

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Multi-Electrode Pacemaker

Joana Ferreira da Cunha

Master's in Electrical and Computers Engineering

Supervisor: Armando Luís Sousa Araújo

Co-Supervisor: Joaquim Adelino Correia Ferreira Leite Moreira

July 30, 2020

Resumo

Estudos revelam que as doenças cardíacas são a principal causa de morte no mundo [1] [2].

A arritmia cardíaca é um dos principais problemas cardíacos. Durante esta fase, o coração pode bater muito rápido, muito lento ou com um ritmo irregular, o que pode levar a complicações de saúde ou até à morte.

Para combater esses problemas, pode ser implantado um pacemaker . Estes dispositivos são responsáveis por fornecer estimulação elétrica ao tecido cardíaco e, assim, controlar a frequência cardíaca disfuncional.

O principal objetivo desta dissertação é apresentar uma pesquisa de mercado sobre as tecnologias existentes bem como descrever um método abrangente associado à implementação de um protótipo de um pacemaker implantável com recursos sem fio.

Assim, e baseando noutros dispositivos existentes, será desenvolvido um protótipo de um pacemaker multi-eléctrodo implantável, capaz de estimular várias regiões do coração. Este protótipo deve ser capaz de ler os impulsos elétricos do coração, detectar anomalias e fornecer estímulos elétricos ao tecido cardíaco.

Abstract

Studies reveal that heart related diseases are the main cause of death worldwide[1][2].

Cardiac arrhythmia is one of the main heart problems. During this phase, the heart can beat very fast, slowly or at an irregular pace, or it can lead to health complications or even death.

To combat these problems, a pacemaker can be implanted. These devices are responsible for providing electrical stimulation to the heart tissue and, thus, controlling the dysfunctional heart rate.

The main objective of this document is to present a market survey on existing technologies as well as to depict a comprehensive method to achieve the prototype of an implantable pacemaker with wireless capabilities.

Herewith, and based on other existing devices, an implantable multi-electrode pacemaker model will be developed, capable of stimulating various regions of the heart. This prototype must be able to read the electrical impulses of the heart, detect anomalies and provide electrical stimuli to the cardiac tissue.

Acknowledgments

I would like to express my great appreciation to Professor Dr. Armando Sousa Araújo, my research supervisor, for his patient guidance, enthusiastic encouragement and technical support during the course of my dissertation. He taught me the meaning of perseverance even in the most difficult moments. For that I'm deeply grateful.

Secondly, I would also like to thank Engineer Flávio Amorim, for his valuable advice and assistance in keeping my progress on schedule.

I would also like to extend my thanks to the technical department, Rui Carvalho, Pedro Alves and Isidro Pereira for their help in offering me the resources for this project and to all my teachers at the university whose efforts made me come so far.

I'm extremely grateful to my parents and brother who guided me throughout the years and gave me love and dedication. Your sacrifices for educating and preparing me for my future will not be in vain.

To my dear friends Inês Castelo Branco, João Azeredo, Ana Silva, Rafaela Machado and Mário Cardoso for all the confidence and friendship they gave me and continue to do so.

I also would like to thank my colleagues whom have helped me overcome many obstacles that allowed me to learn from and grow as an individual.

Finally, a special thanks to my boyfriend and colleague, Bruno Aguiar, who could always manage to find time to help me when I was in need. I'm profoundly grateful for his emotional support and care throughout these years. I could not have completed my Masters' dissertation without him.

Joana Cunha

*“He who asks is a fool for five minutes,
but he who does not ask remains a fool forever.”*

Chinese Proverb

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	1
1.3	Objectives	2
2	The Human Heart	3
2.1	The Circulatory System	3
2.2	Anatomy of the Human Heart	4
2.2.1	Cardiac Cycle	4
2.2.2	Electric Activity within the Heart	5
3	Bibliographic Revision	7
3.1	Existing Solutions	7
3.2	Pacemaker System	8
3.2.1	Pulse Generator	9
3.2.2	Lead Fixation Methods	10
3.2.3	Types of Devices	12
3.3	Conclusions	14
4	Project Overview	15
4.1	Clients Needs	15
4.2	System Concept	16
4.3	System Requirements	18
4.3.1	Functional Requirements	18
4.3.2	Non-Functional Requirements	18
4.4	Proposed Solution	19
4.5	Work Methodology	20
5	System Description: Hardware	21
5.1	System Modules	22
5.1.1	Control Module	22
5.1.2	Sensing Channel	24
5.1.3	Stimulation Phase	26
5.1.4	Power Supply	28
5.1.5	Detection Phase	30
5.2	Prototype Design	33
5.2.1	Sensing Channel	34
5.2.2	Stimulation phase	35

5.2.3	Power Supply	36
5.2.4	Detection Phase	37
5.2.5	Model Assembly	38
6	System Description: Software	41
6.1	Communication Protocol: TCP/IP	41
6.1.1	How does TCP/IP protocol work?	42
6.2	Client and Server	43
6.3	Used Technologies	45
6.3.1	Arduino Software	45
6.3.2	Python Software	46
6.4	Software Development Modules	48
6.4.1	Client Side	48
6.4.2	Server side	50
6.4.3	Interface reads Data from Database	51
6.5	Graphical Interface	52
6.5.1	HomePage	52
6.5.2	Vitals	53
6.5.3	Pacemaker Info	53
6.5.4	Device Management	56
7	Prototype Testing and Validation	59
7.1	Hardware Testing	59
7.1.1	Cardiac Cell	60
7.1.2	Detection Phase	62
7.2	Software Testing	64
7.2.1	Client Side	65
7.2.2	Server Side:	68
7.2.3	Interface reads data from database	69
7.2.4	Database export to Excel	70
	Conclusions and Future Work	71
A	Software Development Modules	73
A.1	Client Side	74
A.1.1	Client Creation	74
A.1.2	Send Data from Server to ESP32	75
A.1.3	Receive Data from Server	76
A.2	Server Side	77
A.2.1	Server Creation	77
A.2.2	Database Creation	78
A.2.3	Receive data from client and write to Database	78
A.3	Interface reads Data from Database	79
B	Graphical Interface	81
B.1	Device Management Page: Data Input	82
B.1.1	Management Function (Views.py)	82
B.1.2	Send to ESP32 Function (Views.py)	82
B.1.3	Forms.py	83

B.1.4	Management.html: Input Data Form	83
B.2	Device Management Page: Data Output	84
B.2.1	Data Export to Excel	84
B.2.2	Download Excel (Views.py)	84
B.2.3	Management.html: Output Data Button	84
References		85

