Value
200 buf[2]=second;     //Reset Value
201
202 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
203 HAL_Delay(1);
204 }
205
206 /*
207 @Brief
208 @Description
209 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
210 @Return value  uint16_t
211 */
212 uint16_t TMP117_get_EEPROM1            (I2C_HandleTypeDef i2c)
213 {
214 static uint8_t buf[3];
215 buf[0]=TMP117_EEPROM1;
216
217 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
218 HAL_Delay(1);
219 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
220 return ((buf[0]<<8)|buf[1]);
221
222 }
223
224 /*
225 @Brief
226 @Description
227 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
228 uint8_t first      -> [15:8]
229 uint8_t second     -> [7:0]
```

```
230 @Return value   void
231 */
232 void TMP117_set_EEPROM2 (I2C_HandleTypeDef i2c,uint8_t first,uint8_t
     second)
233 {
234 static uint8_t buf[3];
235 buf[0]=TMP117_EEPROM2;
236 buf[1]=first;
237 buf[2]=second;
238
239 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
240 HAL_Delay(1);
241 }
242
243 /*
244 @Brief
245 @Description
246 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
247 @Return value   uint16_t
248 */
249 uint16_t TMP117_get_EEPROM2             (I2C_HandleTypeDef i2c)
250 {
251 static uint8_t buf[3];
252 buf[0]=TMP117_EEPROM2;
253
254 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
255 HAL_Delay(1);
256 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
257 return ((buf[0]<<8)|buf[1]);
258 }
259
260 /*
261 @Brief
262 @Description
263 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
264 uint8_t first      -> [15:8]
265 uint8_t second     -> [7:0]
266 @Return value   void
267 */
268 void TMP117_set_EEPROM3                    (I2C_HandleTypeDef i2c,
     uint8_t first,uint8_t second)
269 {
270 static uint8_t buf[3];
271 buf[0]=TMP117_EEPROM3;
272 buf[1]=first;
```

```
273   buf[2]=second;
274
275   HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
276   HAL_Delay(1);
277   }
278
279   /*
280   @Brief          Get EEPROM3 Value
281   @Description
282   @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
283   @Return value   uint16_t
284   */
285   uint16_t TMP117_get_EEPROM3             (I2C_HandleTypeDef i2c)
286   {
287   static uint8_t buf[3];
288   buf[0]=TMP117_EEPROM3;
289
290   HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
291   HAL_Delay(1);
292   HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
293   return ((buf[0]<<8)|buf[1]);
294
295   }
296
297   /*
298   @Brief          Set Temperature Offset Value
299   @Description    Set Temperature Offset Value for Calibrating
300   @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
301   uint8_t first     -> [15:8]
302   uint8_t second    -> [7:0]
303   @Return value   void
304   */
305   void TMP117_set_Temperature_Offset       (I2C_HandleTypeDef i2c,
       uint8_t first,uint8_t second)
306   {
307   static uint8_t buf[3];
308   buf[0]=TMP117_Temperature_Offset;
309   buf[1]=first;
310   buf[2]=second;
311
312   HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,2,100);
313   HAL_Delay(1);
314
315   }
316
```

```
317 /*
318 @Brief          Get Temperature Offset Value
319 @Description    Get Temperature Offset Value for Calibrating
320 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
321 @Return value   uint16_t
322 */
323 uint16_t TMP117_get_Temperature_Offset(I2C_HandleTypeDef i2c)
324 {
325 static uint8_t buf[3];
326 buf[0]=TMP117_Temperature_Offset;
327 
328 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
329 HAL_Delay(1);
330 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
331 return ((buf[0]<<8)|buf[1]);
332 
333 }
334 
335 /*
336 @Brief          Get ID Register
337 @Description    Check Device ID for Error Handler
338 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
339 @Return value   uint16_t
340 */
341 uint16_t TMP117_get_ID_Register(I2C_HandleTypeDef i2c)
342 {
343 static uint8_t buf[3];
344 buf[0]=TMP117_ID_Register;
345 
346 HAL_I2C_Master_Transmit(&i2c,TMP117_DeviceID,buf,1,100);
347 HAL_Delay(1);
348 HAL_I2C_Master_Receive(&i2c,TMP117_DeviceID,buf,2,100);
349 return ((buf[0]<<8)|buf[1]);
350 
351 }
352 
353 /*
354 @Brief          Custom Initialization
355 @Description    Custom Parameters for Sensor
356 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
357 @Return value   void
358 */
359 void TMP117_Initialization(I2C_HandleTypeDef i2c)
360 {
361 TMP117_set_Configuration(i2c, 0x0C, 0x0C);
```

