ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Grado en ingeniería de la salud

Diseño y desarrollo de un medidor de presión sanguínea subcutáneo

Design and development of a subcutaneous blood pressure monitor

Realizado por

Fernando Quevedo Vallejo

Tutorizado por

Antonio Jesús Bandera Rubio

Departamento

Departamento de tecnología electrónica

UNIVERSIDAD DE MÁLAGA

MÁLAGA, Junio 2019

Fecha defensa:

Fdo.: El Secretario del Tribunal

To all those who have sailed
with me through time...

Abstract

In the recent years cardiovascular disease have become one of the most common death causes, and it is expected to rise in the next years. Disease like hypertension, among others, can be easily treated with a constant monitorization of blood pressure, however no devices capable of performing ambulatory continuous monitoring are available for such patients, and only non continuous methods can be used to measure the vital sign. The aim of the project is the design of an implantable solution capable of continuous blood pressure monitoring.

Keywords: blood pressure, Bluetooth, measurement.

Resumen

En los últimos años las enfermedades cardiovasculares se han convertido en una de causas de muerte más comunes, y esta tasa se espera que crezca en los próximos años. Enfermedades como la hipertensión, entre otras, son fácilmente tratables si se lleva una correcta monitorización de la presión sanguínea, sin embargo no existen dispositivos capaces de realizar una monitorización ambulatoria continua para estos pacientes, y solo pueden utilizar soluciones puntual para la medición de este signo vital. El objetivo de este proyecto es el diseño de una solución implantarle capaz de realizar una medición continua de la presión sanguínea.

Palabras cables: presión sanguinea, Bluetooth, medición.

Contents

1. Introduction

1.1 Background

Blood pressure was first measured by Stephen Hales, who in 1708 directly connected the left crural artery of a horse to a 9-foot-tall glass manometer using brass tubes and the trachea of a goose. The German physiologist Karl von Vierordt introduced the indirect method of measuring blood pressure in 1855, based on the principle that blood pressure is equal to the amount of external pressure necessary to obliterate the distal pulse. Indirect measurements required cumbersome mechanical devices and were not widely accepted until 1896, when the Italian Scipione Riva-Rocci invented the blood pressure cuff (McGee, 2016).

Blood pressure is considered one of the four primary vital signs. In critically-ill, blood pressure is measured in order to choose therapies and drugs titration. It can also predict death in patients hospitalised in the intensive care unit (likelihood ratio [LR] = 3.1), in patients with bacteremia (LR = 4.9), pneumonia (LR = 7.6) and myocardial infarction (LR = 15.5) as we can see in Figure 1.1. Presumably, it can predict mortality in many other acute disorders as well, however more research must be made in order to confirm those hypothesis (McGee, 2016).

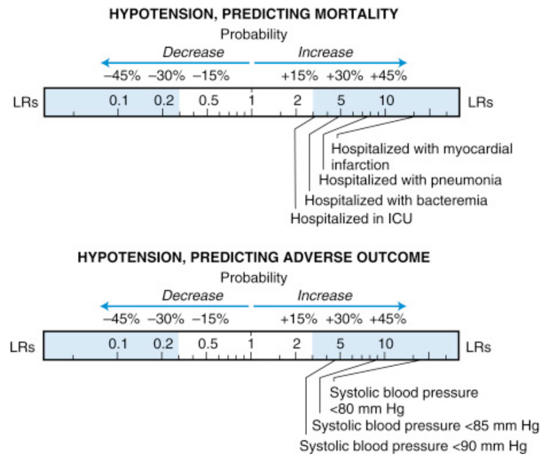


Figure 1.1: Hypotension and prognosis. Source: (McGee, 2016).

Every year, an estimated 17 million people globally die of cardiovascular diseases (CVD), nearly a third of the total global deaths. Within CVD, hypertension is responsible for at least 45% of deaths due to heart diseases (see Figure 1.2) and 51% of deaths due to stroke (Figure 1.3). In 2008, worldwide, approximately 40% of adults aged 25 and above had been diagnosed with hypertension, which has rose from 600 million in 1980 to 1 billion in 2008. This increase of hypertension diagnosis is attributed to population growth, ageing and behavioural risk factors, such as unhealthy diet, harmful use of alcohol, lack of physical activity, excessive weight, and exposure to persistent stress (World Health Organization, 2013).

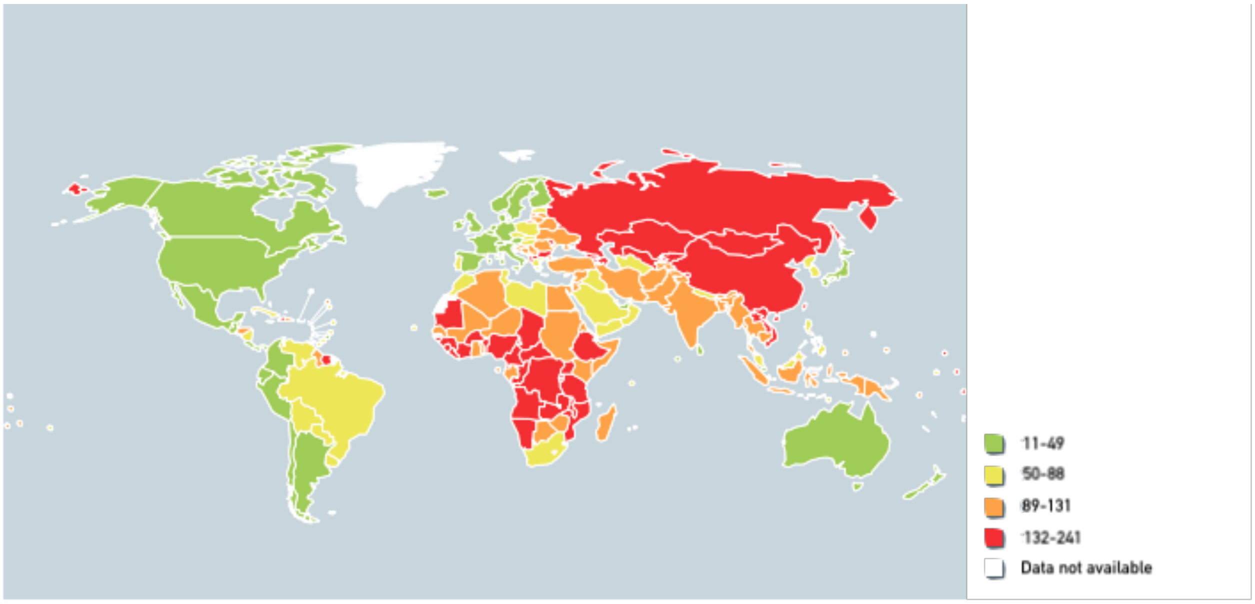


Figure 1.2: Cerebrovascular disease mortality rates. Source: (World Health Organization, 2013).

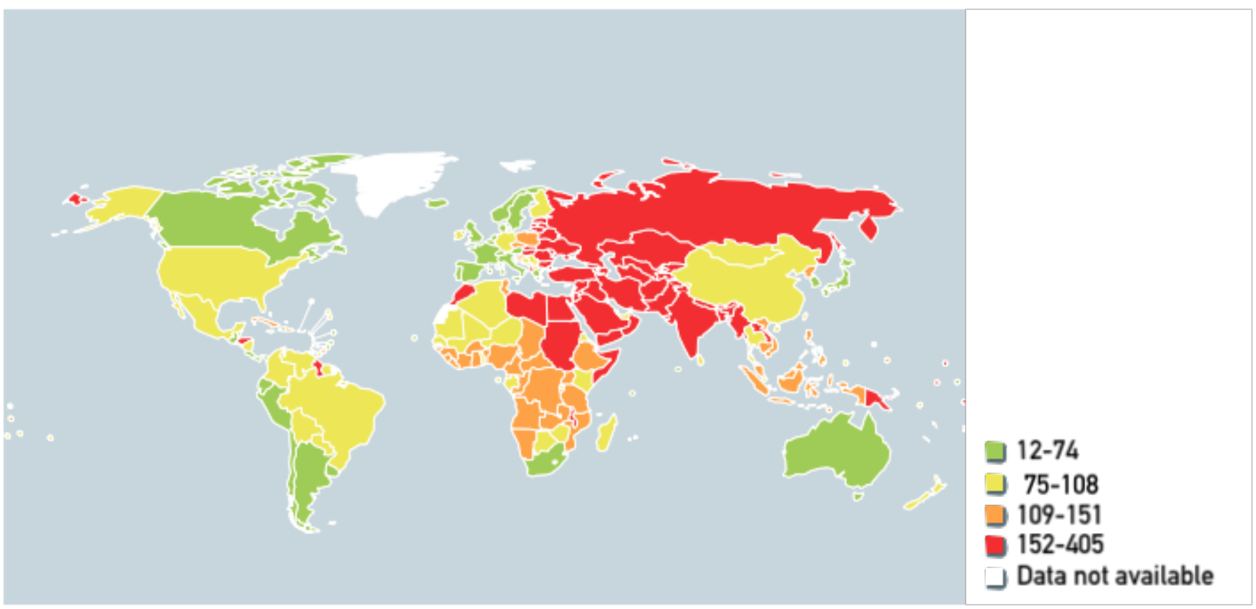


Figure 1.3: Ischemic heart disease mortality rates. Source: (World Health Organization, 2013).

Even though risks are well known, cardiovascular disease are expected to rise to 23% of by 2030, as Figure 1.4 shows.

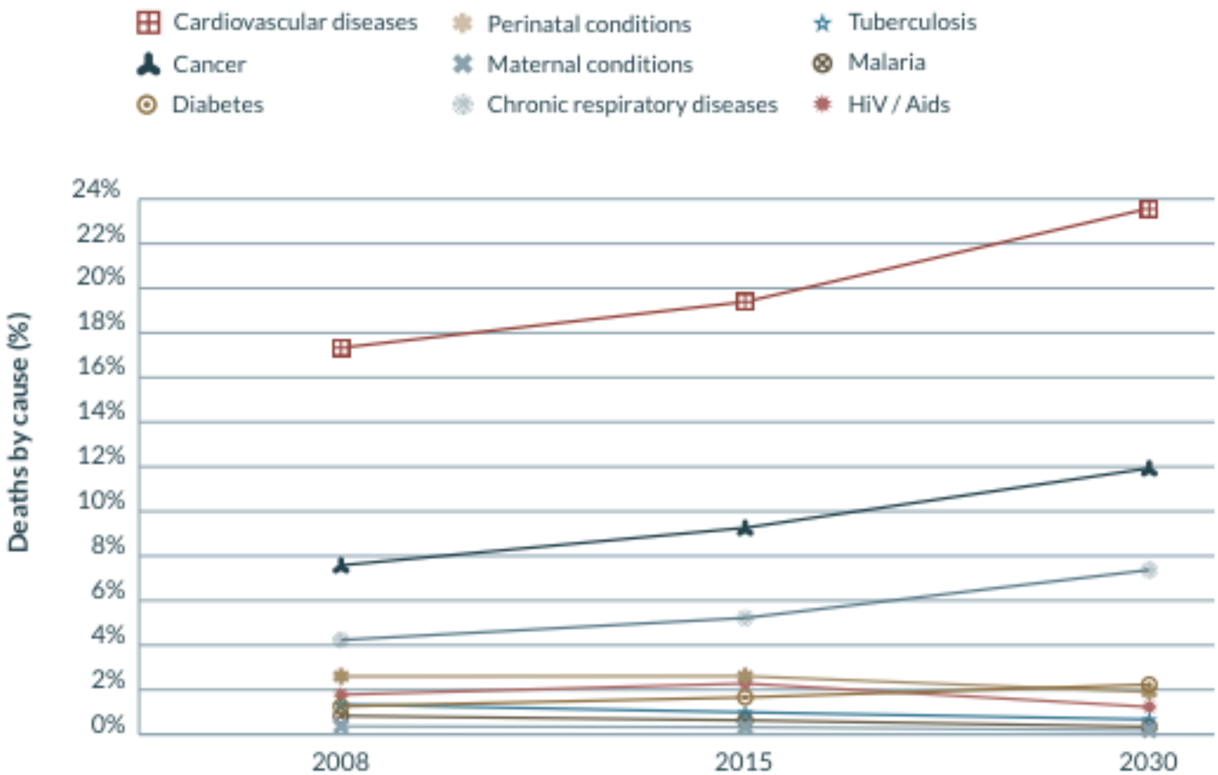


Figure 1.4: the projected mortality trend from 2008 to 2030. Source: (World Health Organization, 2004).

1.2 Motivation

As it has been already disclosed, blood pressure is a vital sign to asses the state of a patient. However, it is rarely measure in healthy patients. Continuous monitoring is only possible by invasive methods, which use catheters such as PiCCO (Pulse Contour Cardiac Output) systems. Ambulatory measurements are only punctual and can be miss leading if they are not done with proper training. In the current market there are no such device certified for continuous measure this vital sign in order to monitor patients diagnosticated with hypertension or other cardiovascular diseases.