List of Figures

2.1	Circulatory System Diagram. Extracted from [3]	3
2.2	Heart Constitution and its Valves. Extracted from [4]	4
2.3	Electrophysiology of the Heart. Extracted from [5]	5
2.4	ECG	6
3.1	Implantable Pacemaker. Extracted from [6]	8
3.2	Types of Leads. Adapted from [7]	11
3.3	A: Unipolar and B: Bipolar Polarity. Extracted from [8]	11
3.4	Single-chamber VS Dual-Chamber Pacemaker. Adapted from [9]	12
3.5	Biventricular pacemaker. Extracted from [9]	13
3.6	ICD. Extracted from [10]	13
3.7	Leadless Pacemaker. Adapted from [11]	14
4.1	System Concept Diagram	16
4.2	Concept Overview	17
4.3	Data Transfer Diagram	19
5.1	System Concept: Hardware Module	22
5.2	ESP32 DEVKIT V1 Pinout	23
5.3	OPA350 Pinout	24
5.4	DAC7311 Pinout	25
5.5	DAC8718 Pinout	26
5.6	MAX4603 Pinout	27
5.7	LTC3400 Pinout	28
5.8	LTC3436 Pinout	29
5.9	IR Emitter Diagram	30
5.10	Photo Transistor Diagram	30
5.11	ADS8661 Pinout	31
5.12	Circuit Diagram	33
5.13	Sensing Circuit	34
5.14	Sensing Schematics	34
5.15	Stimulation Phase Schematics	35
5.16	Power Supply Schematics	36
5.17	ECG Schematics	37
5.18	EAGLE Schematics	38
6.1	TCP/IP Protocol	42
6.2	Concept Diagram: Bidirectional Communication	43
6.3	Bidirectional Communication: Testing Circuit Schematics	44

6.4	Arduino Software (IDE)	45
6.5	Server Side: Data flow for TCP	50
6.6	Home Page	52
6.7	Vitals Page	53
6.8	Login Page	54
6.9	Register Page	54
6.10	Forgot Password Page	55
6.11	Device Information Page	55
6.12	Device Management Page	56
7.1	Cardiac Cell Model	60
7.2	Transient Analysis of the Model. Extracted from [12]	60
7.3	Transient Analysis of the Model: Obtained Result	61
7.4	Implementation of ECG Scheme	62
7.5	Filtered and Amplified Wave	63
7.6	Bidirectional Communication: Testing Circuit Assembly	64
7.7	Client Successfully Created	65
7.8	Connection to Server Failed	65
7.9	Connection to Server Successful	65
7.10	Read data from client: Button not Ticked	66
7.11	Read data from client: Button Ticked	66
7.12	Potentiometer Values on serial monitor	66
7.13	Potentiometer Values on backend terminal	66
7.14	Send Values to Server	67
7.15	Receive data from Server	67
7.16	Server Creation: Pycharm Terminal	68
7.17	Server Creation: Establishing connection to backend.py	68
7.18	Project without database	69
7.19	Database Created	69
7.20	Export Data to an Excel file	70
7.21	Values Sent to the Server	70
7.22	Values Saved in the Excel File	70

List of Tables

4.1	Client Needs	15
4.2	Functional Requirements	18
4.3	Non-Functional Requirements	18
5.1	ESP32 DEVKIT V1 Pinout description	23
5.2	OPA350 Pinout terminal description	24
5.3	DAC7311 Pinout terminal description	25
5.4	MAX4603 Pinout terminal description	27
5.5	LTC3400 Pinout terminal description	28
5.6	LTC3436 Pinout terminal description	29
5.7	ADS8661 Pinout terminal description	31
6.1	Django Vs Flask	46
6.2	Python Libraries	47

Acronyms and Symbols

SA	Sinoatrial
AV	Atrioventricular
RA	Right Atrium
LA	Left Atrium
RV	Right Ventricle
LV	Left Ventricle
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
I2C	Inter-Integrated Circuit
SDIO	Secure Digital Input Output
ECG	Electrocardiogram
PC	Personal Computer
TCP/IP	Transmission Control Protocol/Internet Protocol
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
RTP	Real-time Transport Protocol
DNS	Domain Name System
UDP	User Datagram Protocol
ICMP	Internet Control Message Protocol
LAN	Local Area Network
ARP	Address Resolution Protocol
DSL	Digital subscriber Line
JSON	JavaScript Object Notation
IoT	Internet of Things
App	Application
PCB	Printed Circuit Board
SSID	Service Set Identifier
API	Application Programming Interface
WWW	<i>World Wide Web</i>

Chapter 1

Introduction

This chapter has as main objective to present the dissertation structure whose project consists in the study, development and test of an implantable pacemaker. A global view of the proposed work is provided, emphasising its importance following the contextualisation, motivation and the objectives to which this dissertation is proposed.

1.1 Context

Implantable pacemakers are devices that apply electric impulses to maintain the rhythm of the heart and, if necessary, restart its beat. These devices perform a very important role in the treatment of heart-related diseases that provoke the dysfunctional behaviour of the heart.

There are many reasons for implanting a pacemaker. The natural ageing of the heart's electrical system, with the respective failures of electrical conduction is the main reason, as well as arrhythmias (abnormal heart rhythms), genetic alterations and interventions in heart valves. Currently, we are faced with the art and ingenuity of converting pre-developed technologies for a specific purpose into new applications, as well as great diversity, flexibility, in the adaptation and development of new technologies.

1.2 Motivation

Continuous efforts in order to further develop medical healthcare tend to need technological support and development.

Knowing that cardiac-related diseases tend to have the highest mortality rate worldwide, it is necessary to make an effort in order to develop newer technologies or upgrade existing ones.

It is possible to find numerous technological innovations and solutions already existing on the market that are improving global healthcare drastically.

1.3 Objectives

The objectives of this dissertation are:

- Analysis of the state-of-the-art of the pacemaker technology:
- The human heart:
 - Understanding the circulatory system;
 - Anatomical study of the human heart;
- Study and choice of materials to be used during the development of the prototype;
- Development of the prototype using multiple leads;
- Development of a control application to communicate (via Wi-Fi) with module and allow data display;
- Tests and validation of the prototype (hardware and software);

Chapter 2

The Human Heart

In order to better understand the importance and function of the device to be developed, the outline of the basic functionalities of the human heart will be presented in this chapter.

2.1 The Circulatory System

The circulatory system is made up of vessels and muscles that control the blood flow around the body. This is called circulation. The same system is composed of both the systemic and pulmonary circulation.

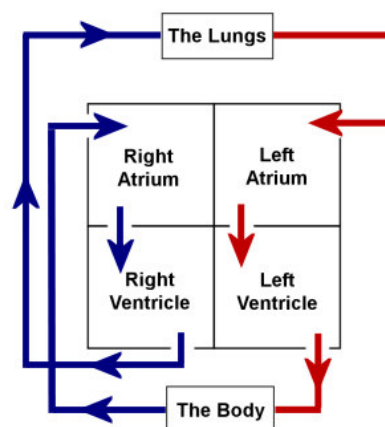


Figure 2.1: Circulatory System Diagram. Extracted from [3]

The main parts of the circulatory system are the **heart** and **vessels (arteries and veins)** that does a very important job in the human body, that is, they are responsible to pump blood (heart) and carry in oxygen and essential nutrients to all cells around the body (arteries) as well as carrying out the waste products and carbon dioxide (veins).

The vessels of the circulatory system are the **Arteries** (responsible of carrying blood out of the heart), **Veins** (responsible of carrying blood to the heart) and the **Capillaries** (smaller blood vessels that allow gas and nutrients exchange).