```
362 TMP117_set_Temperature_Offset(i2c,0x00,0x00);
363 TMP117_set_LowLimit(i2c,0x12,0x80);
364 TMP117_set_HighLimit(i2c,0x76,0x80);
365 }
366
367 /*
368 @Brief          Default Initialization
369 @Description    Default Parameters for Sensor
370 @Parameter      I2C_HandleTypeDef -> HAL_I2C Handle
371 @Return value   void
372 */
373 void TMP117_Initialization_DEFAULT(I2C_HandleTypeDef i2c)
374 {
375 TMP117_set_Configuration(i2c,0x02,0x20);
376 TMP117_set_Temperature_Offset(i2c,0x00,0x00);
377 TMP117_set_LowLimit(i2c,0x80,0x00);
378 TMP117_set_HighLimit(i2c,0x60,0x00);
379 TMP117_set_EEPROM_Unlock(i2c,0x00,0x00);
380 }
381
```

Listing B.4: MCU - TMP117 sensor Manager

B.4 CAT24C128

```
1  /*
2  * CAT24C128.c
3  *
4  *  Created on: Dec 2, 2019
5  *      Author: detjon
6  */
7
8
9
10
11 #include "main.h"
12 #include "CAT24C128.h"
13 #include "string.h"
14 #include "stdio.h"
15
16
17
18 /**
19 * @brief               : This function handles Writing Array of Bytes
     on the specific Address
```

```
20
21  * @param  hi2c          : Pointer to a I2C_HandleTypeDef structure
     that contains
22  *                        the configuration information for the
     specified I2C
23  * @param  PageNumber    : Internal memory page address (WHERE YOU
     WANNA WRITE TO)
24  * @param  pData         : Pointer to data buffer
25  * @retval
26  */
27  int cat24_HAL_WritePage(I2C_HandleTypeDef *hi2c, uint8_t PageNumber,
     uint8_t *pData, uint8_t TxBufferSize)
28  {
29  /*
30  * give PageNumber for the location you want to write to
31  * give Data buffer so it can write Data on this location
32  */
33  uint8_t Size;
34  Size = 2 + TxBufferSize;
35  uint8_t Buff[Size];
36  int i;
37  Buff[0] = PageNumber;
38  Buff[1] = 0x00;
39  for(i=2; i<=Size;i++)
40  {
41  Buff[i] = pData[i-2];
42  }
43
44  HAL_I2C_Master_Transmit(hi2c,CAT_Slave_W,Buff,Size,50);
45
46
47  return 1;
48  }
49
50  /**
51  * @brief               : This function handles Reading Array of Bytes
     on the specific Address
52
53  * @param  hi2c          : Pointer to a I2C_HandleTypeDef structure
     that contains
54  *                        the configuration information for the
     specified I2C
55  * @param  PageNumber    : Internal memory page address (WHERE YOU
     WANNA READ FROM)
56  * @param  pData         : Pointer to data buffer
```