1.3 Objectives

The aim of this project is to create a device capable of measuring the blood pressure from an individual and send the information to a remote device, where values could be analysed by a clinical expert. Additional goals are the creation of a wireless power delivery system, and an application to visualise and store the measured results.

These objectives can be divided in:

- **Hardware:**
  1. Creation of a device capable of measuring blood pressure.
  2. Design and implementation of a signal conditioning circuit.
  3. Integration of a microcontroller unit.
  4. Integration of a communication unit.
  5. Powering by means of a wireless power delivery system.
- **Software:**
  1. Generation of the microcontroller code.
  2. Creation of an app for receiving information.
  3. Design and implementation of visualisation tools inside the app.

1.4 Employed technologies

For achieving the aforementioned objectives, we will use the following technologies.

- Design Spark PCB - A Computer-Aided Design software for designing (schematics and layout) printed circuit boards (PCB).
- MBED compiler - An online operating system for programming ARM-based platforms. It provides libraries and APIs that can help us in our design.

- Matlab - A software framework for analysing data, developing algorithms, or creating models.
- Android Studio - A collection of tools for designing and building applications for Android devices.

1.5 Work procedure

The procedure for the thesis will be:

- **Study of art state**: documentation of the principles of hemodynamics and measurement of blood pressure from articles and books.
- **Design and prototyping for the desired system**: with all prior information a solution for the problem will be produced.
- **System testing and results**:Once the system is establish test will be establish and performed on the device in order to verify a correct behaviour of the device.
- **Thesis**: Documentation of all processes done and the redaction of conclusions.

1.6 Project structure

The project is divided in the next 6 parts

- **1. Introduction**: where the motivation and objectives for the work are manifested, as well as the working procedures to reach the goals.
- **2. Cardiovascular system background**: a brief introduction to all anatomical and physiological principles of the cardiovascular system.
- **3. State of art**: where the solutions and the principles of measuring blood pressure available in the market for the average consumer will be described.
- **4. Designed Solution**: in it the process of design, prototyping and development will be designed

- **5. Results**: where the methods for verification of the correct behaviour of the device will be establish and the result for those test will be displayed

- **6. Conclusions**: in it all conclusion for the work will be presented, as well as future lines of work.

2. Cardiovascular system background

The main functions of the cardiovascular system are gas transport, nutrient delivery, and waste removal. These functions are dependent on the continuous circulation of blood through the whole body, a process that requires the structural integrity of a complex system, mainly composed by the heart and vessels. The circulation is obtained thanks to the heart, which is an organ with 4 chambers that pumps the blood by a periodic sequence of contractions and relaxations. As aforementioned, the system is also conformed by blood vessels, a infrastructure which irrigate the cells along the whole body and guide the blood to and from the heart.

2.1 Heart physiology

Figure 2.1 shows that the human heart consists of four chambers: the two muscular ventricles and the two atria. The atria function is to allow a continuous circulation of blood to the heart. The ventricles function is to contract, increasing the pressure in the chamber, making blood flow out of the heart.

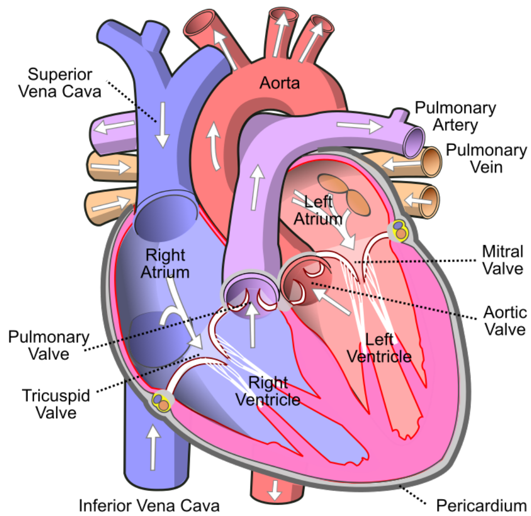


Figure 2.1: Human heart and its main parts. Source: (Wapcaplet, 2012).

In order to block the flow from the ventricle back to the atria or back to the ventricle from the arteries, there are also four one–way valves. These are the tricuspic valve, pulmonary valve, mitral valve, and the aortic valve, which block the circulation from the right ventricle to the right atria, from the pulmonary arteries to the right ventricle, from the left ventricle to the left atria, and from the systemic arteries back to the left ventricle, respectively.

When contracting, the left ventricle produces a higher pressure than the left ventricle. This is caused for the different ways that the ventricle retracts. The left ventricles contracts from the base to the apex of the chamber, while the right ventricle contracts in a medial fashion.

2.2 General scheme of blood circulation

Launched by the heart's activity, the blood circulation can be divided in two parts:

- **Pulmonary circulation**: which guides blood from the right ventricle of the heart to the lungs, where it is oxygenated, and back to the left atria of the heart.
- **Systemic circulation**: which guides blood from the left ventricle of the heart to all other parts of the body and back to the right atria, depleted of oxygen.

On the other hand, and grouped according to their functionality, the blood vessels can be also divided in three categories (Figure 2.2):

- **Arteries**: they take the blood from the heart to all parts of the body.
- **Veins**: they take back the blood to the heart from all over the body.
- **Capillary**: they connect the arteries to the veins.

2.3 Heart beat

The heart contracts at a normal rate of 70 times per minute, however the contraction follows a strict sequence in order to maximise the efficiency of every beat. Firstly, the sinoatrial node starts the beat by depolarising. The depolarisation marks the initiation of the atrial systole, in which the atria is compacted injecting as much blood as possible in the ventricle. When the atria finishes, the heart enters in the diastole phase, in which the blood flows back to the chambers and the cardiac tissues start relaxing. The ventricle starts its contraction, ventricular systole, pumping the blood out of the heart through the arteries.

2.4 Vessels physiology

The structures of arteries and veins include three tissue layers:

- **Tunica intima**: the most inner layer of the vessel. It consist of a single layer of endothelial cells.
- **Tunica media**: the layer consist in smooth muscle cell
- **Tunica adventitia**:the outer layer and it is formed of connective tissue.

The difference between arteries and veins, as well as between small and large vessels, comes down to the absolute and relative thickness of media and adventitia. Capillar only consists of a Tunica intima layer. Details can be seen at Figure 2.2.

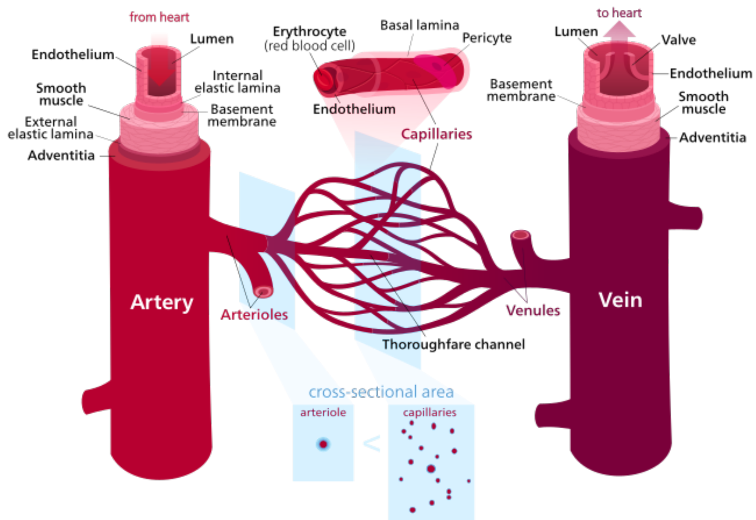


Figure 2.2: Blood vessels structures. Source: (Kelvinsong, 2006)

2.5 Hemodinamics

Hemodynamics is the study of the forces involved in circulation of blood. The pressure in all the cardiovascular system is not . There exist a gradient of pressure in the system creating a flow from high pressure areas to low pressure areas. The flow rate can be determined via Eq. (2.1).

$$Q = \frac{\Delta P}{R} \tag{2.1}$$

Where Q is the flow rate, ΔP the difference of pressure and R is the resistance of the vessels.The blood pressures in the venous systems of both pulmonary and systemic circulations are considerably lower than pressures in the respective arterial systems (Figure 2.3). In addition, blood pressures in the pulmonary circulation are lower than corresponding pressures in the systemic circulations. Systemic arterial pressure at rest is normally about 120/80 mm Hg, compared with 25/10 mm Hg for pulmonary artery pressure.

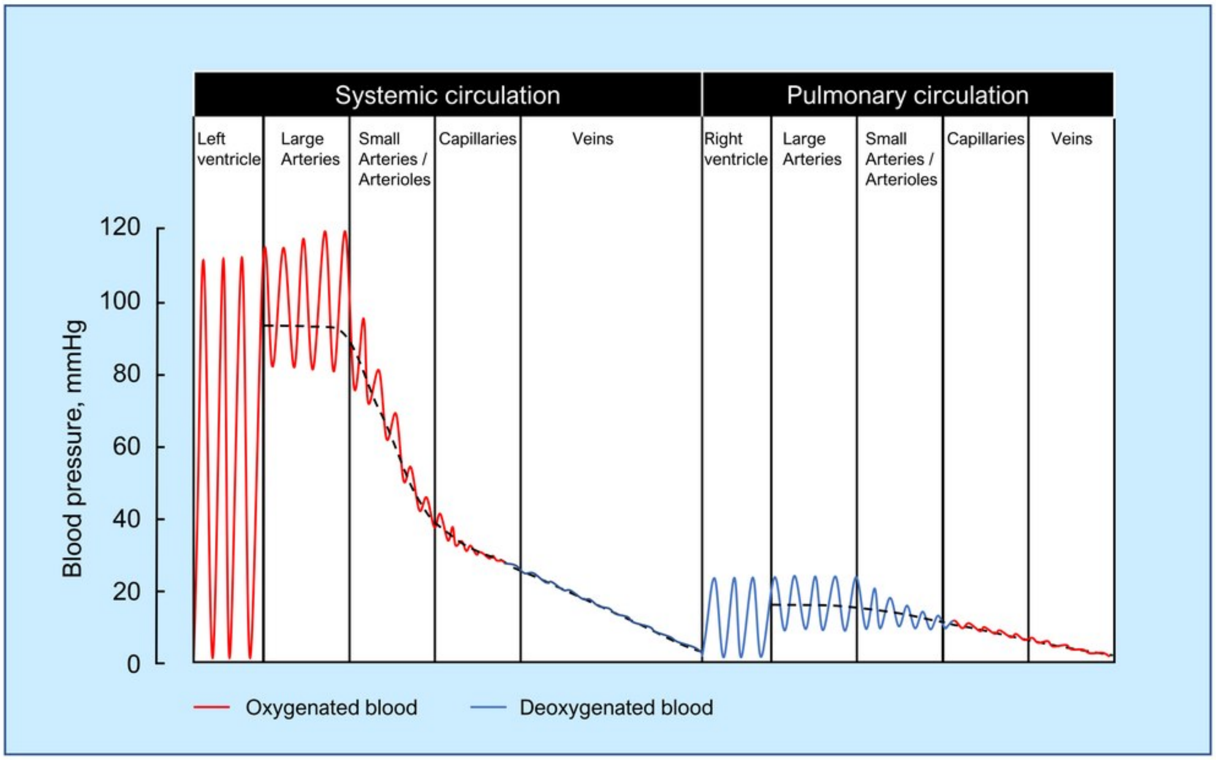


Figure 2.3: Blood pressure decrease in systemic and pulmonary arteries. Source: (Adh30, 2019)

The most typical measure of the blood pressure is the systemic arterial pressure. Three parameters are obtained:

- Systolic pressure: Peak arterial pressure reached during ejection of blood by the heart.
- Dyastolic pressure: Lowest arterial pressure reached during diastole.
- Mean Arterial Pressure: Although it may seem that it could be calculated via a simple arithmetic method, the form of the pressure wave forces to approximate it via Equation (2.2).

$$MAP = P_{diatolic} + \frac{P_{Sistolic} - P_{diatolic}}{3} \tag{2.2}$$

A normal arterial blood pressure wave is shown in Figure 2.4.