Systole and diastole are referred to ventricular contraction and relaxation respectively.

Blood enters the heart right atrium in diastole through the superior and inferior vena cava. As the right atrium contracts blood is passed to the right ventricle. Then the right ventricle contracts causing the blood to go to the pulmonary artery directly to the lungs. The blood arrives from the lungs and returns to the heart through the pulmonary veins, and to the left atrium. The left atrium contracts and blood is passed to the left ventricle through the mitral valve, then the left ventricle contracts and the blood is passed through the aorta artery where it is distributed throughout the body.

2.2 Anatomy of the Human Heart

The heart itself is the main propulsive organ for circulating the blood. It is a valvular, muscular pump that is cone shaped and is composed of 4 chambers: 2 auricles/atria and 2 ventricles. On each side, the atrium is connected to the ventricle by a one-way valve. The blood is pumped as these cavities contract and relax in sequence.

2.2.1 Cardiac Cycle

The cardiac cycle refers to a sequence of events that occur repeatedly in every heartbeat. It can be divided in two phases: systole and diastole. Systole and diastole can be divided into smaller phases.

Some of the basic principles are:

- The blood always flows from higher to lower pressure;
- Contraction increases the pressure within the chamber while relaxation lowers the pressure;
- *Atrioventricular* valves open when atrial pressure is higher than ventricular pressure and close when the pressure gradient is reversed;
- *Semilunar* valves (composed by Pulmonary and Aortic valves) open when ventricular pressure is higher than aortic/pulmonary pressure and close when the reverse is true;

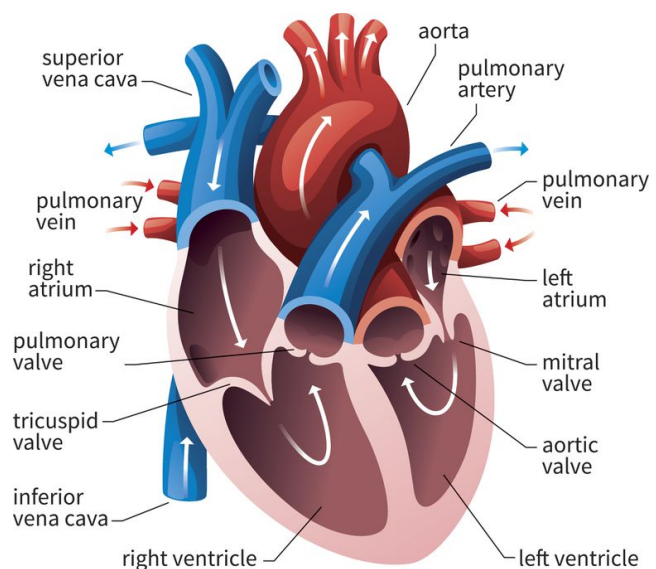


Figure 2.2: Heart Constitution and its Valves. Extracted from [4]

Cardiac circuits supply the tissues in the body with oxygen and nutrients, which are then transported by the blood. The pulmonary circuit carries the blood that needs to be oxygenated in the lungs. The exchange of oxygen and carbon dioxide produced by the body occurs before the blood returns to the heart.

2.2.2 Electric Activity within the Heart

The beating of the heart is regulated by electrical impulses. This sequence begins as the atria fills with deoxygenated blood from the body and oxygenated blood from the lungs. Then, an electrical signal from the *Sinoatrial* node causes the atria to contract forcing blood into the ventricles. The electrical signal is then detected by the *Atrioventricular* node and directed into the Purkinje fibers located in the ventricle walls causing the ventricles to contract.

After this, the blood is pumped through the pulmonary valve on the right to the lungs and the aortic valve on the left to the rest of the body. These valves close and the cycle restarts. [13, 14]

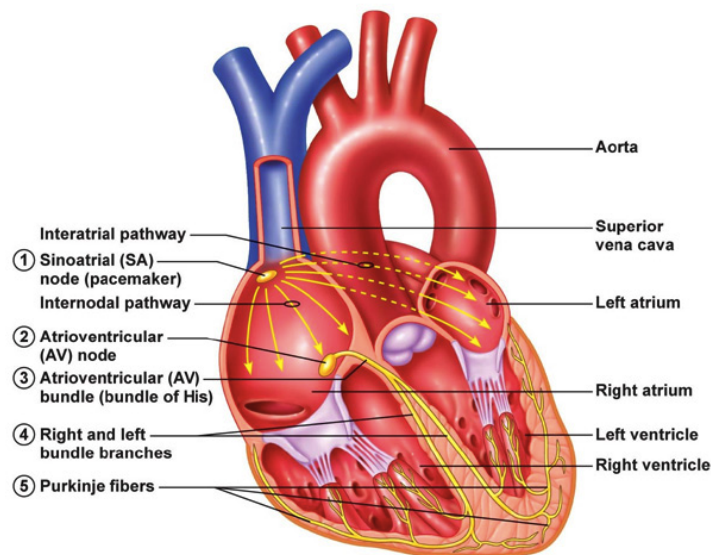


Figure 2.3: Electrophysiology of the Heart. Extracted from [5]

When the *Sinoatrial* node fails to properly function, an artificial pacemaker can be implanted to help regulate the heartbeat with small evenly timed electric shocks. This involves implanting electrodes into one or more of the hearts' cavities by inserting probes into the vein near the collarbone and implanting the generator of electric signals under the skin. [15]

2.2.2.1 Electrocardiogram (ECG)

An electrocardiogram will record the electrical activity in the heart.

At every beat, the heart is depolarized to trigger its contraction. This electrical activity is transmitted throughout the body and can be picked up on the skin.

The basic pattern of the ECG is that:

- Electrical activity towards a lead causes an upward deflection;
- Electrical activity away from a lead causes a downward deflection
- Depolarization and repolarization deflections occur in opposite directions;

The basic pattern of this electrical activity comprises three waves, which are named P, QRS (a wave complex), and T [16].

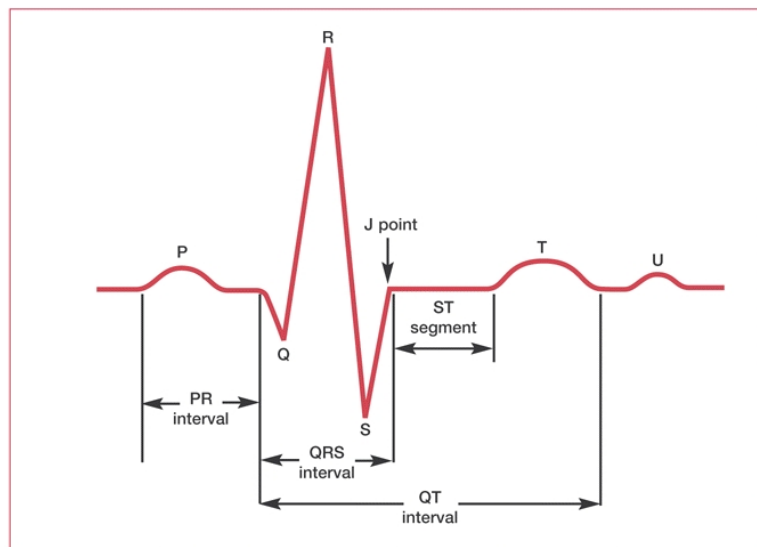


Figure 2.4: ECG

The P wave is a small deflection wave that represents atrial depolarisation. The PR interval is the time between the first deflection of the P wave and the first deflection of the QRS complex. The three waves of the QRS complex represent ventricular depolarisation.