```
57  * @retval
58  */
59  int cat24_HAL_ReadPage(I2C_HandleTypeDef *hi2c, uint8_t PageNumber,
      uint8_t *pData, uint8_t TxBufferSize)
60  {
61  //  char BufferUart[10];
62  uint8_t Buff[3];
63  uint8_t BuffData[TxBufferSize];
64  int i;
65  /*
66  * get the PageNumber for the location you want to read data on it
67  * get the Data buffer so it can read Data starting from this location
68  */
69  Buff[0] = PageNumber;
70  Buff[1] = 0x00;
71
72  HAL_I2C_Master_Transmit(hi2c,CAT_Slave_R,Buff,2,50);
73  HAL_I2C_Master_Receive(hi2c,CAT_Slave_R,BuffData,TxBufferSize,50);
74  for(i=0; i<64;i++)
75  {
76  pData[i] = BuffData[i];
77  }
78
79  return 1;
80  }
81
82
83
84  /**
85  * @brief               : This function handles Specific Page ERASE
86
87  * @param  hi2c         : Pointer to a I2C_HandleTypeDef structure
      that contains
88  *                        the configuration information for the
      specified I2C
89  * @param  PageNumber   : Internal memory page address (PAGE YOU WANT
     TO ERASE)
90  * @retval
91  */
92
93  int cat24_HAL_ErasePage(I2C_HandleTypeDef *hi2c, uint8_t PageNumber)
94  {
95  uint8_t EraseBuf[66] ={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

```
96  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
      ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00};
97  EraseBuf[0] = PageNumber;
98
99  HAL_I2C_Master_Transmit(hi2c,CAT_Slave_W,EraseBuf,66,100);
100
101
102 return 1;
103 }
104 /*
105 * @brief                : This function handles Erase Full chip.
106 * @param  hi2c          : Pointer to a I2C_HandleTypeDef structure
      that contains
107 *                         the configuration information for the
      specified I2C.
108 * @retval
109 */
110
111 int cat24_HAL_EraseMemFull(I2C_HandleTypeDef *hi2c)
112 {
113 /*
114 * 0x00 means that block is empty
115 */
116 uint8_t EraseBuf[66] ={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
117 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
      ,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
      x00,0x00,0x00,0x00,0x00};
118 int i;
119 for (i =0 ; i<256 ; i++)
120 {
121 HAL_I2C_Master_Transmit(hi2c,CAT_Slave_W,EraseBuf,66,250);
122 EraseBuf[0] = EraseBuf[0] + 0x01;
123
124 }
125 return 1;
126 }
127
128 /*
129 * @brief                : This function handles Reading Array of Bytes
      from the specific Address .
130 *              This program have this feature that don't force you to
      use absolute 16 bytes
```

```
131  *              you can use more than 16 bytes buffer.
132  * @param  hi2c          : Pointer to a I2C_HandleTypeDef structure
       that contains
133  *                        the configuration information for the
       specified I2C.
134  * @param  DevAddress    : specifies the slave address to be programmed
       (EEPROM ADDRESS).
135  * @param  MemAddress    : Internal memory address (WHERE YOU WANNA
       READ FROM)
136  * @param  pData         : Pointer to data buffer
137  * @param  TxBufferSize : Amount of data to be Read
138  * @retval
139  */
140
```

Listing B.5: MCU - TMP117 sensor Manager

APPENDIX C

APPLICATION DESKTOP SCRIPTS

C.1 Main

```
from Tkinter import *
import Serial_Read_Device
import Serial_Read_Temperature
import Plot_Device_data
import Dlete_EEPROM
import os

i = 0


def func():
Serial_Read_Temperature.run()


def func1():
Serial_Read_Device.run()


def func2():
Plot_Device_data.run()


def func3():
Dlete_EEPROM.run()


# create a tkinter window
root = Tk()
# Open window having dimension 100x100
root.geometry('400x700')
```

```
31
32      var = StringVar()
33      label = Label(root, textvariable=var, relief=RAISED, font=50)
34
35      var.set("Choose what you want to do:")
36      label.pack()
37
38      # Create a Button
39      b_readT = Button(root, text='Read Temperature!', width=44, height
    =7, bd='5', command=func, font=100)
40      b_readP = Button(root, text='Read PPG!', width=44, height=7, bd='5'
    , command=func1,
41      font=100)
42      b_Plot = Button(root, text='Plot !', width=44, height=7, bd='5',
    command=func2, font=100)
43      b_Delete = Button(root, text='Delete memory!', width=44, height=7,
    bd='5', command=func3, font=100)
44
45      # Set the position of button on the top of window.
46      b_Plot.pack()
47      b_readT.pack()
48      b_readP.pack()
49      b_Delete.pack()
50
51      b_readT.place(x=0, y=50)
52      b_readP.place(x=0, y=200)
53      b_Plot.place(x=0, y=350)
54      b_Delete.place(x=0, y=500)
55
56      root.mainloop()
57
```