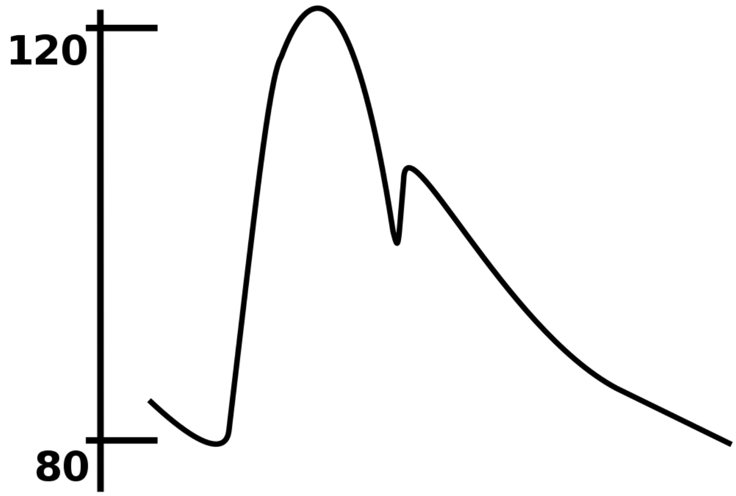


Figure 2.4: Normal blood pressure wave

3. State of art

The techniques used to measure blood pressure are divided into two groups, invasive and non invasive. Non invasive methods can be furthermore divided in two categories, depending on the use of an external pressure source or if they do not need it.

- Blood pressure meassuring methods
  - Invasive
    - -Extravascular
    - -Intravascular
  - Non invasive
    - -With external pressure
      - +Sphygmomanometry
      - +Tonometry
      - +Volume Clamp
    - -Without external pressure
      - +Pulse Transit Time

3.1 Invasive methods

It consists of inserting a saline filled catheter through the patient's vascular system to the point at which it is desired to perform the measurements. The catheter is connected to an electronic pressure transducer, which measures the pressure in the artery. The advantage of this system is that pressure is constantly monitored beat-by-beat, and a waveform can be displayed. What is also remarkable from this method is that it provides the most accurate arterial pressure reading from patients.

3.2 Non invasive methods

3.2.1 Sphygmomanometry

The most common approach for measuring blood pressure until the present day is the sphygmomanometric one. In this approach, arterial pressure is measured by a sphygmomanometer (see Figure 3.1), which consists of:

1. A manometer
2. A bracelet with a an inflatable bag
3. A pump
4. A deflating air valve

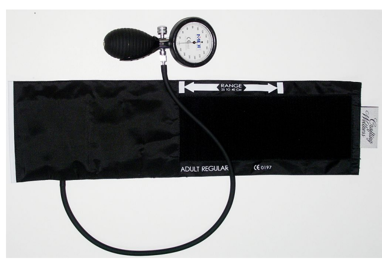


Figure 3.1: Sphygmomanometer.

In order to measure the pressure the cuff is inflated choking the arm on which is placed. There are different techniques to determinate the diastolic and systolic pressure values.

Palpatory

After the bracelet is placed snugly over the arm, the radial artery is palpated while the pressure is increased until the pulse can no longer be felt. Then it is over-inflated 30-40 $mmHg$ and the valve is slowly open until the pulse is felt again, marking the systolic pressure. No systolic pressure is measured by this method.

Auscultatory

This method is based on the Korotkoff noises, and requires an additional stethoscope. After the bracelet is placed it is inflated until no sounds can be hear. Then the cuff is slowly deflated. After the first two audible noises are heard, the sistolic pressure is marked. While continuing the deinflation, the noise will become more clear and intense, until a significant decrees is produced. In children, this situation marks the diastolic pressure. For adult people, when the noise is not audible the diastolic pressure is measured.

Oscillometric

The oscillometric method follows a process similar to the auscultatory, however instead of hearing noises, an electronic pressure sensor is placed in the bracelet that senses the magnitude of oscillations caused by the blood as it begins to flow again into the limb after being choked.

3.2.2 Tonometry

A force is applied on the artery to flatten it out against the bone. The force applied is recorded resulting on a similar waveform as invasive methods.

3.2.3 Volume Clamp

This method is based on a finger cuff, a photopletismographer, and a pressure controller unit. The technique keeps a constant volume of blood in the finger by a force applied by the finger cuff, that represents the blood pressure (Figure 3.2).

3.2.4 Pulse Transit Time

The method measures the time it takes for the pressure wave to arrive to a site of peripheral circulation, such as a finger or an earlobe.

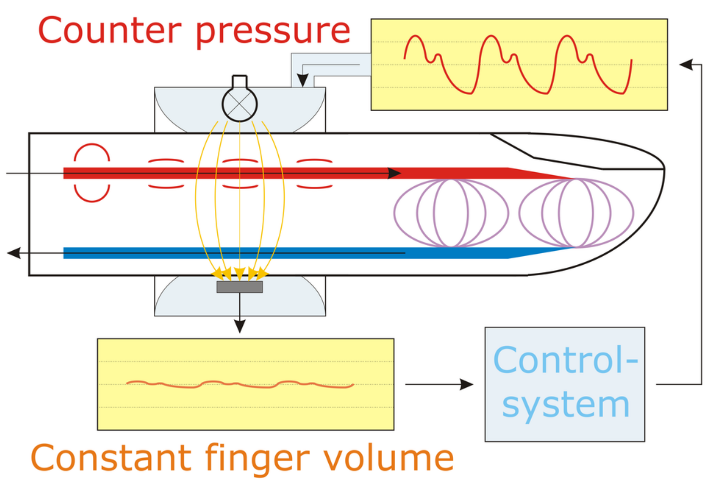


Figure 3.2: Volume clam scheme. Source: (ProfBondi, 2012).

4. Designed Solution

In order to solve the problem of measuring the blood pressure, two goals must be reached. Firstly, we should design and build a hardware device capable of measuring the pressure in the required scenario. The proposal must be able to send the data to a remote device. And secondly, we need to design and develop the source code for capturing the data and sending this information to an app on the remote device, where data can be visualised.

4.1 Hardware

Figure 4.1 provides an overview of the proposed device. Basically, the collection of blocks on the architecture are:

- Sensor
- Microprocessor
- Conditioning circuit
- Bluetooth module
- Power Module

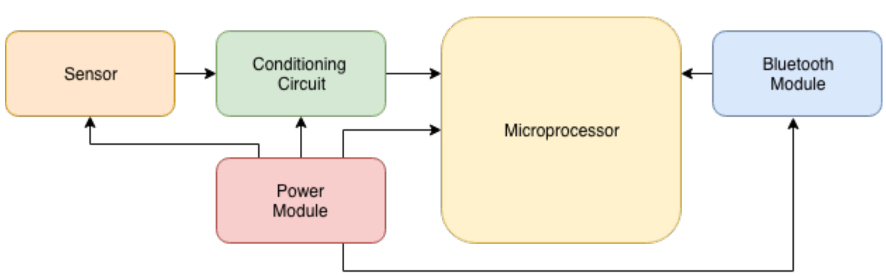


Figure 4.1: Device diagram.

Section 4.1.1 describes the features driving the selection and/or design of each module on the diagram. For designing the whole device, we will follow an incremental approach. That is, once each part is selected, it will be individually tested and then connected to the device. The whole device will be then tested, considering its extended functionality. For the case of the power module, we will initially power the device from an external source. Then, when we will have our power module, we will remove the external source and test if the whole device can be powered from this module.

4.1.1 Selection and design

Sensor

The first and one of the most important choices was the selection of the sensor for such a device. There exist many sensors that can measure pressure, however the first consideration to take is the pressure range they can measure. Normally aortic blood pressure (the greatest in the body) stays around 60-140 $mmHg$, but in persons with hypertension it can go as high as 300 $mmHg$. The problem is that in the market there are no available sensors that can sense from 0 to 300 $mmHg$, at least explicitly, so we have to convert it to other units for addressing this searching. Table 4.1 provides this conversion or our highest value.

Unit	**Value**
$mmHg$	300
$torr$	300
mmH_2O	4078.52
Pa	39996.7
psi	5.80103
atm	0.394737
bar	0.399967

Table 4.1: Conversion table for 300 $mmHg$ to other units

The next consideration to make is that the body pressure is relative to atmospheric pressure. Pressure sensors are commonly divided in gauge, that compares to atmospheric pressure; differential, that compares two different entrances; or absolute, that compares to vacuum. Due to the nature of blood pressure, gauge is the most adequate, but others can be used if well considered.

One last consideration, that was overlook is the resistance to ionic fluids, like water. When the catheter is placed inside the artery, a force is applied on the wall due to the difference in pressure. If the catheter is filled with water, the volume cannot be changed and transmitted to the transducer. If the catheter is filled with air, the volume inside is changed: however the walls of the catheter will oppose to the change, dampening the signal.

Our options are summarised at Table 4.2.

Sensor	Scale	Output	Type
1210A-005D-3L	$0 - 5psi$	$0 - 100mV$	Diferential
MPRLS0015PA0000SAB	$0 - 15psi$	Serial Peripheral Interface	Absolute
DLC-L60G	$0 - 60inH_2O$	24-bit digital output	Gauge
NPC-1210-005G-3L	$0 - 5psi$	$0 - 100mV$	Gauge

Table 4.2: Comparison among sensors

The best fit for our task is the NPC-1210-005G-3L (Figure 4.2), which covers all our necessities for the sensor.

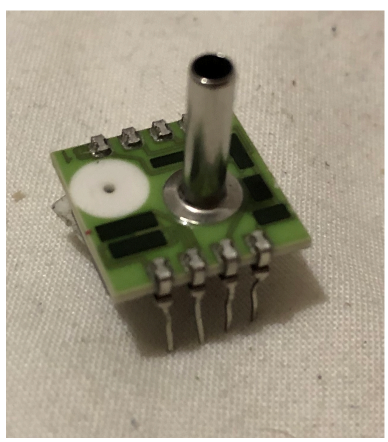


Figure 4.2: Sensor NPC-1210-005G-3L.

Development board

To simplify the process of adding a microcontroller to our proposal, avoiding the design of a specific printed circuit board and the assembly of all circuits and passive elements, we will build our system over a development board. The requirements for the board are to include a microcontroller, with options for analogic to digital conversion, low-power consumption, and a size as small as possible. Some options were studied. Table 4.3 provides a brief description of them.

Board	Analog to digital pins	Microprocessor	Power consumption	Size
Arduino Nano	8	ATmega328	19 mA	18 x 45 mm
Tiny BLE	4	nRF51822	Not disclosed	43.3 x 29.0 mm
Adafruit Trinket	3	ATSAMD21E18	Not disclosed	27mm x 15.3mm
Apollo3 Blue	No	ARM Cortex-M4F	1.6 mA	40.6mm x 40.6mm

Table 4.3: Comparison between development boards

All of these options cover our needs. However, the Tiny BLE comes with the nRF51822 chip of Nordic, which includes a transceptor able of managing Bluetooth Low Energy (BLE). It also includes a AN2051 antenna. This avoids us to include an external antenna and a wireless module. Moreover, it ensures the low energy consumption, a mode that can be easily monitored via software (libraries and APIs are available). It supports ARM mbed cloud-based IDE with an handy C/C++ SDK. Figure 4.3 shows how the Tiny BLE is organized. The board is divided up into two parts, the one of the left is built around a LPC11U35FHI33 microcontroller, whose aim is to provide a CMSIS DAP, i.e. a standardised way to access the Coresight Debug Access Port (DAP) of the nRF51822, an ARM Cortex M via USB. It allows the drag-n-drop programming of the nRF51822. This chip consitutes the core of the second part of the Tiny BLE (the one on the right). If it is needed (when you have your final program on the nRF), you can cut the board and only including on your prototype this second part.

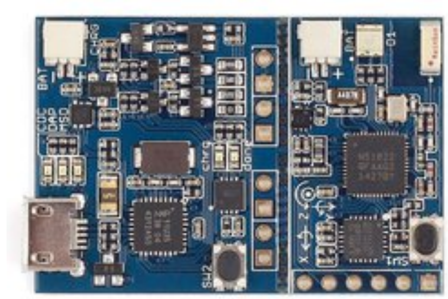


Figure 4.3: Tiny BLE development board

Conditioning Circuit

The NPC-1210-005G-3L sensor provides an analog output, which must be correctly captured to avoid noise. It must be then included within a conditioning circuit. In the datasheet of the sensor, Figure 4.4 illustrates how to design this circuit, using two operational amplifiers (opamp). As it is shown on the figure, this would provide a full scale output (FSO) of 3 Volts. If we analyse the output of the circuit, we can determine that the amplification is:

$$(N_+ - N_-) = (1 + 2\frac{R}{G_{ainSet}})(V_{Out+} - V_{Out-}) \tag{4.1}$$

However, the circuit gives a floating voltage, which is not referred to ground.