The ST segment is the time between the end of the QRS complex and the start of the T wave. It reflects the period of zero potential between ventricular depolarisation and repolarization.

Lastly, the T waves represent ventricular repolarization.

Chapter 3

Bibliographic Revision

In this chapter, a theoretical survey will be carried out to better understand the different pacemaker technologies, followed by an analysis of existing solutions on the market and consequent analysis on the pacemaker system to be developed.

3.1 Existing Solutions

The history of cardiac pacing therapy must be viewed within the extensive framework of electro-therapy. Moreover it can be observed that the development of electro-therapy usually preceded the understanding of what was actually occurring within the heart. Electro-therapy has a simple base concept: the use of an outside source of electricity to stimulate human tissue to produce a beneficial therapeutic effect.

Over the last fifty years, electro-therapy has shown a very rapid growth of innovations contributing to the development of the modern pacemaker.

The evolution of the modern pacemaker dates back to 1958 when the first implementation took place. They are small and lightweight devices that have electronic algorithms, can be programmed, help a very physiological behaviour and monitor all heartbeats in real time. Only when the heart has a delay in the depolarisation, does the pacemaker perform electrical stimulation.

Currently, these devices have evolved in terms of increasing the accuracy of the specific location of action near the heart, as well as the introduction of sensors that allow the device to assess the exact situation, not forgetting to mention the significant reduction in size .

Depending on the implantation method, there are two types of pacemakers.

- The Endocardial pacemaker, the most used, is implanted under the skin of the chest or abdomen through a very simple operation, while the lead is inserted through a vein until its fixation is achieved;
- The Epicardial pacemaker, requires a more complex surgical intervention, since the lead is implanted in the external wall of the heart, while the device can be placed like the previous one or even outside the body. In any case, this type of pacemaker is only used as a temporary preventive measure.

3.2 Pacemaker System

The pacemaker consists of a battery or power source, a generator that emits electrical pulses at a certain frequency and a lead (with an electrode attached), connected to the generator, which conducts electrical impulses to the heart muscle. Many pacemakers also have movement or breathing sensors that allow the devices to adjust to the need for a faster heart rate when exercising.

The pulse generator contains a lithium battery and electrical circuitry attached to one, two, or three leads that are inserted into the heart and the stimulation leads are covered in an isolating material

This device must be at all times monitoring the frequency of the hearts' pulsation and the electrical signal delivered must have an amplitude high enough to stimulate the patients' heart.

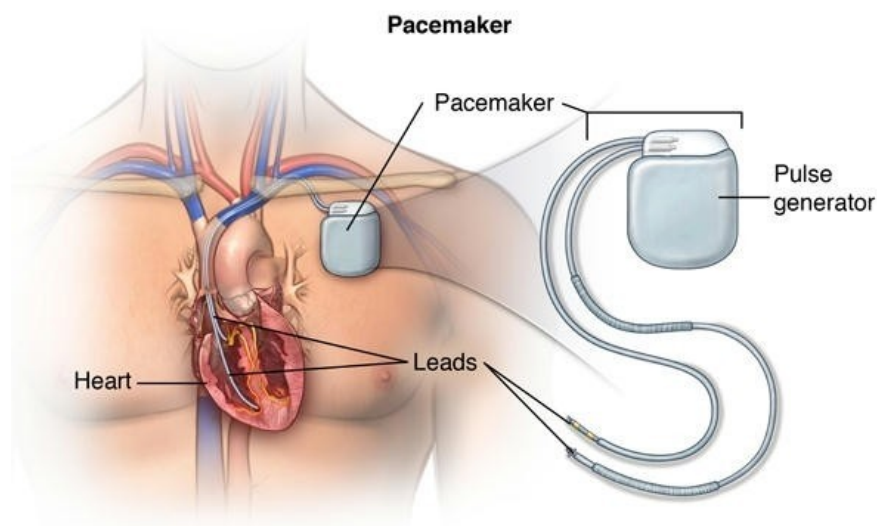


Figure 3.1: Implantable Pacemaker. Extracted from [6]

3.2.1 Pulse Generator

The pulse generator has a shell of titanium being this a well tolerated material by the surrounding tissues. This device is made of four modules:

- Power Source;
- Detection Circuit;
- Electrical Circuit;
- Control;

In this section, these modules will be depicted.

3.2.1.1 Power Source

The pacemaker devices also need a power source such as batteries. The batteries used in implantable cardiac pacemakers imply exclusive challenges for their developers and manufacturers in terms of levels of safety and reliability.

Additionally, batteries must have longevity to avoid frequent replacements. A cardiac pacemaker uses half of the battery's energy for cardiac stimulation and the other half for household tasks, such as monitoring and recording data. The first cardiac pacemaker implanted used a rechargeable nickel-cadmium battery, later it was developed and used a zinc-mercury battery that lasted more than 2 years.

The lithium iodine battery invented in 1972 had a real impact on implanted cardiac pacemakers. This battery lasts about 10 years and is still the source of energy for many manufacturers of cardiac pacemakers.

The main objective of the battery is to store enough energy to stimulate the heart with a jolt of electricity. Additionally, it also provides power to the sensors and timing devices.

They must be able to generate about five volts of power, a slightly higher value than the amount required to stimulate the heart.

3.2.1.2 Detection Circuit

This module is responsible of amplifying and filtering cardiac signals and respective comparison with a reference signal in order to detect possible anomalies.

The sensing channel must be active during most part of the cardiac cycle and therefore, will need to have low energy consumption.

3.2.1.3 Electrical Circuit

Modern pacemaker circuitry is a vast improvement over earlier models. With the application of semiconductors, circuit boards became much smaller. They also require less energy, produce less heat, and are highly reliable.

A reading routine will be conducted, taking the information to the microcontroller. Depending on the received information, the electrical circuit will send a signal of stimulation to the heart through the electrodes.

Nowadays, there are pacemakers without batteries, which are connected directly to the heart, and there also modules that are capable of sending and receiving signal information without the need of wires (wireless).

3.2.1.4 Control Module

A pacemaker which includes a conventional programmable pulse generator, a physiological sensor (e.g. digital motion sensor), and a processor, generates heart stimulation pulses on demand.

The physiological sensor generates a raw signal which varies as a function of some physiological parameter, such as activity level to provide some indication of whether the heart rate should increase or decrease, and hence whether the pacemaker should change the rate at which pacing pulses are provided.

The microprocessor converts the raw signal to the sensor-indicated rate signal. This sensor remains at a base rate for all sensor level index signals below a prescribed rate response threshold and thus it is possible to program the duration time period of the stimulation phase.