Listing C.1: Desktop Application - Main

C.2 Serial Read Device

```
1       # This script open the serial port and reads line by line
2       # and print each line in a desired file
3       # then the file can be used to print
4       # a plot of the data
5       import serial
6       import time
7
8
9       def run():
```

```
10      try:
11      ser = serial.Serial('COM5', 4800, timeout=3)
12      i = 0
13      j = 0
14      val = 0
15
16      print ("Write a name for the file to save the Data\n")
17      print ("Rule:\n 1)Not use space, instead use underscore --> _ \n2
    )Do not add a file extension\n")
18      print ("Insert a file name\n")
19      filename = raw_input()
20      filename = filename + ".txt"
21
22      while True and ser.isOpen() and val == 0:
23      data = ser.readline()
24      while data == "":
25      ser.write("READ")
26      ser.write("\n")
27      data = ser.readline()
28      print "waiting for data"
29
30      print "Reading..."
31      while data != "":
32      f = open(filename, 'a')
33      data = str(data)
34      f.write(data)
35      f.close()
36      i += 1
37      data = ser.readline()
38
39      if i == 1104:
40      val = 1
41      print "Done, Bye bye"
42      ser.close()
43      except serial.serialutil.SerialException:
44      print "No device connected, check and try again"
45      time.sleep(1)
46
47
48      if __name__ == "__main__":
49      run()
50
51
```

Listing C.2: Desktop Application - Reading Script, 48h period

C.3 Serial Read Temperature

```
    # This script open the serial port and reads line by line
    # and print each line in a desired file
    # then the file is used to print a plot
    # This is the PPG 4 places test
    import serial
    import time


    def run():
    try:
    ser = serial.Serial('COM5', 4800, timeout=3)

    i = 0
    j = 0
    val = 0

    print ("Write a name for the file to save the Temperature Data\n"
)
    print ("Rule:\n 1)Not use space, instead use underscore --> _ \n2
)Do not add a file extension\n")
    print ("Insert a file name\n")
    filename = raw_input()
    filename = filename + ".txt"
    while True and ser.isOpen() and val == 0:
    data = ser.readline()
    while data == "":
    ser.write("READ")
    data = ser.readline()
    print "waiting for data"

    print "Reading..."
    while data != "":
    f = open(filename, 'a')
    data = str(data)
    f.write(data)
    f.close()
    i += 1
    data = ser.readline()
    if i == 1440:
    val = 1
    print "Done, Bye bye"
    ser.close()
```

```
41
42      except serial.serialutil.SerialException:
43      print "No device connected, check and try again"
44      time.sleep(1)
45
46
47      if __name__ == "__main__":
48      run()
49
```

Listing C.3: Desktop Application - Reading Script, 72h period

C.4 Plot Data

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import Tkinter
4   import tkFileDialog
5   import os.path
6   import time
7
8
9   def openfile():
10  return tkFileDialog.askopenfilename(initialdir="/C", title="Select
    file", filetype=(("Text files", "*.txt"),))
11
12
13  def run():
14  ok = 0
15  while ok == 0:
16  i = 0
17  case = 0
18  print "Welcome!"
19  print "Please, choose a file to load data"
20  time.sleep(1.8)
21  try:
22  filename = openfile()
23  Data = np.loadtxt(filename, delimiter='\n', unpack=True)
24  Data_len = len(Data)
25  print Data_len
26
27  if Data_len == (1440-1):     # 72h Measure = 1440 Temperature (@3min
    )
28  Temperature = [0 for q in range(Data_len)]
29  Temp_time = [0 for q in range(Data_len)]
```

```
30      case = 1
31      else:     # 48h Measure = 960 Temperature(@3min) + 72 HR(@40min) +
    72 SpO2(@40min)
32      Temperature = [0 for q in range(960 - 1)]
33      Temp_time = [0 for q in range(960 - 1)]
34      HR = [0 for q in range(72 - 1)]
35      SpO2 = [0 for q in range(72 - 1)]
36      PPG_time = [0 for q in range(72 - 1)]
37      case = 2
38
39      if case == 1:
40      for i in range(len(Temperature)):
41      Temperature[i] = Data[i]
42      Temp_time[i] = i
43
44      Plot_name = os.path.basename(filename)
45      lab = Plot_name[:-4]
46
47      plt.subplot(1, 1, 1)
48      plt.title(lab)
49      plt.ylabel("Temperature")
50      plt.xlabel('sample (i)')
51      plt.plot(Temp_time, Temperature, 'r')
52
53      plt.show()
54
55      if case == 2:
56      for i in range(len(Temperature)):
57      Temperature[i] = Data[i]
58      Temp_time[i] = i
59      for i in range(len(HR)):
60      HR[i] = Data[i+960]
61      SpO2[i] = Data[i+960+72]
62      PPG_time[i] = i
63
64      Plot_name = os.path.basename(filename)
65      lab = Plot_name[:-4]
66
67      f = plt.figure(1)
68      plt.subplot(1, 1, 1)
69      plt.title(lab)
70      plt.ylabel("Temperature")
71      plt.xlabel('sample (i)')
72      plt.plot(Temp_time, Temperature, 'k')
73
```