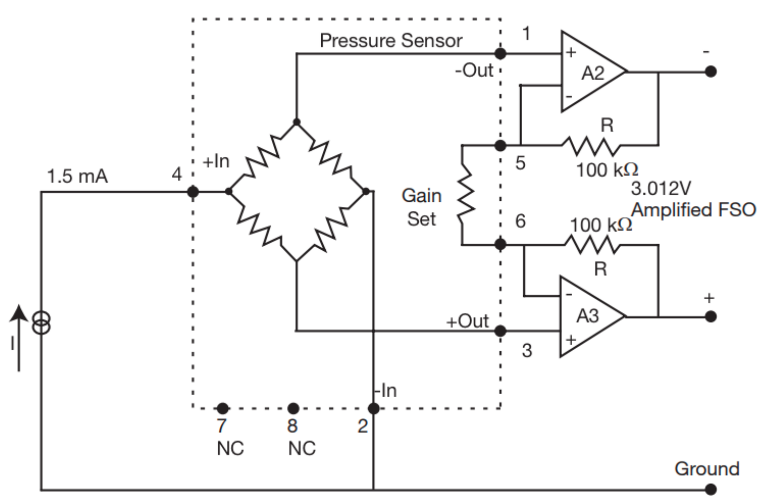


Figure 4.4: NPC-1210 series Schematic Diagram. Source: (*NovaSensor NPC-1210 Medium/Low Pressure*, 2017).

To solve this last inconvenience, another step can be added to the circuit, completing the basic scheme of a instrumentation amplifier (Figure 4.5).

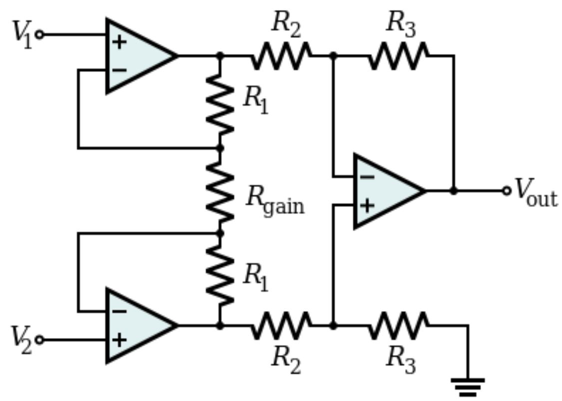


Figure 4.5: Typical instrumentation amplifier schematic

The output of this circuit would be:

$$V_{out} = \frac{R_3}{R_2}(1 + 2\frac{R_1}{R_{gain}})(V_{Out+} - V_{Out-}) \tag{4.2}$$

Where $V_1 = V_{Out-}$ and $V_2 = V_{Out+}$. This configuration is very convenient since R_1 is set to be $100K\Omega$ to have a FSO of $3V$, from the first step, we can give values to R_2 and R_3 to match the analog to digital converter from the Tiny BLE unit, that it is 3.3 V, so no additional amplification is needed, and R_2 will be equal value to R_3.

Choosing the opamp was really simple. One of the most used is the UA741, which presents great properties for general purpose. As a major problem, the amplifier is rated to work with a minimal power supply of $\pm 5V$, but the Tiny BLE can only give $0 - 5V$. We should solve this issue by adding a dc/dc converter.

The final schematic for the conditioning circuit is illustrated in Figure 4.6. We can see the three amplifiers and the sensor. Additional resistors are also present.

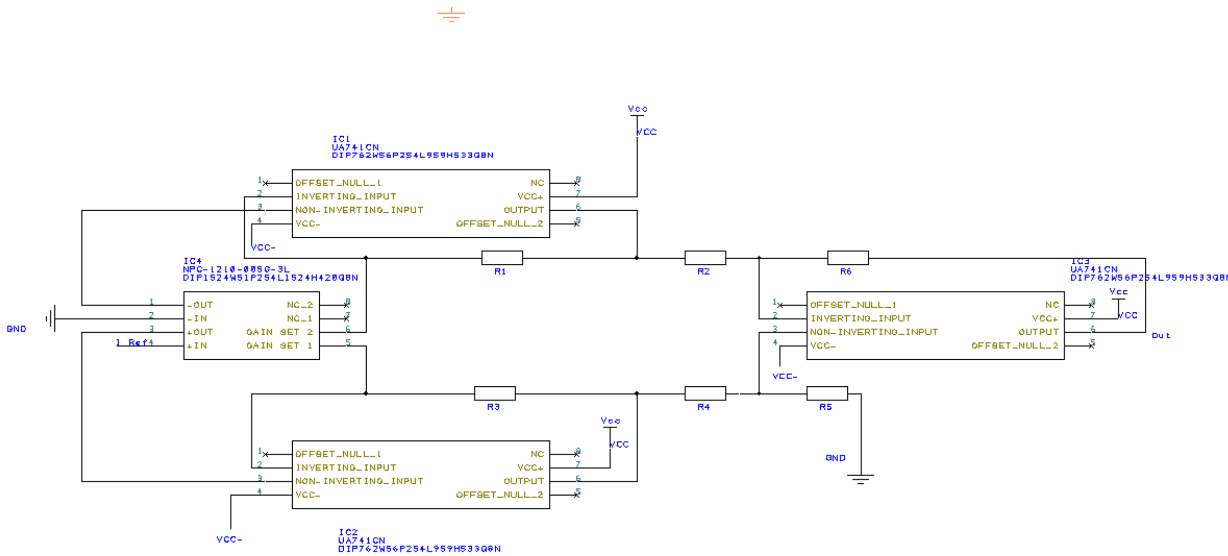


Figure 4.6: Final schematic for the conditioning circuit

Power delivery

One of the objectives of the device is to couple it with an external power supply, so the device does not need a battery, making it safer for the body. Designs were attempted using an old induction coil to create a resonant inductive coupling, however after reading literature about it was discarded, due to a lack of equipment. Instead a commercial solution was used. Figure 4.7 shows the acquired USB wireless charger receiver, attached to our system for providing power supply. The module is rated 5V and a maximum of 1 A, enough to power the Tiny BLE and the conditioning circuit.

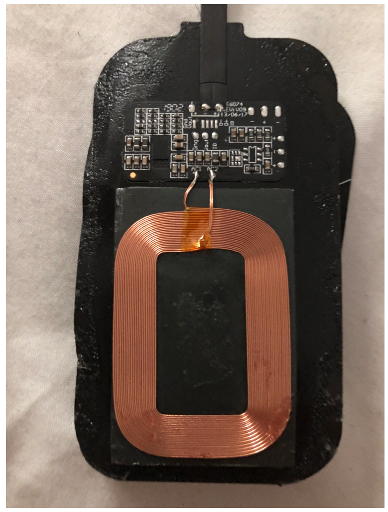


Figure 4.7: USB wireless charger receiver

However, as aforementioned, the opamp chosen need to be supplied with $\pm 5V$. For providing the negative voltage, an inverter is needed. The LT1054 was chosen for the task and configured as a basic voltage inverter (Figure 4.8).

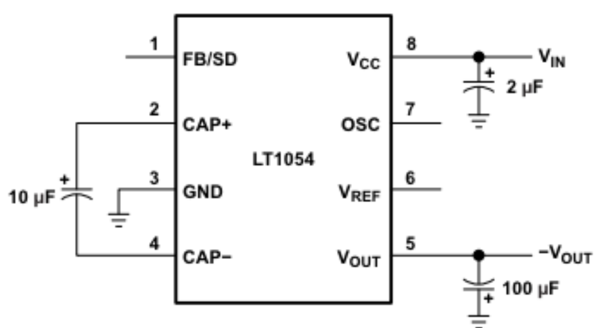


Figure 4.8: Basic Voltage inverter. Source: (Texas Instrument, 2015).

Figure 4.4 illustrates that the sensor also needs a 1.5 mA as reference (pin 4). This reference value can be obtained with an LM334 (Figure 4.9). With the information given in the schematic we can easily determine R_{set}, as it is shown in Equations (4.3) and (4.4).

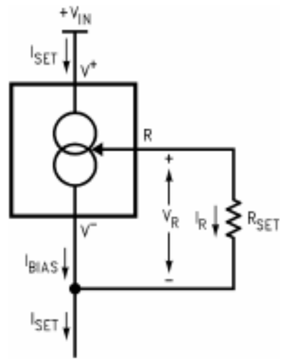


Figure 4.9: Basic Current Source. Source: (Texas Instrument, 2013)

$$I_{set} = \frac{67.7mV}{R_{set}} \tag{4.3}$$

$$R_{set} = \frac{67.7 * 10^{-3}}{1.5 * 10^{-3}} = 45.13\Omega \tag{4.4}$$

After adding the dc/dc converter and the current source, the final schematic for the circuit looks like Figure 4.10 shows.

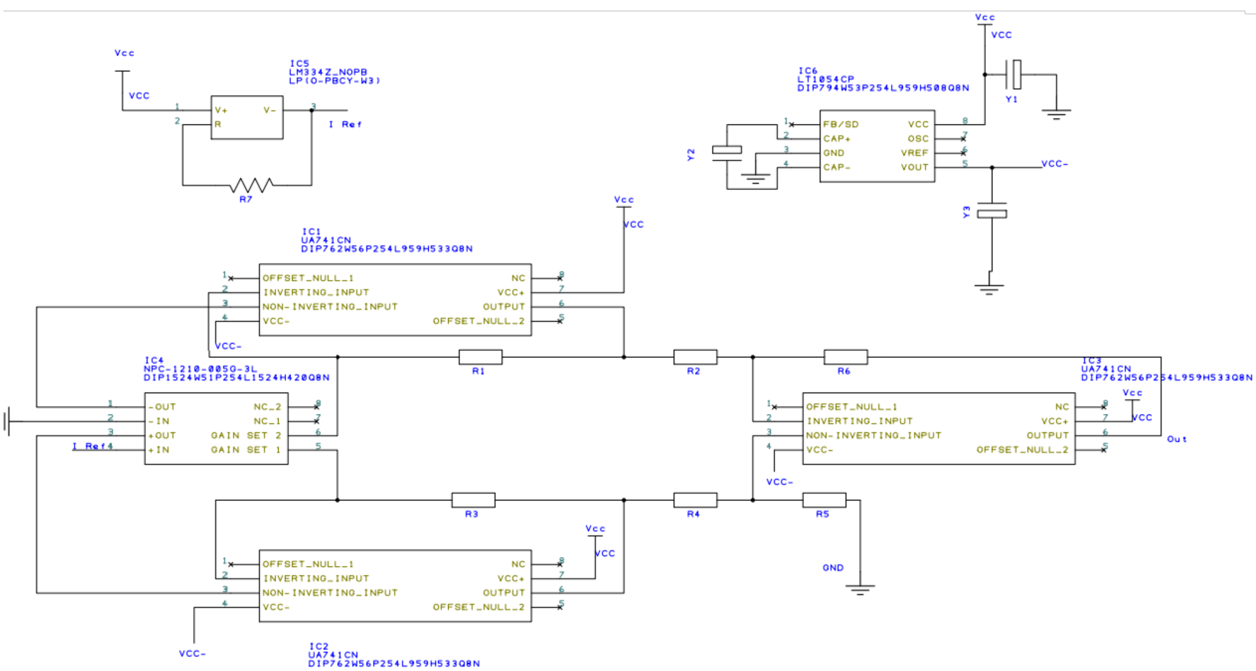


Figure 4.10: Complete schematic for the device

4.1.2 Prototype

Conditioning circuit

Although the conditioning circuit was not the first designed part that we close, it was the first prototype made, due to the sensor needing a current supply of 1.5mA, and the selection of a development board, instead of building one from the ground up.

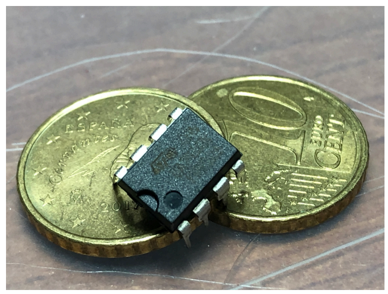


Figure 4.11: Picture of an opamp ua741

The final conditioning circuit consists in tree opamp configured as a basic instrumental amplifier, as it was shown in Figure 4.6. As aforementioned, we use UA741 opamps (Figure 4.11). If we build the circuit, it will look like Figure 4.12 shows. The circuit was verified by setting a sinusoidal wave between the V+ and V- inputs and verifying the correct behaviour of the circuit.

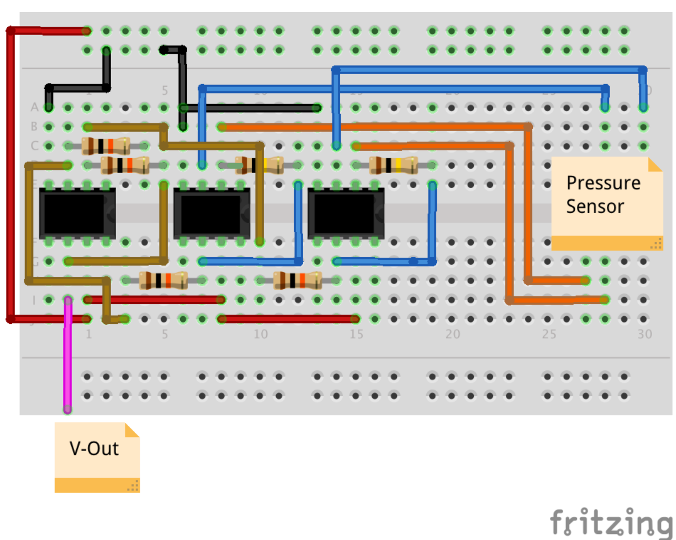


Figure 4.12: Conditioning circuit sketch

The output was verified by setting as the R_{gain} a $10K\Omega$ resistor and measuring the amplification, which should be around 21,00 V/V.