With this, both the Electrical Circuit and Detection Circuit aforementioned are controlled by a microprocessor making the system fully automatised.

In modern pacemakers, the microprocessor has low power making it durable, reliable and robust.

3.2.2 Lead Fixation Methods

Poor electrode placement can result in mistaken interpretation, and then possible misdiagnosis. Development of fixation technology has also played a role in lead transformation. Stability of fixation to ensure effective long-term pacing is the main principle.

Pacemaker leads are composed of a connector pin, conductor, insulation, electrode, and fixation device. They can be distinguished as epicardial or endocardial leads.

Endocardial leads are fixed in the myocardium by passive fixation with barbs or active fixation with screws in which active fixation can be facilitated by either fixed or retractable screws, providing secure fixation in locations with low trabeculation or the atrium.

On the other hand, epicardial leads are used in special situations or as a bail-out when endocardial leads insertion is not possible

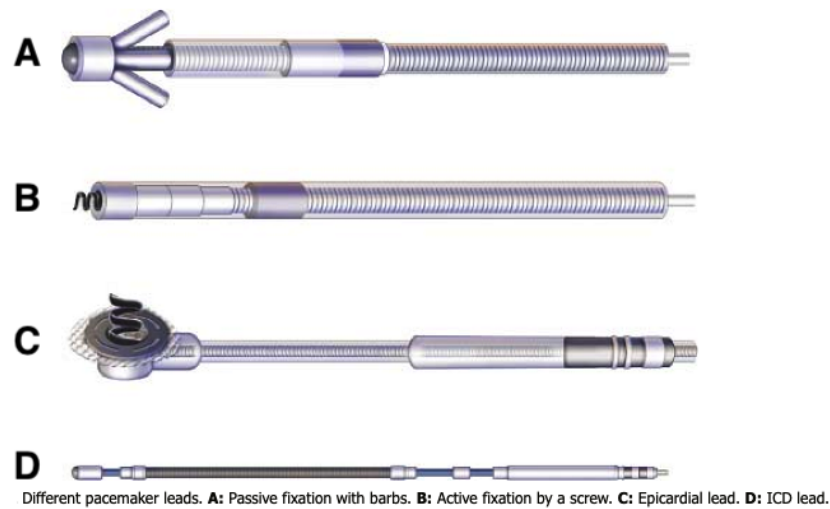


Figure 3.2: Types of Leads. Adapted from [7]

3.2.2.1 Lead Polarity

There are two types of lead stimulation: unipolar and bipolar. Unipolar lead is a single conductor lead with an electrode located at the tip and a bipolar lead has two separate and isolated conductors within a single lead;

The difference between both is in the way the pacing current is injected into the heart. With bipolar, the return circuit is through the leads which is more efficient while with unipolar the anode is connected to the capsule. Electrons travel a longer distance in unipolar pacing which requires more energy to successfully depolarise the myocardium.

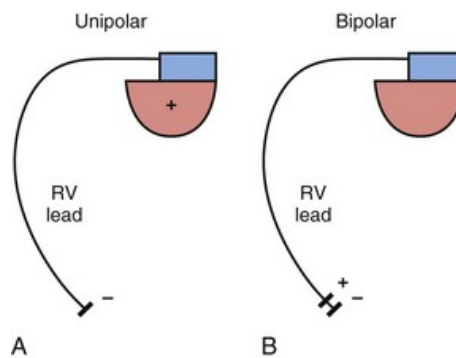


Figure 3.3: A: Unipolar and B: Bipolar Polarity. Extracted from [8]

3.2.3 Types of Devices

3.2.3.1 Single-chamber Pacemaker and Dual-Chamber Pacemaker:

The single-chamber pacemakers, have only one lead and pace only the ventricle, so all these devices can do is to generate a regular electrical pulse with no variation. It also does not account for the atrium, or upper chamber of the heart in patients who have an underlying rhythm in the atrium.

On the other hand, the dual-chamber pacemaker is a device that has two leads and can pace both the atrium, the ventricle and the inferior chamber in sequence. This represented a major evolution in the pacemaker technology since it could function like a normal heart does.

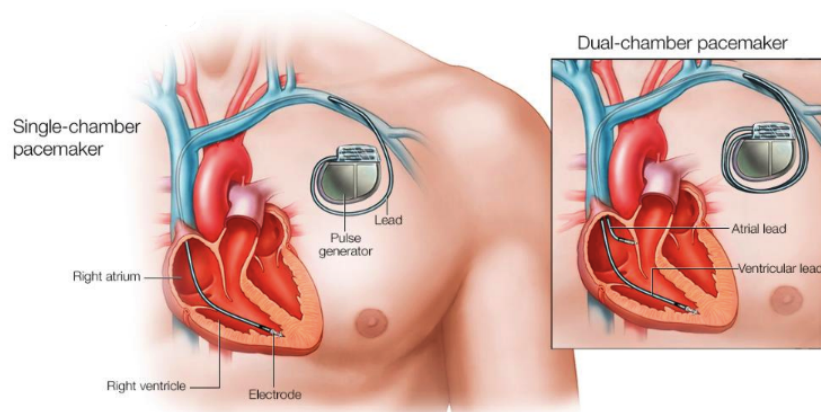


Figure 3.4: Single-chamber VS Dual-Chamber Pacemaker. Adapted from [9]

3.2.3.2 Biventricular Pacemaker:

The pace of the heart should also be taken in consideration in order to allow the patient to have increased heart rate with activity and thus, the "rate responsive pacemaker" appeared. This pacemaker used a piezoelectric crystal that would deform with body activity. The crystal has since been replaced by an "accelerometer" to reflect activity more accurately (there are also other types of sensors).

A normal pacemaker has a lead that paces the right ventricle. Some patients have decreased heart function, and a pacemaker that paces both ventricles simultaneously has been proven to improve heart function. This is called "biventricular pacing" and has leads that can synchronise the contractions of all chambers to optimise the heart function. This kind of pacemaker is normally used to treat severe heart failure.

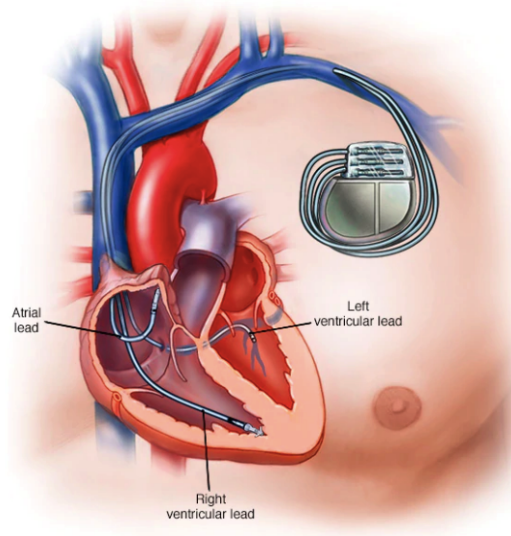


Figure 3.5: Biventricular pacemaker. Extracted from [9]

3.2.3.3 Implantable Cardioverter Defibrillator (ICD):

For more severe heart conditions, an implantable defibrillator can be used. This device has many similarities with the pacemaker itself but with the difference that it is also capable of sensing a stopped heart and delivering a powerful electric shock to restart it.