```
74      g = plt.figure(2)
75      plt.subplot(2, 1, 1)
76      plt.title(Plot_name)
77      plt.ylabel("Heart Rate (bpm)")
78      plt.plot(PPG_time, HR, 'r')
79      plt.subplot(2, 1, 2)
80      plt.ylabel("Oxygen saturation (%)")
81      plt.xlabel('sample (i)')
82      plt.plot(PPG_time, SpO2, 'r')
83
84      plt.show()
85      ok = 1
86      except ValueError:
87      print "The file chosen is not in the correct format"
88      ok = 1
89      except IOError:
90      ok = 1
91
92      time.sleep(1)
93
94
95      if __name__ == "__main__":
96      run()
97
98
```

Listing C.4: Desktop Application - Plotting Script

C.5 Delete EEPROM

```
1       # This script open the serial port and writes DELETE
2       # in order to communicate with the MCU and induce it
3       # to Delete all the written pages
4       import serial
5       import time
6
7
8       def run():
9       try:
10      ser = serial.Serial('COM5', 4800, timeout=3)
11      i = 0
12      print "This program will Delete all the data in the memory"
13      time.sleep(1)
14      print "Continue?[y]yes, any other key to exit "
15      OK = raw_input()
```

```
16      if OK == "y":
17      while i < 5:
18      ser.write("DELETE")
19      ser.write("\n")
20      i += 1
21      else:
22      exit()
23      print "Memory deleted successfully"
24      except serial.serialutil.SerialException:
25      print "No device connected, check and try again"
26      time.sleep(1)
27
28
29      if __name__ == "__main__":
30      run()
31
32
```

Listing C.5: Desktop Application - Delete EEPROM Script

BIBLIOGRAPHY

[1] Duarte Dias, Joao Paulo Silva Cunha, Wearable Health Devices-Vital Sign Monitoring, Systems and Technologies, [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6111409/ (Accessed on 27 February 2020)

[2] CNC University of Coimbra(UC), Centro Hospitalar of UC et. al., Sleep Apnea in a blood drop, Coimbra -ISR , Portugal, Agust, 2019.

[3] Czeisler&Klerman, .RecentProgress in Hormone Research, 1999

[4] Conroy et al. , Journal of Circadian Rhythms, 2005

[5] Hastings et al. , Journal of Endocrinology, 2007

[6] Word Health Organization, Cardiovascular diseases (CVDs),[Online] Available: https://www.who.int/en/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds) , (Accessed on 27 February 2020).

[7] Department of Anesthesia & Analgesia, Takuo Aoyagi: Discovery of Pulse Oximetry, University of California San Francisco, San Francisco, California, USA, vol. 105, issue 6. P S1-S4, 2007

[8] Ohad Yossef Hay, Meir Cohen,Itamar Nitzan et al., Pulse Oximetry with Two Infrared Wavelengths without Calibration in Extracted Arterial Blood, Published online 2018 Oct 15

[9] Maxim Integrated, GUIDELINES FOR SPO2 MEASUREMENT, [Online] Available: https://www.maximintegrated.com/en/design/technical-documents/app-notes/6/6845.html , (Accessed on 27 February 2020)

[10] Sally K. Longmore, Gough Y. Lui et al., A Comparison of Reflective Photoplethysmography for Detection of Heart Rate, Blood Oxygen Saturation, and Respiration Rate at Various Anatomical Locations, [Online] Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6514840/, (Accessed on 27 February 2020)

[11] F. Mendonca, S. S. Mostafa, A. G. Ravelo-Garcia, F. Morgado-Dias, and T. Penzel, "A Review of Obstructive Sleep Apnea Detection Approaches," IEEE J. Biomed. Heal. Informatics, vol. 23, no. 2, pp. 825–837, Mar. 2019.

[12] Dias, D., & Paulo Silva Cunha J., Wearable Health Devices-Vital Sign Monitoring, Systems and Technologies. Sensors (Basel, Switzerland), 18(8), 2414, 2018

[13] Frank M., Your Head Is Better for Sensors than Your Wrist. Outside-Live Bravely; Santa Fe, NM, USA: 2015.