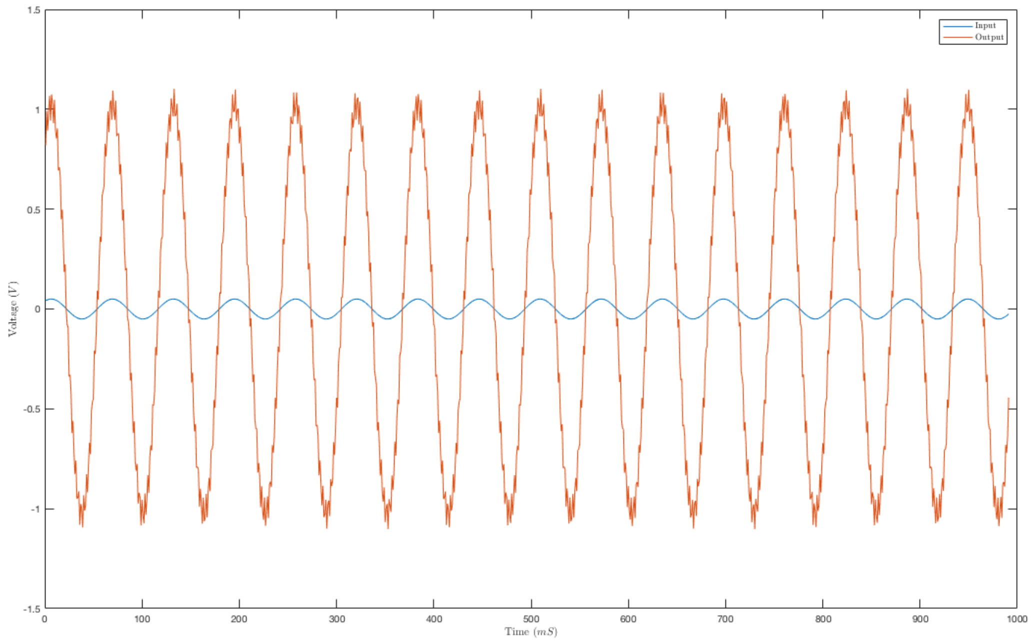


Figure 4.13: Output of the conditioning circuit to a sinusoidal input with amplitude $50mV$

Figure 4.13 shows the output for 50 mV sinusoidal wave (100 mV$_{p-p}$). The maximum output of the circuit was 2.05 V$_{p-p}$, so the amplification is 20,50 V/V. The result is as expected, verifying that the conditioning circuit works correctly.

Power module

The next step taken was to add the power module. The schematic to follow can be found in Figures 4.9 and 4.8. A schematic view of the connection on a proto-board is shown in Figure 4.14.

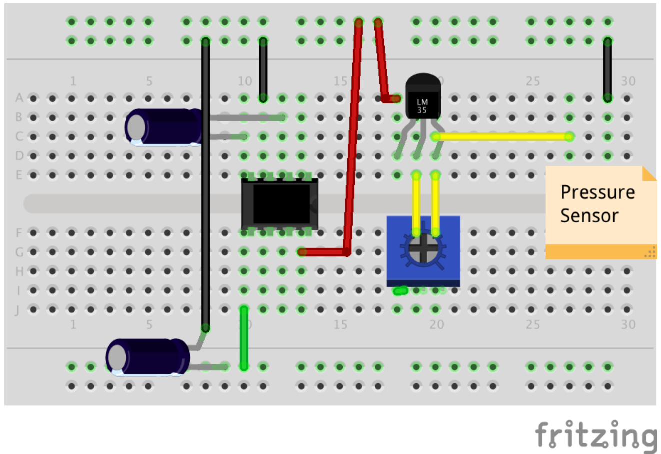


Figure 4.14: Power module Sketch

Final prototype

Once the power module was created, all parts were joined and tested. Figure 4.15 provides a snapshot of the real implementation. Conditioning circuit and power module were mounted on a proto-board. We could also see on the figure the Tiny-BLE and the charger receiver.

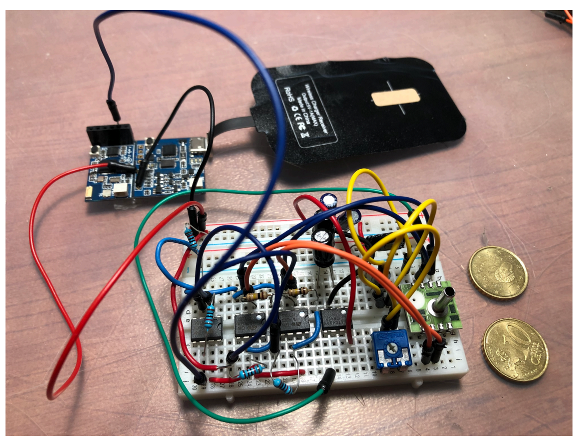


Figure 4.15: Final prototype

In order to verify the correct behaviour of this prototype, a simple code was developed. The aim is to capture the voltage sensed by the pressure sensor (amplified by our conditioning circuit) using the pin 3 of the Tiny BLE, and to send this value over Bluetooth. Details about software implementation will be provided in Section 4.2. For provoking a change on the pressure conditions, the test consisted on blowing gently on the sensor tube to see if a change in pressure could be sensed. However, this proved to be not enough force, so it was changed for a gentle tap on the top of the tube.

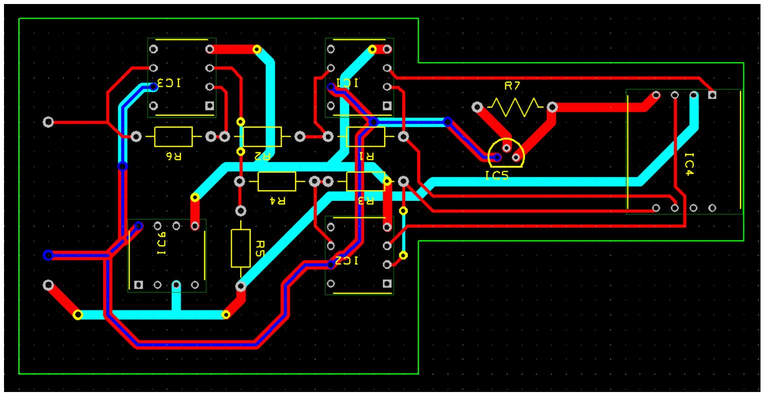


Figure 4.17: PCB design. The total dimensions are 98 x 48mm

Using an app from Nordic (nRF Connect), we verified that the output was correct (Figure 4.16). Thus, we have now a complete, working prototype.

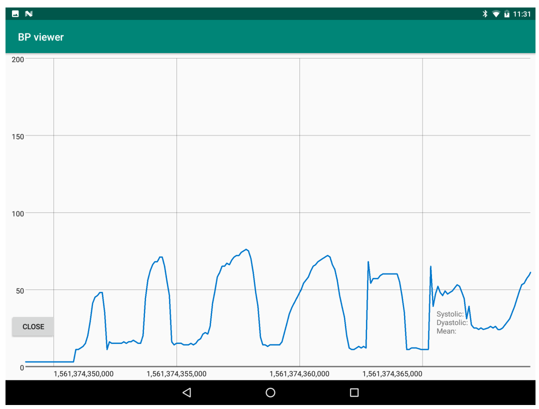


Figure 4.16: Results for the final verification test: the pressure values are visualized using an app on our smartphone

PCB

The final stage related to the hardware prototype was to design a printed circuit board (PCB), which allows us to substitute the proto-board for a more stable design. The schematic for this design was available (Figure 4.10) and we use DesignSpark for building the layout of the PCB. Figure 4.17 provides a snapshot of our design. We could reduce size with a more detailed design. Currently, the research group does not have at their disposal a workshop where building the board. Hence, and due to time-related reasons, we take the decision of do not produce it[1].

[1]There are fast available solutions for solving this problem (pcbway, expresspcb), but costs for fast delivery are excessively expensive.

4.2 Software

The whole software to be implemented can be divided in two blocks:

- C++ code for the microcontroller
- Android app

4.2.1 Tiny BLE

Measuring the wave pressure is not as simple as reading the value and sending it via Bluetooth. This allows us to verify that the system was correctly. But, in order to calculate the systolic and diastolic values, an algorithm must be implemented. Our code will be based in the algorithm described in (Ramírez, 2001).

The algorithm uses a digital derivative filter based on Equation (4.5), and updates a threshold value, Equation (4.6). The digital filter is updated with every new input value. When the output $y(n)$ surpasses the threshold value, the algorithm waits until the value of the filter is 0 or lower, and then it marks this point as the systolic pressure. Once the filter surpasses the 20% value, the diastolic pressure is recorded.

$$y(n) = 4x(n) + x(n-1) + x(n-2) - 4x(n-3) - x(n-4) - x(n-5) \tag{4.5}$$

$$U = U + 0.2(0.7F_{max} - U) \tag{4.6}$$

Therefore, the whole program on the nRF chip is divided in three sections:

- A main program for reading the last captured value and calculating the diastolic and systolic pressures.
- A class to create and handle all Bluetooth protocol calls.
- A definition for the Bluetooth service used.

main

The main program diagram flow can be found in Figure 4.18.

- First it creates all variables needed.
- Then it reads the value and updates the digital filter and the threshold value.
- If the filtered value is greater than the threshold value, then a systolic_event is triggered. When the filter value is lower or equal than 0, the systolic pressure is updated.
- If the value is lower than the threshold value times 0.2, then a diastolic_event is triggered. When the filter value is greater or equal than 0, the diastolic pressure is updated.
- All values are updated in the Bluetooth services.
- Wait for an event and go back to point 2.

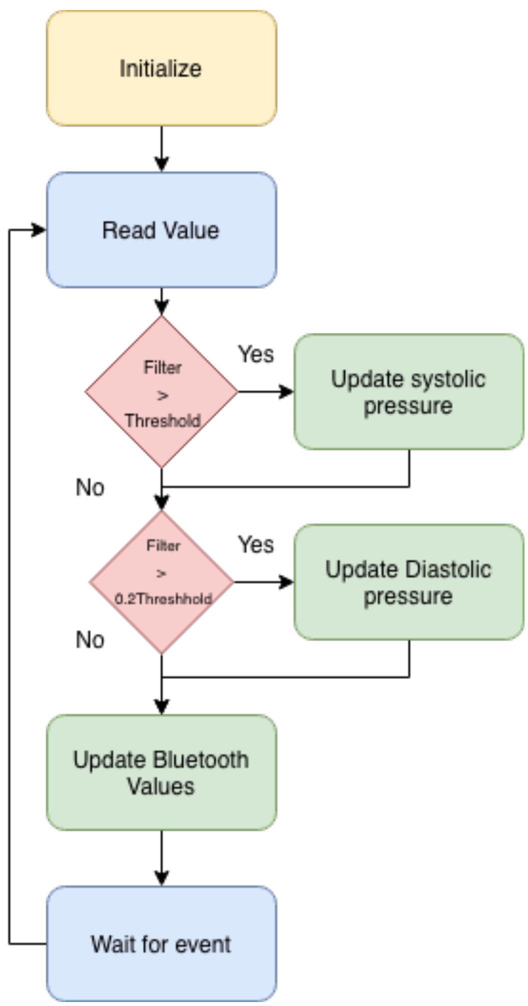


Figure 4.18: Main program diagram flow.

Source code can be found at Part II (Code).

ble_handeling

The BLE handling is created in order to control all Bluetooth related code. The two most important functions are *bleInitComplete*, which handles the creation and publication of the services offered by the device, *update_Pressure_BloodPressureStream* and *update_Pressure_BloodPressureValue*, which changes the values that are send over the Bluetooth services. Both methods are called over in the main loop when needed.

BloodPressureService

The Bluetooth service is based on an example application provided by the MBED Operating System, related to the standard BLE service *BloodPressure*. However, this service has been customized to include an extra attribute that sends the value read by the sensor. This allows us to plot the pressure wave on the remote device.

4.2.2 Android application

The android app functions are three: (i) providing a visualiser for the pressure wave; (ii) receiving the information coming from the device; and (ii) storing these values for later use. In order to create the app for all Bluetooth protocol–related topics, we have worked with the BLE developer guide (*Bluetooth Developer Starter Kit 5.0*, 2019). The graphical plotter has been developed using a publicly available library (Gehring, 2019).

The app flow diagram can be found in Figure 4.19.