Since this device has the need to create electric shocks, the battery drainage is higher leading to a shorter lifetime of the battery.

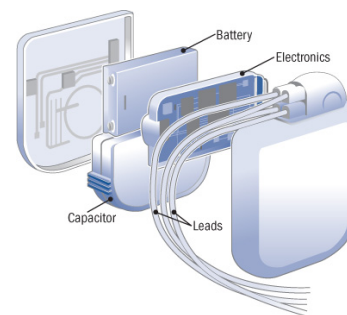


Figure 3.6: ICD. Extracted from [10]

3.2.3.4 Leadless Pacemaker:

Although the leadless pacemaker is a new technology, the preliminary reports of long-term performance and complications are favourable, including low complications, few system revisions, and stable pacing parameters. Leadless pacemaker is a self-contained generator and electrode system implanted directly into the right ventricle. The device is implanted via a femoral vein transcatheter approach; it requires no chest incision or subcutaneous generator pocket. It also has cosmetic appeal because there is no chest incision or visible pacemaker pocket.

This device provides only single-chamber ventricular pacing and lack defibrillation capacity and there are two available. [11]

Two leadless pacing systems are currently available: the Micra transcatheter Pacing system (Medtronic) and the Nanostim Leadless Cardiac Pacemaker (St. Jude Medical). Both systems provide right ventricular sensing, pacing, and rate responsiveness.[17]

The future of leadless device technology is promising and might eventually lead to expanded pacing capabilities.

A wireless cardiac system for left ventricular pacing is currently under investigation. This pacing electrode is able to convert the acoustic energy to an electric pacing impulse.[18]

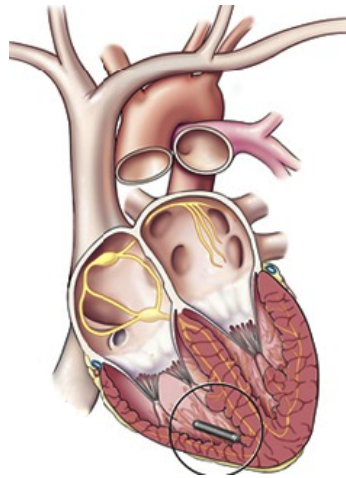


Figure 3.7: Leadless Pacemaker. Adapted from [11]

3.2.3.5 Battery-less Pacemaker:

The battery-less pacemaker, developed by researchers at Rice University and the Texas Heart Institute, can be implanted directly into the heart and doesn't require surgeries to periodically replace the batteries, as is the case in traditional pacemakers.

This device harvests energy wirelessly from radio frequency radiation transmitted by an external battery pack.

The frequency of the pacing signal produced by the pacemaker can be adjusted by increasing or decreasing the power transmitted to the receiving antenna, which stores the power until a predetermined threshold - then releasing the electrical charge to the heart and filling up again.[19]

3.3 Conclusions

Progress in battery technology, such as using radioactive isotopes for power, will likely improve the longevity of implanted pacemakers.

Developments in microelectronics will provide even smaller devices which are less prone to environmental interferences. There's also being conducted some developments in the field of the application of cardiac pacemaking technology to the brain. In this system, the lead wires are connected to a specific site on the brain and stimulate it as needed to regulate heartbeat. This device has been shown to be particularly effective in calming the tremors associated with Parkinson's disease.

Chapter 4

Project Overview

In order to develop a project overview, it must meet several requirements set by the client. In this case the client can be the patient or the medic maintaining the device.

4.1 Clients Needs

This section lists the client's needs for the project in question.

#	Description
N1	The system must be able to show the client his pace status at all times
N2	The system should be easily scalable and with a certain degree of abstraction in order to allow addition and removal of new functionalities
N3	The system must be able to communicate directly with the control module and a control application
N4	The system must be able to support different communication protocols
N5	The system must be able to support a graphical interface
N6	The system must be able to read signals from the heart
N7	The system must be able to regulate the stimuli voltage and precision voltage
N8	The system must have a power source require low power consumption
N9	The system must be meticulously built allowing its electrical circuit to be protected from its surroundings
N10	The system must be able to support Wi-Fi communication between module and interface

Table 4.1: Client Needs

4.2 System Concept

In order to characterise the problem, a conceptual diagram is depicted. The following image (Figure 4.1) shows the system conceptual diagram in order to better understand its intended structure:

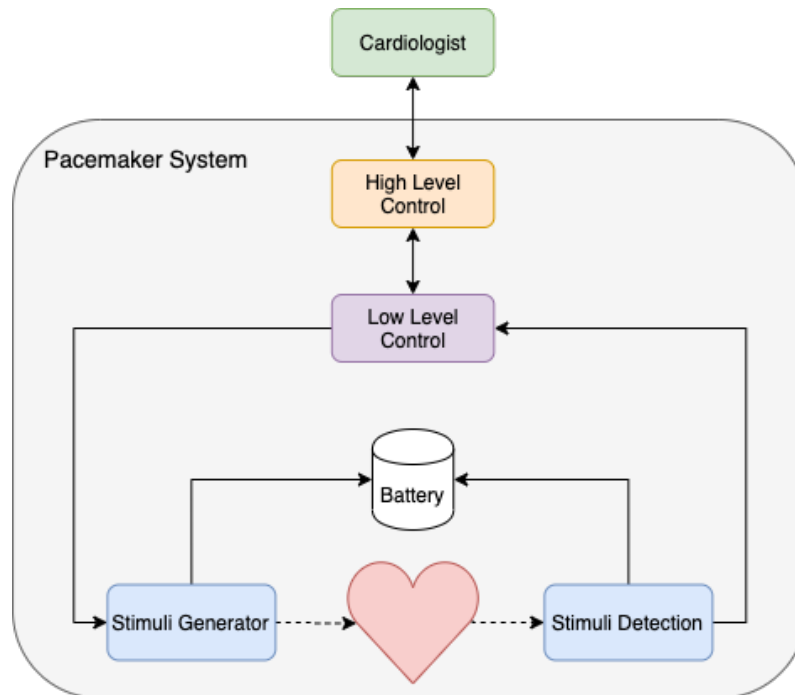


Figure 4.1: System Concept Diagram

As mentioned in previous chapters, pacemaker devices are seen as an extremely valuable resource in order to treat certain heart conditions.

Herewith, there is an increasing interest in exploring this technology and this is one of the major objectives of the system development: conceptualising and implementing a fully functional pacemaker prototype.

These two fundamental components of the product underpin the basics of a functional, yet rudimentary pacemaker. The signals sent and received from this device will be very important to enable the application to communicate not only with the device but also with the application interface in order to control the status in real-time so that present information about the state of the heart is always available. On the other hand, it will also provide a connection to information on previous data.

So, the following diagram presents an overview of the project concept.