[14] Cosinuss, "Cosinuss° One – Performance Monitoring". [Online]. Available: https://www.cosinuss.com/products/one/ (Accessed on 27 February 2020).

[15] Lukowicz, P.; Anliker, U.; Ward, J.; Troster, G.; Hirt, E.; Neufelt, AMON C., A wearable medical computer for high risk patients. In Proceedings of the Sixth International Symposium on Wearable Computers, Seattle, WA, USA, pp. 133–134, 2002

[16] Bieber, G.; Haescher, M.; Vahl, M., Sensor requirements for activity recognition on smart watches. In Proceedings of the Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments, Rhodes, Greece, 29–31, 2013.

[17] BASIS. PEAK—The Ultimate Fitness and Sleep Tracker. Available online: https://www.mybasis.com/ (Accessed on 10 December 2019).

[18] B. Part, Caretaker4 User Manual. Caretaker4, 2018.

[19] Nonin Medical Inc., WristOx2 Pulse Oximeter, 2018.

[20] Severinghaus JW, Honda Y (April 1987). "History of blood gas analysis. VII. Pulse oximetry". Journal of Clinical Monitoring. 3 (2): 135–8. doi:10.1007/bf00858362.

[21] Valenza T (April 2008). Keeping a Pulse on Oximetry. Archived from the original on February 10, 2012.

[22] Elite HRV, CorSense®, [Online]. Available: https://elitehrv.com/corsense (Accessed on 27 February 2020).

[23] Motiv, Motiv ring - Performance. [Online]. Available: https://mymotiv.com/the-ring/ (Accessed on 27 February 2020).

[24] Vol $23003_510(k)$ summary, Food and Drug Administration, 2019. [Online]. Available: https://www.accessdata.fda.gov/cdrh_docs/pdf18/K183078.pdf (Accessed on 10 December 2019).

[25] VitalConnect, VitalPatch. [Online]. Available: https://mymotiv.com/the-ring/ (Accessed on 10 December 2019).

[26] QARDIO, QardioCore. [Online]. Available: https://www.getqardio.com/qardiocore-wearable-ecg-ekg-monitor-iphone/ (Accessed on 10 December 2019).

[27] Adam Gacek; Witold Pedrycz, ECG Signal Processing, Classification and Interpretation: A Comprehensive Framework of Computational Intelligence. Springer, p. 108, 2011

[28] S.A. VitalJacket®, Biodevices, [Online]. Available: http://www.vitaljacket.com/ (Accessed on 10 December 2019).

[29] Zephyr, Performance Systems. BioHarness™ 3. [Online]. Available: http://www.zephyranywhere.com/ products/bioharness-3 (Accessed on 10 December 2019).

[30] Cityzen Sciences, D-Shirt. [Online]. Available: http://www.cityzensciences.fr/en/ (Accessed on 10 December 2019).

[31] AiQ, Smart Clothing. Bioman. [Online]. Available: http://www.aiqsmartclothing.com/ (Accessed on 10 December 2019).

[32] Hexoskin. [Online]. Available: http://www.hexoskin.com/ (Accessed on 10 December 2019).

[33] Clearbridge Vitalsigns Pte Ltd., CardioLeaf® FIT Shirt. [Online]. Available: http://www.clearbridgevitalsigns. com/ (Accessed on 10 December 2019).

[34] Vivonoeticsk, Smartex WWS. [Online]. Available: http://vivonoetics.com/ (Accessed on 10 December 2019).

[35] Nuubo, Wearable Medical Technologies. [Online]. Available: http://www.nuubo.com/ (Accessed on 10 December 2019).

[36] iRhythm, ZIO XT Patch. [Online]. Available: https://www.irhythmtech.com/products-services/zio-xt (Accessed on 10 December 2019).

[37] Garmin Ltd., HRM-Tri™. [Online]. Available: https://buy.garmin.com (Accessed on 10 December 2019).

[38] Clearbridge Vitalsigns Pte Ltd., CardioLeaf FIT. [Online]. Available: http://www.clearbridgevitalsigns.com/ (Accessed on 10 December 2019).

[39] BioTelemetry Technology ApS, The ePatch® technology. [Online]. Available: http://epatch.madebydelta.com/ (Accessed on 10 December 2019).

[40] Vital Connect, HealthPatch® MD. [Online]. Available: http://www.vitalconnect.com/ (Accessed on 10 December 2019).

[41] Clearbridge Vitalsigns Pte Ltd., CardioLeaf ULTRA. [Online]. Available: http://www.clearbridgevitalsigns. com/ (Accessed on 10 December 2019).