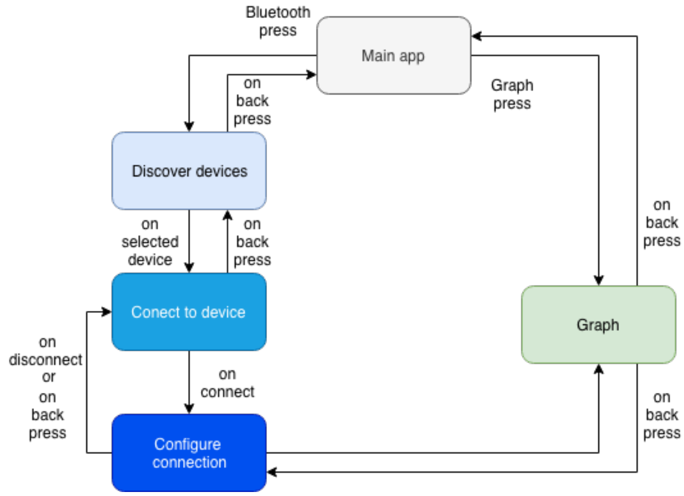


Figure 4.19: Blood Pressure (BP) app flow diagram

Main app

Figure 4.20: Main app activity

The main app is a select screen, where we can choose between Device connection or to visualise the graph associated to previously captured data.

Discover devices

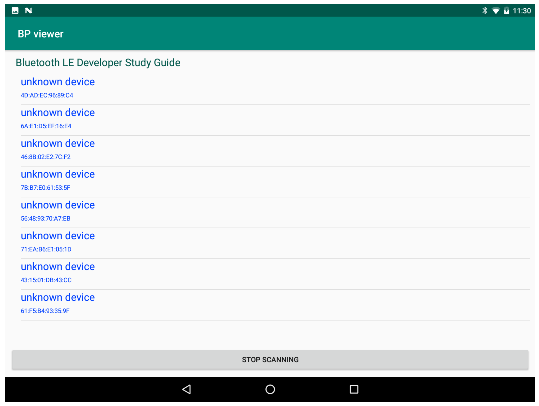


Figure 4.21: Discover service activity

The 'Discover service activity' has a button on the bottom to start a scan for Bluetooth devices, and to display them in a list in the middle of the screen. Once we press on the desired device, the screen 'Connect to device' will be prompted with all the needed information.

Connect to device

Figure 4.22: Connect to device activity

Once the app is connected with a device, the information related to the connection will be displayed at the top, as well as some inactive buttons. In the bottom, a CONNECT button (and a text on top of it) allows us to asses the state of the connection.

Configure connection

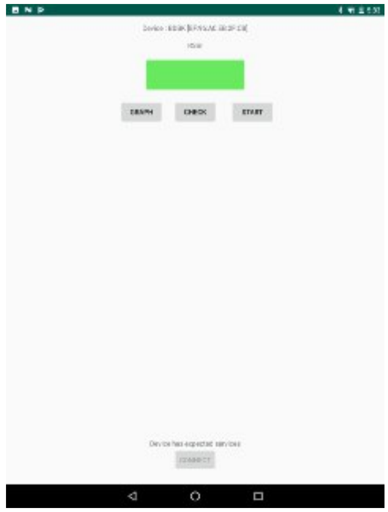


Figure 4.23: Configured connection activity.

Once connected, the inactive buttons will be enabled. The right one provides a stream of data between the device and the app. The middle one checks the services provided by the device. The left one prompts the graph app, where the pressure wave can be visualise in real-time.

Graph

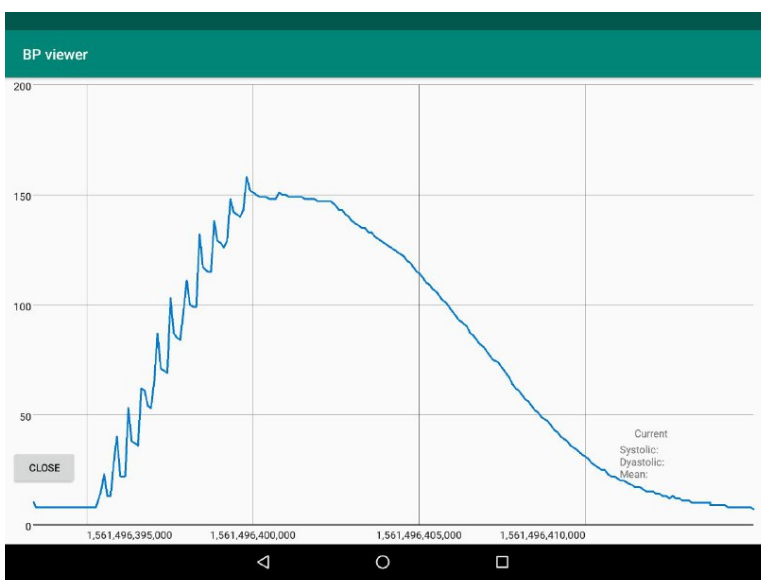


Figure 4.24: Graph activity

The final activity is the graph where the data received is plotted. Besides, in the bottom right corner, the normal values for the service are displayed. The button on the left (CLOSE) restores to the last activity used.

5. Results

This Chapter reviews the tests addressed to validate the correct behaviour of our prototype. In order to test the device, two different test were performed:

- A static test, where the device was tested to measure pressure in static fluids.

- A dynamic test, where The device was tested to measure pressure in a dynamic flow.

To conduct the test a simple contraction has to be created. Figure 5.1(left) shows the device built to test a static response. It is composed by a tube filled with water and, on top, an air pump and a manometer. Basically, the idea is to maintain a constant pressure value. We have at our disposal a sphygmomanometer, which can be an alternative for conducting this test. Figure 5.1(right) illustrates the same device with a severe modification: a water pump has been added to create a flow of water.

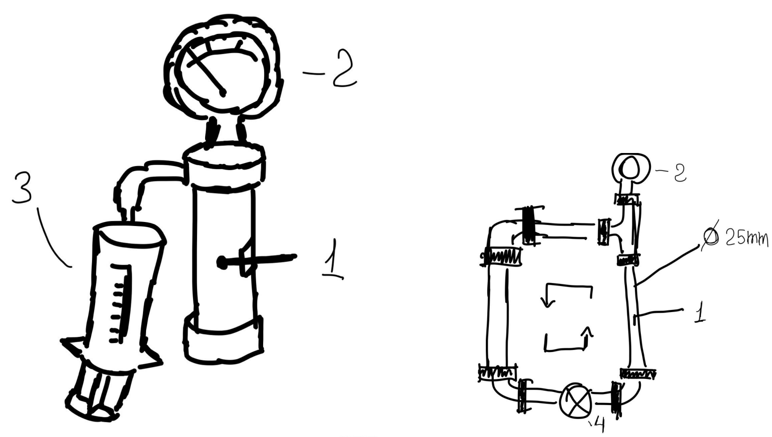


Figure 5.1: Caption

It must be noted that the device requires a previous calibration step. It is described in Section 5.1.

5.1 Calibration

The datasheet for the sensor does not give a direct way to convert the output voltage to a pressure value. It only specifies that the output of the sensor is typically 100 mV, but it can range from 75 mV to 150 mV, so just a simple transformation (see Equation (5.1)) is not correct.

$$P_{ressure} = \frac{O_{utput}}{3012} * 258.575 \tag{5.1}$$

Instead the sensor was connected to an sphygmomanometer and the output was recorded, as well as the pressure value measured by the manometer. Table 5.1 provides output values for five series and the pressure values provided by the sphygmomanometer.

Pressure	20	40	60	80	90	100	110	120	140	160	180	220	260
Series 1	37	89	139	187	203	232	253	278	302	375	430	424	610
Series 2	36	85	136	186	204	228	251	278	329	373	423	512	616
Series 3	35	88	136	184	202	229	256	280	323	371	425	517	623
Series 4	32	82	136	183	210	230	253	276	311	372	425	518	620
Series 5	32	83	135	182	205	234	257	281	324	372	428	520	610

Table 5.1: Pressure values (20 to 260) and measured output values

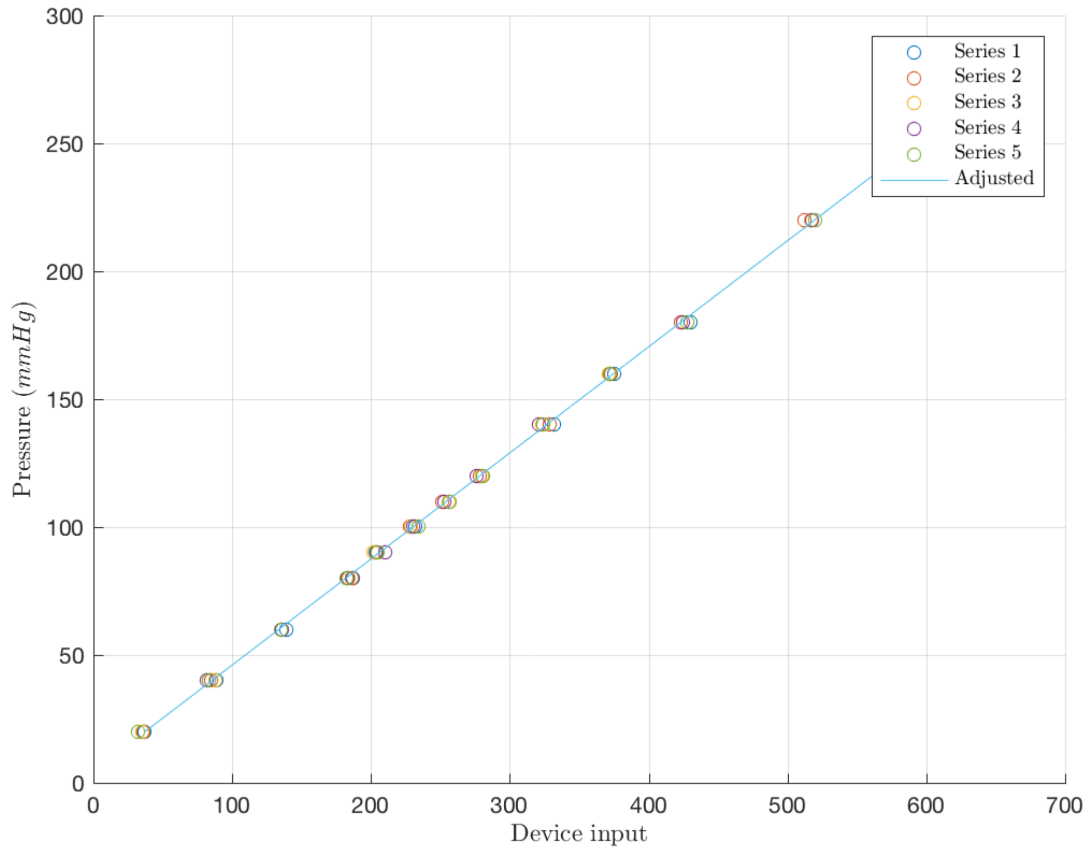


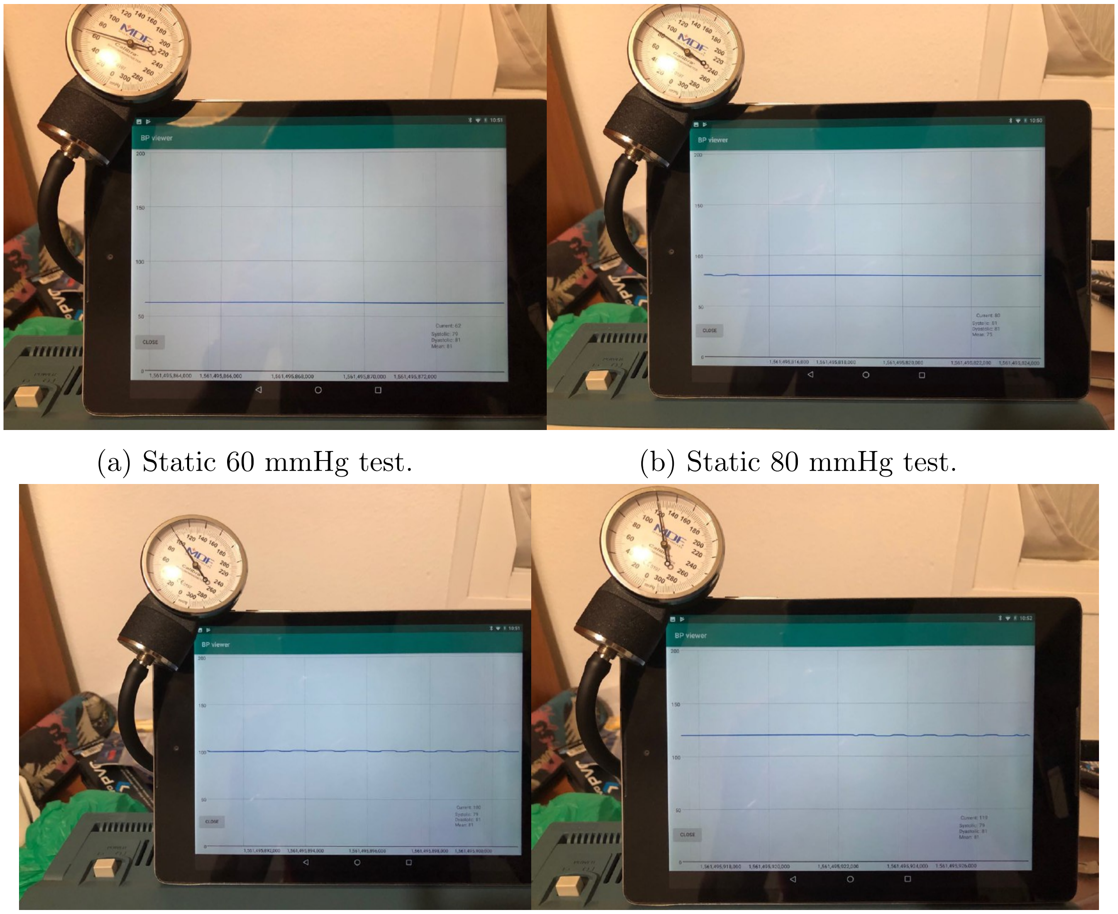
Figure 5.2: Relationship between the values measured in the Table 5.1 and the adjusted equation (see text for details)

With this information, and using Matlab, we can fit the device output to the pressure values (Figure 5.2). A first degree polynomial (Equation (5.2)) was used to fit a line to the set of points. Figure shows that the line correctly fits all obtained points.

$$P_{ressure} = 4.5517 I_n + 4.55 \tag{5.2}$$

5.2 Static pressure

As aforementioned, a sphygmomanometer was used to provide a static pressure. Thus, the pressure inside the manometer is set to a certain pressure an maintained for 3 seconds. For the test to be successful, the output must be held constant within a ±3 Hg margin of error. The pressures chosen to be test are from 60 to 120, since they are the most common values.