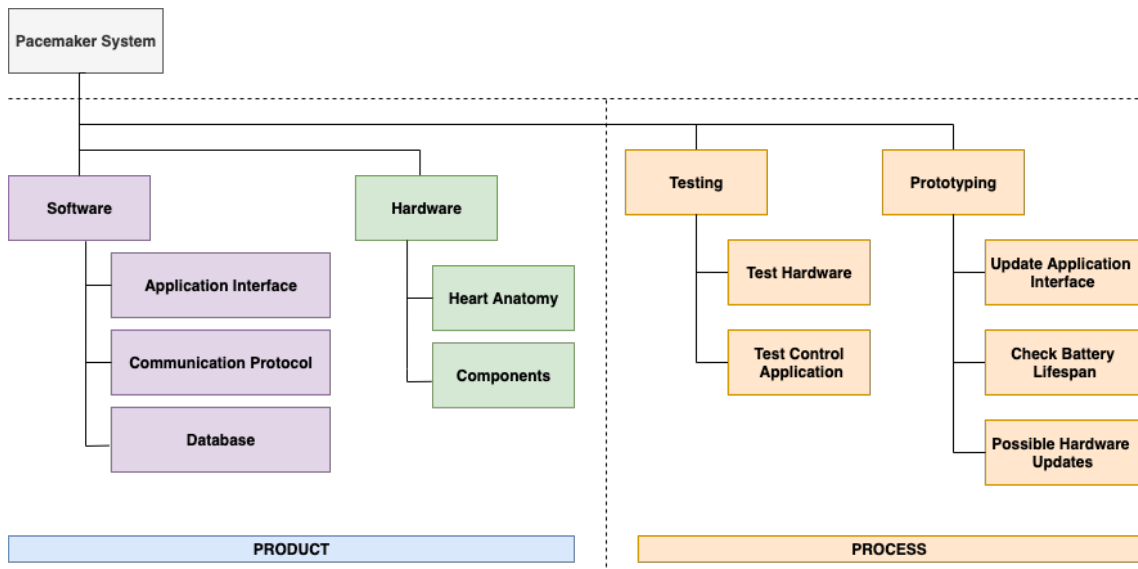


Figure 4.2: Concept Overview

In order to comply to the constraints defined by the project's client, the system must be based on the anatomy of the heart, more precisely, the hearts electrophysiology. Therefore, in order to interact with the natural behaviour of the heart, electric circuitry must be cautiously developed (in this case is the hardware structure depicted in the figure).

As observed, another functionality to be implemented in the application is relative to the testing and prototyping of the system when they're needed.

This involves continuous hardware testing and also application control testing in order to check if given values are correctly depicted.

Regarding the prototyping, the system must be capable of performing the necessary updates determined by the test results. Another relevant process is to frequently check the battery lifespan so the device wont run out of power.

4.3 System Requirements

This section sets out the requirements to which the system must meet to satisfy client needs. System requirements are divided into functional and non-functional requirements. The priorities of each requirement are determined according the clients' needs previously defined.

4.3.1 Functional Requirements

Description	Clients Needs
Capability of multi-electrode stimulation	N1
Capability of reading the hearts natural stimuli	N7
Capability of regulating stimuli voltage between 0V and 5V	N8
Communication and control of the module performed by the user	N1, N10
Precision regulation must be in the milivolts range	N8
System powered by a battery	N2, N9
Components of the prototype must have low power consumption	N9

Table 4.2: Functional Requirements

4.3.2 Non-Functional Requirements

Description	Clients Needs
The system should be of a small size in order to be implantable	N2
The system should be reliable and safe	N2, N9, N10
The system should be robust	N2, N9

Table 4.3: Non-Functional Requirements

4.4 Proposed Solution

One of the major objectives to be developed is to implement data transfer wirelessly from the pacemaker module to the application interface and vice-versa.

The following image shows a conceptual diagram of the data flow from the pacemaker device to the graphical interface.

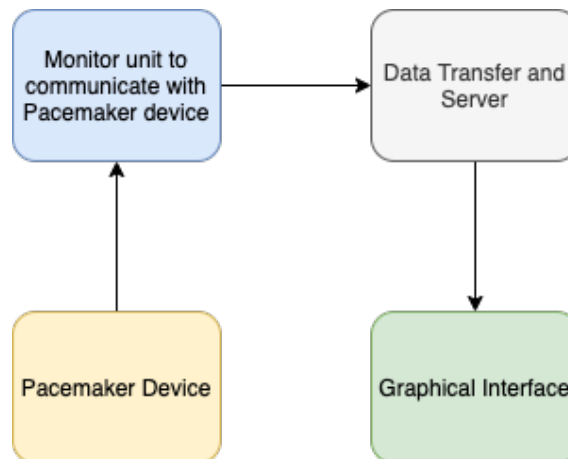


Figure 4.3: Data Transfer Diagram

The data transmission includes information stored in the device about arrhythmias, device integrity, physiological criteria, battery deficiency and lead failures that may result in inappropriate stimulation increasing the risk of life-threatening events.

The data can be transferred from the device to the patient monitor unit through different ways such as radio frequency and wireless telemetry that is used to download data from the device (it can also be transferred manually by the user).

The graphical interface must show both automated and manual data transmission.

Another important objective is to have equipment with low energy consumption in order to avoid damage to the devices' components and, consequently, lower its lifespan.

The microcontroller to be used will be ESP32 DevKit V1 for the stimulation circuit as well as the transmission circuit which can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

Therefore, the purpose of this dissertation is to develop a previously developed model by Engineer Flávio Amorim in his master thesis [20] and its objective is to deepen existent knowledge on the pacemaker device and develop newer competences on this field by designing and implementing a multi-electrode system capable of stimulus location with wireless capabilities.

4.5 Work Methodology

This dissertation is based on the premise of initial planning to outline gradual improvement strategies.

The conception of this work envisaged for the consolidation, organization and methods where the main objective is to make the proposed time more profitable in order to guarantee the compliance with all deadlines.

The methodology used in this work is descriptive, analytical and interpretative throughout the development of the project.

Therefore, the project must be analysed weekly in order to comply to the deadlines imposed:

- Components Orders:

Verification of needed material;

- Tasks Update:

Usage of agile methodology (using [trello](#)) in order to keep track on the current tasks and future ones, prioritising some tasks over others;

- Testing and Prototyping:

Preform tests during the development of the hardware module as well as the software module;

Chapter 5

System Description: Hardware

Modern pacemakers add several features to the basic function of stimulating the heart such as a detection of the spontaneous hearts' contraction, a telemetry link to receive and send programmable parameters, battery voltage measurement and methods for sensing physical activity in order to adapt the heart rate accordingly.

At the conclusion of the proposed solution phase, research and definition of the concrete characteristics of the prototype followed. These characteristics consist mainly in the definition of the the process of the proposed solution as well as its design, implementation and testing phase.

In order to achieve the proposed tasks, the project is divided in two parts: Hardware (chapter 5) and Software (chapter 6).

Moreover, a technical explanation of the decisions made regarding the use and function of each component and its integration as a whole is presented in this chapter.