[42] ZephyrTM, Performance Systems. HxM. [Online]. Available: https://www.zephyranywhere.com (Accessed on 10 December 2019).

[43] Duun S.B.; Haahr R.G.; Birkelund K. et al., A Novel Photodiode for Reflectance Pulse Oximetry in low-power applications, Proceedings of the 29th Annual International Conference of the IEEE EMBS, 2007

[44] Duun S.B.; Haahr R.G.; Birkelund K.; Thomsen E.V., A Ring-Shaped Photodiode Designed for Use in a Reflectance Pulse Oximetry Sensor in Wireless Health Monitoring Applications, IEEE SENSORS JOURNAL, VOL. 10, NO. 2, 2010

[45] Duun S.B.; Haahr R.G.; Birkelund K. et al., An Electronic Patch for Wearable Health Monitoring by Reflectance Pulse Oximetry, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, VOL. 6, NO. 1, 2012

[46] A. Pantelopoulos and N. Bourbakis, A survey on wearable biosensor systems for health monitoring, in 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Aug. 2008, pp. 4887–4890.

[47] Texas Instruments , TMP117 high-accuracy, low-power, digital temperature sensor with SMBus™- and I2C-compatible interface datasheet (Rev. B). [Online] Available: http://www.ti.com/product/TMP117/technicaldocuments

[48] Maxim Integrated, High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health. [Online]. Available: https://www.maximintegrated.com/en/products/interface/sensor-interface/MAX30102.html

[49] Eetimes embedded, 2017 Embedded Markets Study-Integrating IoT and Advanced Technology Designs, Application Development & Processing Environments. April 2017. [Online] Available: https://m.eet.com/media/1246048/2017-embedded-market-study.pdf

[50] ST Microelectronics, Ultra-low-power STM32L072xx. [Online] Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l072kb.html

[51] Maxim Integrated, Low-Voltage, 400mA Step-Down DC-DC Converter. [Online] Available: https://datasheets.maximintegrated.com/en/ds/MAX1920-MAX1921.pdf

[52] ON Semiconductor , EEPROM Serial 128-Kb I2C. [Online] Available: https://www.onsemi.com/pub/Collateral/CAT24C128-D.PDF

[53] Voltera, Voltera V-One. [Online] Available: https://www.voltera.io/docs/user-manual/

[54] ST Microelectronics®, Ultra-low-power STM32L072xx. [Online] Available: https://www.st.com/en/microcontrollers-microprocessors/stm32l072kb.html (Accessed on 10 December 2019).

[55] Max Integrated®, MAXREFDES117#: Heart rate and pulse oximetry monitor [Online] Available: https://www.maximintegrated.com/en/design/reference-design-center/system-board/6300.html/tb_tab2 (Accessed on 10 December 2019).

[56] Future Technology Devices International Ltd, TTL-232RG [Online] Available:http://www.farnell.com/datasheets/1066297.pdf?_ga=2.219477209.1594757145.1581430776-1581430776.CjwKCAiAvonyBRB7EiwAadauqSRMFeP6XWO8V3WV_gjkmcq9qNxiSdnuuUCFrltZhp7r16sLwiqFxhoC4z4QAvD_BwE (Accessed on 10 December 2019).

[57] Ibrahim D., ARM-based Microcontroller Projects Using mbed, Newnes, Cambridge, first edition, pp. 305-314, 2019.

[58] Raj Kamal, Embedded systems: architecture, programming and design Tata McGraw Hill Publishing Company Limited, New Delhi, pp. 97-117, 2003.

[59] ARM®, Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2[Online] Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0031c/index.html (Accessed on 10 December 2019).

[60] Python,Tkinter — Python interface to Tcl/Tk.[Online] Available:https://docs.python.org/2/library/tkinter.html(Accessed on 27 February 2020).

[61] ST Microelectronics, AN2606-Application note, [Online] Available: https://www.st.com/content/ccc/resource/technical/document/application_note/b9/9

b/16/3a/12/1e/40/0c/CD00167594.pdf/files/CD00167594.pdf/jcr:content/translations/en.CD00167594.pdf, (Accessed on 27 February 2020)