(a) Static 60 mmHg test. (b) Static 80 mmHg test.

(c) Static 100 mmHg test. (d) Static 120 mmHg test.

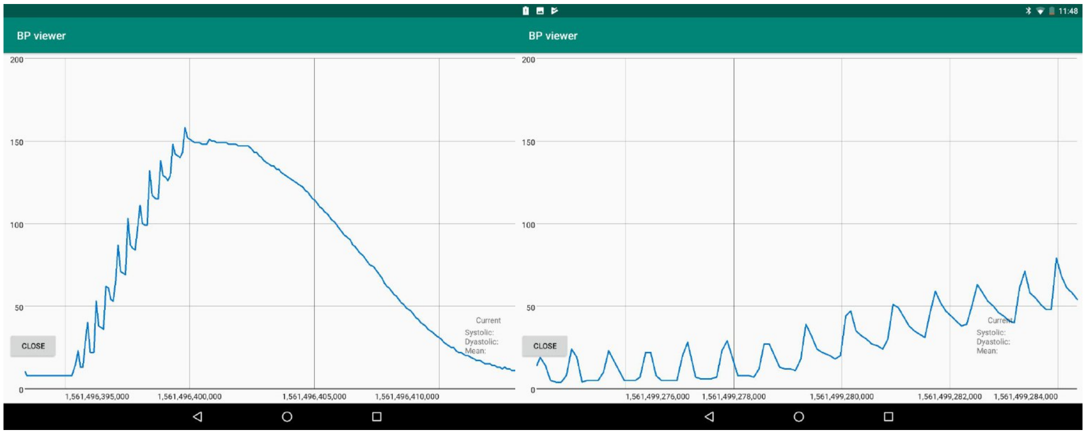
Figure 5.3: All static tests results.

As we can see in Figures 5.3, all tests were successful. Using the manometer as a reference value, we check that the device can capture a correct static pressure.

5.3 Dynamic pressure

For the dynamic two test will be done. In the first one, the sphygmomanometer will be as fast as possible filled to 150 mmHg, then the valve will be slowly open until depleted. The second test will held the valve open.

The result for the test one can be seen in Figure 5.4(a). The result is exactly as expected. The peak value is 150. When the valve is open, this value firstly decreases rapidly and then the decreasing rate slows down until the atmospheric pressure is recovered. When inflating the bracelet, we can see peaks of pressure, suggesting that the sensor also can capture the



(a) Dynamic test 1.

(b) Dynamic test 2.

Figure 5.4: Dynamic tests results.

increase in pressure due to the fluid flowing through the tubes. Figure 5.4(b) confirms the expected behaviour: the valve is open and the pressure values shows some peaks. If the sensor could not capture the flow, these peaks will not appear.

6. Conclusions and future work

Although technology for continuous pressure monitoring exists in the market, the goal of developing a low-cost, commercial solution for continuous invasive measurement is a segment to be filled. The proposed system can be categorized as the first draft of an electronic sphygmomanometer, which could be used for auscultatory methods. Obviously, it is only a prototype and serious, further testing must be performed in order to verify the correct behaviour of the device. These evaluations will surely carry out redesigning steps, in the hardware and software parts.

Moreover, in order to enable its subcutaneous use, the device should be shrink as much as possible, and covered with polydimethylsiloxane (PDMS) as considered by (Song et al., 2016). This will allow us to make it biocompatible. Also a critical part of the device is the catheter, which will be inserted into the blood vessels. This catheter should have flexible walls in order to maintain the signal as close as possible to the real one.

Other invasive method that could infer the value of the pressure wave could be the deformation of the blood vessels wall, since they are constantly deforming due to the pressures wave. Sphygmomanometry uses the same principle, however they collapse the vessel and measure the force used, instead of the expansion of the walls.

6.1 Future work

Future work should focus on building a extended printed circuit board, including the analog part, but also the nRF chip and all its surrounding electronics (antenna, resistors, quartz clocks,...). These two parts (analog and digital ones) can share the same board if we take into account the required restrictions. The device will be connected to the patient, so specific security restrictions should be considered in the signal conditioning circuit.

Once a new version of the hardware is available, a complete validation stage should be conducted. A major effort should be done in developing a system for emulating the behaviour of a real blood flow, circulating within a major artery. This should allow us to evaluate the algorithm for measuring systolic and diastolic values. Finally, the effort for moving the device to the evaluation on real patients should be clearly an ending stage, but it is also out of the scope of the academic framework where this work is presented.

Conclusiones y futuras líneas de trabajo

Aunque la tecnología para una monitorización ambulatoria existe en el mercado, el desarrollo de una solución comercial para la medición invasiva es un mercado a explotar. El sistema propuesto puede ser categorizado como un primer boceto de un esfigmómetro electrónico, que podría ser usado para métodos de auscultación. Obviamente el trabajo realizado es únicamente un prototipo y un testeo más profundo y estricto debe ser realizado para verificar el correcto funcionamiento tanto del software como del hardware.

Más allá, para habilitar el aparato para uso subcutáneo, el dispositivo debe ser reducido lo máximo posible, y recubrirse con polidimetilsiloxano (PDMS) como sugiere (Song et al., 2016). Esto nos permitirá hacerlo biocompatible. A de más una parte critica del dispositivo es el catéter que se emplee para introducirse en los vasos sanguíneos. El catéter ha de tener unas paredes flexibles para mantener la señal lo más parecido posible a la real.

Futuras líneas de trabajo

La continuación del trabajo se debe concentrar en la creación de un circuito impreso, que incluya la parte analógica, pero además el chip nRF, y todos los componentes necesarios para integrarlo (antena, resistor, reguladores...). Estas dos partes (la analógico y la digital) pueden compartir la misma placa siempre y cuando se tengan en cuenta las restricciones necesarias. El dispositivo estará conectado a un paciente, así que restricciones de seguridad especificas deberán ser consideradas en el circuito de acondicionamiento.

Una vez una nueva versión del hardware este disponible, un completo proceso de certificación deberá ser realizado. Un en el desarrollo de un sistema de emulación del comportamiento sanguíneo, circulando en una de las grandes arterias. Finalmente, la utilización del dispositivo para la evaluación en paciente reales debe ser la meta de este, pero este ya queda fuera del entorno académico del trabajo.

References

Adh30. (2019). *Circulation pressures.* Retrieved from https://commons.wikimedia.org/wiki/File:Circulation_pressures_v1.tif ([CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0/)])

Bluetooth developer starter kit 5.0. (2019). Retrieved from https://www.bluetooth.com/blog/bluetooth-developer-starter-kit-5-0/

Gehring, J. (2019). *Graphview.* Retrieved from https://github.com/jjoe64/GraphView/wiki

Kelvinsong. (2006). *Blood vessels.* Retrieved from https://commons.wikimedia.org/wiki/File:Blood_vessels_(retouched)_-en.svg ([CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0/)])

McGee, S. (2016). *Evidence-based physical diagnosis e-book.* Elsevier Health Sciences. Retrieved from https://books.google.es/books?id=k_sTDgAAQBAJ

Novasensor npc-1210 medium/low pressure. (2017). Retrieved from https://www.mouser.es/datasheet/2/18/AAS-920-303D-NovaSensor-NPC-1210-110117-web-850296.pdf

ProfBondi. (2012). *Vascaular unloading.* Retrieved from https://commons.wikimedia.org/wiki/File:Vascaular_unloading.png ([CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0/)])

Ramírez, C. A. (2001). Algoritmo para el cálculo de la presion sistolica y diastolica en el ventrículo izquierdo. In *Memorias ii congreso latinoamericano de ingeniería biomédica, habana, cuba.*

Texas Instrument. (2013). *Lm134/lm234/lm334 3-terminal adjustable current sources.* Retrieved from http://www.ti.com/lit/ds/symlink/lm334.pdf

Texas Instrument. (2015). *Lt1054 switched-capacitor voltage converters with regulators.* Retrieved from http://www.ti.com/lit/ds/symlink/lt1054.pdf

World Health Organization. (2004). *The global burden of disease: 2004 update.*

World Health Organization. (2013). *A global brief on hypertension.*

Song, K., Han, J. H., Lim, T., Kim, N., Shin, S., Kim, J., ... others (2016). Subdermal flexible solar cell arrays for powering medical electronic implants. *Advanced healthcare* *materials*, *5*(13), 1572–1580.

Wapcaplet. (2012). *Diagram of the human heart.* Retrieved from https://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg (Wapcaplet [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0/)])

I

Appendix

A. Tiny BLE app

main.cpp

```
#include "mbed.h"
#include "ble/BLE.h"
#include "BLE_handling.h"
#include "tiny_ble.h"

DigitalOut red(LED_RED);                // RED LED
DigitalOut green(LED_GREEN);            // GREEN LED
DigitalOut blue(LED_BLUE);              // BLUE LED
AnalogIn Pressure(PRESSURE_IN);

uint16_t Parray[6];
int16_t filter=0;
uint16_t Umbral=0;
uint16_t maxvalue=0;

uint16_t SystolicValue = 0x0000; //  mmHg
uint16_t DiastolicValue = 0x0000; //
uint16_t MeanArterialValue = 55.0f; //

volatile uint8_t periodic_event = 0;
volatile uint8_t systolicpressure_event = 0;
volatile uint8_t diastolicpressure_event = 0;
volatile uint8_t UpdateValue_event = 0;

void periodic(void)
{
```

```
    periodic_event = 1;
}
int32_t vintoTorr(int32_t vin){
    //return vin;
    return ((vin*0.4186+4.55));
    }
void Parrayadd(void){
    Parray[0]=Parray[1];
        Parray[1]=Parray[2];
        Parray[2]=Parray[3];
        Parray[3]=Parray[4];
        Parray[4]=Parray[5];
        Parray[5]=vintoTorr(Pressure.read_u16());

    }

//////////////////////////////////////////////////////////////////////
//main
int main()
{

    Ticker PeriodicTrigger;
    red = 1;
    green =1;
    blue=0;

    PeriodicTrigger.attach(periodic, 0.125);
    BLE &ble = BLE::Instance();
    ble.init(bleInitComplete);
    while (ble.hasInitialized() == false) {}

    for(int a =0;a<6;a++){
         Parray[a]=vintoTorr(Pressure.read_u16());
         }
    //////////////////////////////////////////////////

    while(1) {
```

```
        if(periodic_event){
        Parrayadd();
        filter=4*Parray[5]+Parray[4]+Parray[3]-4*Parray[2]-Parray[1]-Parray[0];

        if(maxvalue<filter){maxvalue=filter;}
      Umbral=Umbral+0.2*(0.7*maxvalue-Umbral);
        if(filter>Umbral){
            systolicpressure_event = 1;
            }
        }
        if(systolicpressure_event){
            if(filter<=0){
                SystolicValue=Parray[4];
                systolicpressure_event = 0;
                UpdateValue_event=1;
                diastolicpressure_event=1;
                }
            }
        if(diastolicpressure_event){
            if(filter>=0)
            DiastolicValue=Parray[4];
            diastolicpressure_event=0;
            UpdateValue_event=1;
            }
        if(UpdateValue_event){
        update_Pressure_BloodPressureService(SystolicValue ,DiastolicValue,MeanArterialValue);

        MeanArterialValue=(SystolicValue+DiastolicValue)/2;
        if (SystolicValue>0x00ff){
            green=false;
            }
            UpdateValue_event=0;
        }
        update_Pressure_BloodPressureStream(Parray[5],Umbral,filter);

```

```

        red=!red;
        ble.waitForEvent();
    }
}
```

BLE_handling.cpp

```
#include "BLE_handling.h"

BloodPressureService *BloPressureService;

volatile bool bleIsConnected = false;
volatile bool STEP_reset = false;

void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE&        ble   = params->ble;
    ble_error_t error = params->error;

    if (error != BLE_ERROR_NONE) {
        onBleInitError(ble, error);
        return;
    }

    if(ble.getInstanceID() != BLE::DEFAULT_INSTANCE) {
        return;
    }

    ble.gap().onConnection(connectionCallback);
    ble.gap().onDisconnection(disconnectionCallback);


    uint16_t SystolicValue = 0; //  mmHg
    uint16_t DiastolicValue = 0; //
    uint16_t MeanArterialValue = 0; //

    BloPressureService=new BloodPressureService(ble, SystolicValue,
```

```
DiastolicValue, MeanArterialValue, 0);

    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE); //solo BLE y no visible
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LIST_16BIT_SERVICE_IDS, (uint8_t *)uuid16_list, sizeof(uuid16_list));//el servicio que se ofrece
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_NAME, (uint8_t *)BLE_DEVICE_NAME, sizeof(BLE_DEVICE_NAME)); //que nombre tendra el dispositivo
    ble.gap().setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED);
    ble.gap().setAdvertisingInterval(1000); /* 1000ms. */
    ble.gap().startAdvertising();
}
void onBleInitError(BLE &ble, ble_error_t error) {}

void connectionCallback(const Gap::ConnectionCallbackParams_t *params)
{
    bleIsConnected = true;
}
void disconnectionCallback(const Gap::DisconnectionCallbackParams_t *params)
{
    BLE::Instance().gap().startAdvertising();
    bleIsConnected = false;
}

void update_Pressure_BloodPressureService(uint16_t v1,uint16_t v2,uint16_t v3)
{
    BloPressureService->updateBloodPressureMeasurement(v1,v2,v3);
}
void update_Pressure_BloodPressureStream(uint16_t v1,uint16_t v2,uint16_t v3)
{
    BloPressureService->updateBloodPressureStream(v1,v2,v3);
}
```

BloodPressureService.h

```
#include "BLE.h"
//#include "Gap.h"

/*
* @This example can be demonstrated with Blood Pressure Android App downloaded from
* @https://github.com/Marcomissyou/BluetoothLeBooldPressure.git
* @Author: Marco.Hsu
* @Email: marco.missyou@gmail.com
*/
class BloodPressureService {
public:
const static uint16_t BloodPressureStreamservice              = 0xA020;
const static uint16_t BloodPressureStreamAttr              = 0xA021;
    BloodPressureService(BLEDevice &_ble, uint16_t Systolic, uint16_t Diastolic, uint16_t mean_arterial, uint16_t BloodFeature):
        ble(_ble),
        BloodPressureValue(Systolic, Diastolic, mean_arterial),
        BloodPressureStreamValue(0,0,0),
        BloodFeatureValue(BloodFeature),
        //GlucoseFeatureValue(GluFeature),
        BloodPressureGatt(GattCharacteristic::UUID_BLOOD_PRESSURE_MEASUREMENT_CHAR, BloodPressureValue.getPointer(),
                sizeof(uint8_t), BloodPressureStruc::MAX_VALUE_BYTES, GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_INDICATE),
        BloodFeature(GattCharacteristic::UUID_BLOOD_PRESSURE_FEATURE_CHAR, &BloodFeatureValue),
        BloodPressureStream(BloodPressureService::BloodPressureStreamAttr, BloodPressureStreamValue.getPointer(),
                sizeof(uint8_t), BloodPressureStruc::MAX_VALUE_BYTES, GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_INDICATE)
        {
            setupService();
        }

```

```
    void updateBloodPressureMeasurement(uint16_t Systolic, uint16_t Diastolic, uint16_t mean_arterial) {
        BloodPressureValue.uploadBloodPressureValue(Systolic, Diastolic, mean_arterial);
        ble.updateCharacteristicValue(BloodPressureGatt.getValueAttribute().getHandle(), BloodPressureValue.getPointer(), 6);
    }
    void updateBloodPressureStream(uint16_t v1,uint16_t v2,uint16_t v3) {
        BloodPressureStreamValue.uploadBloodPressureValue(v1, v2, v3);
        ble.updateCharacteristicValue(BloodPressureStream.getValueAttribute().getHandle(), BloodPressureStreamValue.getPointer(), 6);
    }



private:
    void setupService(void) {
        static bool serviceAdded = false; /* We should only ever need to add the heart rate service once. */
        if (serviceAdded) {
            return;
        }

        GattCharacteristic *charTable[] = {&BloodPressureGatt, &BloodFeature,&BloodPressureStream};
        GattService         BloodService(GattService::UUID_BLOOD_PRESSURE_SERVICE, charTable, sizeof(charTable) / sizeof(GattCharacteristic *));
        //GattCharacteristic *charTable2[] = {&BloodPressureStream};
        //GattService         BloodStream(GattService::UUID_HEART_RATE_SERVICE, charTable2, sizeof(charTable2) / sizeof(GattCharacteristic *));

        ble.addService(BloodService);
        //ble.addService(BloodStream);
        serviceAdded = true;

    }
```

```

//==========================================================================
private:
struct BloodPressureStruc {
        static const unsigned MAX_VALUE_BYTES  = 8;
        static const unsigned FLAGS_BYTE_INDEX = 0;

        static const unsigned Blood_Pressure_Units_Flag = 0;    // 0 in units of mmHg, 1 in units of kPa
        static const unsigned Time_Stamp_Flag = 0;              //
        static const unsigned Pulse_Rate_Flag = 0;
        static const unsigned User_ID_Flag = 0;
        static const unsigned Measurement_Status_Flag = 0;

        static const unsigned VALUE_FORMAT_BITNUM = 0;
        static const uint8_t  VALUE_FORMAT_FLAG   = 0;

        // Initial setting Blood Pressure measurement value
        BloodPressureStruc(uint16_t Systolic, uint16_t Diastolic, uint16_t mean_arterial) : BloodPressureValue() {
            uploadBloodPressureValue(Systolic, Diastolic, mean_arterial);
        }

        // function to update Blood Pressure measurement value in unit of mmHg
        void uploadBloodPressureValue(uint16_t Systolic, uint16_t Diastolic, uint16_t mean_arterial) {
            uint16_t ieee11073_Systolic = (Systolic);
            uint16_t ieee11073_Diastolic = (Diastolic);
            uint16_t ieee11073_mean_arterial = (mean_arterial);
            BloodPressureValue[FLAGS_BYTE_INDEX]    = (Blood_Pressure_Units_Flag << 0)|(Time_Stamp_Flag << 1)|
                                                      (Pulse_Rate_Flag << 2)|(User_ID_Flag << 3)|(Measurement_Status_Flag << 4);
            // memcpy(&BloodPressureValue[FLAGS_BYTE_INDEX], &ieee11073_Systolic, 2);
            // memcpy(&BloodPressureValue[FLAGS_BYTE_INDEX+2], &ieee11073_Diastolic, 2);
```

```
            // memcpy(&BloodPressureValue[FLAGS_BYTE_INDEX+4], &ieee11073_mean_arterial, 2);
            BloodPressureValue[FLAGS_BYTE_INDEX + 0] = (uint8_t)(ieee11073_Systolic &0xff);
            BloodPressureValue[FLAGS_BYTE_INDEX + 1] = (uint8_t)(ieee11073_Systolic >> 8);
            BloodPressureValue[FLAGS_BYTE_INDEX + 2] = (uint8_t)(ieee11073_Diastolic &0xff);
            BloodPressureValue[FLAGS_BYTE_INDEX + 3] = (uint8_t)(ieee11073_Diastolic >> 8);
            BloodPressureValue[FLAGS_BYTE_INDEX + 4] = (uint8_t)(ieee11073_mean_arterial &0xff);
            BloodPressureValue[FLAGS_BYTE_INDEX + 5] = (uint8_t)(ieee11073_mean_arterial >>8);
        }


        uint8_t       *getPointer(void) {
            return BloodPressureValue;
        }

        const uint8_t *getPointer(void) const {
            return BloodPressureValue;
        }

        uint16_t ieee11073_float16(float BloodPressure) {
            uint8_t  exponent = 0;
            uint16_t mantissa = (uint16_t)BloodPressure;
            return ((exponent << 12)&0xF0) | (mantissa << 0) & 0x0FFF ;
        }
private:
        uint8_t BloodPressureValue[MAX_VALUE_BYTES];
};


private:
    BLEDevice          &ble;
```

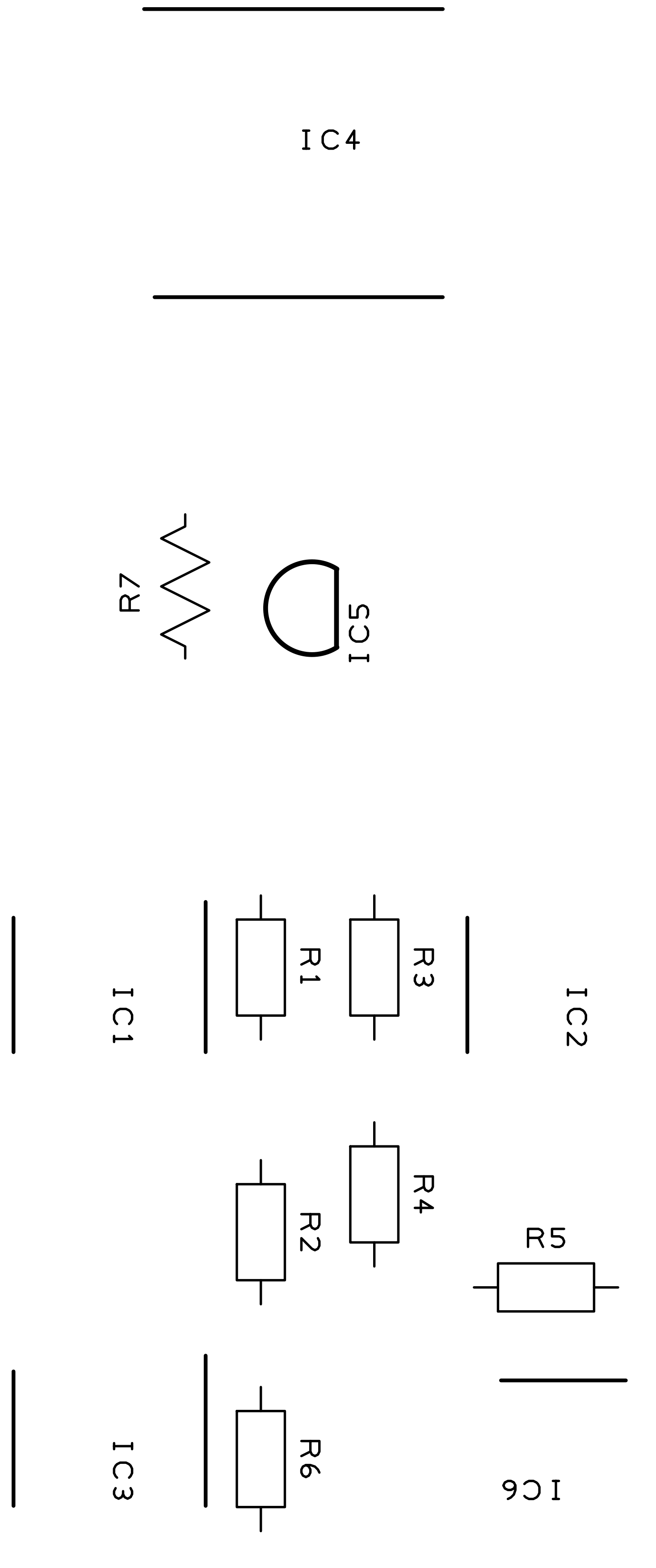
```

    BloodPressureStruc              BloodPressureValue;
    BloodPressureStruc              BloodPressureStreamValue;
    GattCharacteristic              BloodPressureGatt;  // data carrier
    GattCharacteristic              BloodPressureStream;

    uint16_t                        BloodFeatureValue;
    ReadOnlyGattCharacteristic<uint16_t>  BloodFeature;

};
```

B. Gerber output files


IC4
R7
IC5
R1
R3
IC1
IC2
R2
R4
R5
R6
IC3
IC6