5.1 System Modules

As previously mentioned, in order for the system to interact with the natural behaviour of the heart, the electric circuitry must be cautiously developed and so, the proposed solution starts with the division of the system by modules being these the Control Module, Sensing Channel, Stimulation Phase, Power Supply and Detection Phase.

It is essential to note that the choosing of the components depended on several characteristics such as energy consumption, price, compatibility with other components and availability of said components from the university providers.

When analysing the system concept, a module of the hardware can be visualised in the following diagram and further explained in the subsequent sections.

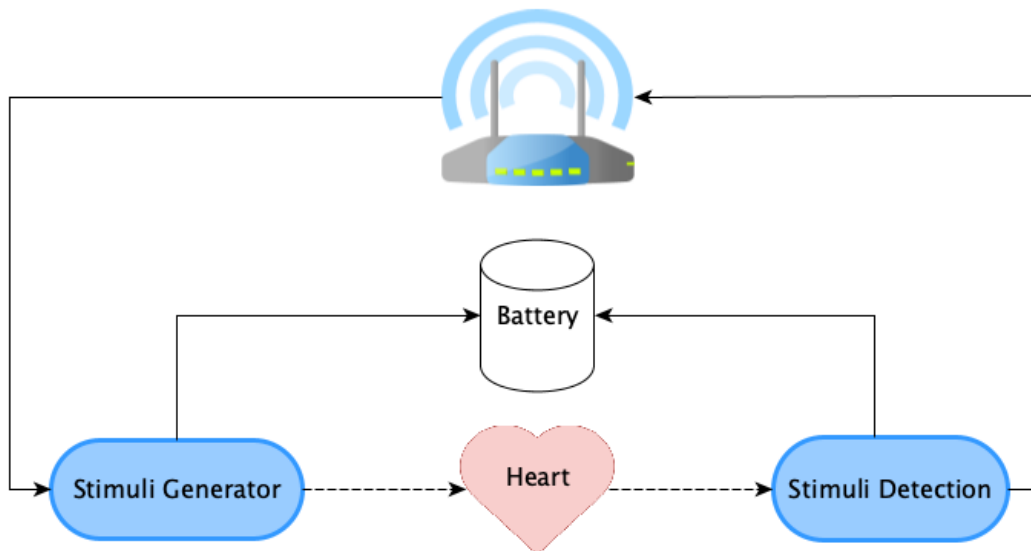


Figure 5.1: System Concept: Hardware Module

The hardware module is composed of the microcontroller (control module), the power supply, the stimuli generator, the heart and the stimuli detection.

5.1.1 Control Module

The control module *ESP32 DEVKIT V1* contained in the *WROOM 32 Series* is intentionally designed for Internet of Things (IoT) projects. It is equipped with a dual core 32-bit CPU, built in Wi-Fi and dual-mode Bluetooth with sufficient amount of 30 I/O pins and it can be programmed in C language directly from the Arduino IDE (subsection 6.3.1).

Figure 5.2 depicts the ESP32 DEVKIT V1 pinout [21] following a pinout terminal description is present in table 5.1.

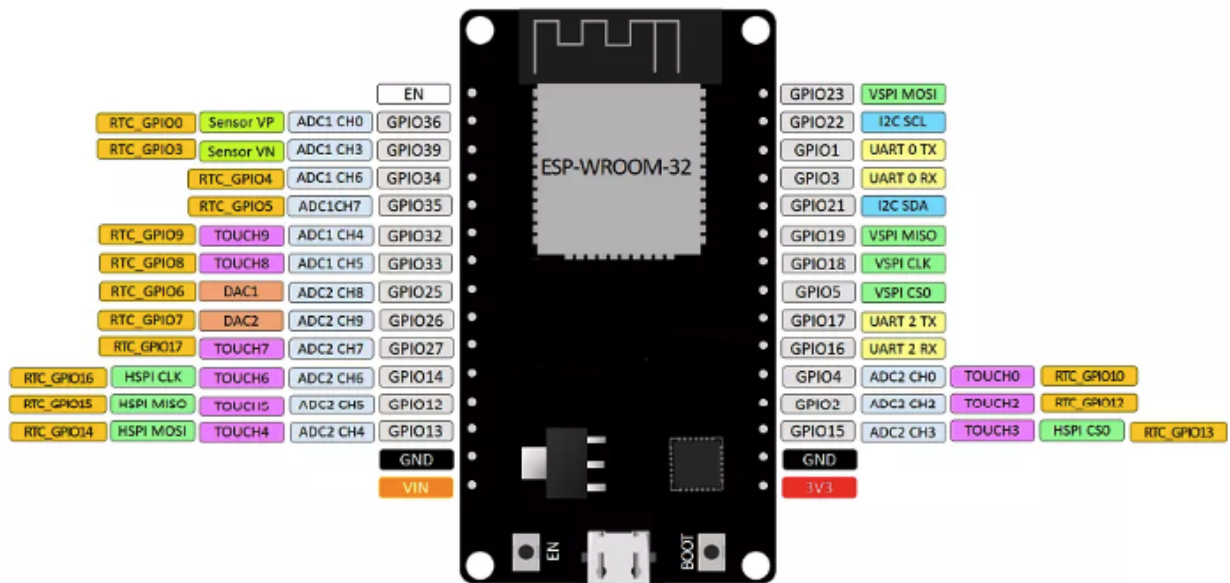


Figure 5.2: ESP32 DEVKIT V1 Pinout

Name	Function
ADC1 ₀₋₅ , ADC2 ₀₋₉	Used to measure analog voltage in the range of 0-3.3V.
DAC1, DAC2	Used for digital to analog conversion
GPIO ₀₋₃₉	Can be used as input or output pins but pins 34 to 39 can be used as input only.
T ₀₋₉	Capacitive Touch pins are used for capacitive pads
GND	Ground
A _{REF}	To provide reference voltage for input voltage.
RTCIO ₀₋₁₇	Used to wake up the ESP32 from deep sleep mode.
R _x , T _x	Used to receive and transmit TTL serial data.
All GPIO	can be use to trigger an interrupt and work as PWM
GPIO ₂₃ (MOSI), GPIO ₁₉ (MISO), GPIO ₁₈ (CLK), GPIO ₅ (CS)	Used for SPI-1 communication.
GPIO ₁₃ (MOSI), GPIO ₁₂ (MISO), GPIO ₁₄ (CLK), GPIO ₁₅ (CS)	Used for SPI-2 communication.
Micro-USB, 3.3V, 5V	Micro-USB: ESP32 powered through USB port 5V Regulated 5V can be supplied to this pin which is regulated to 3.3V 3.3V: Regulated 3.3V can be supplied to this pin to power the board.

Table 5.1: ESP32 DEVKIT V1 Pinout description

Since this projects needs to comply to the requirements, this development board is ideal due to its versatility to the application with minimal Printed Circuit Board (PCB).

5.1.2 Sensing Channel

The circuit in charge of processing the bioelectric signals present in the cardiac muscle to detect spontaneous cardiac activity is known as sensing channel and its components are detailed below being this solution explained in section 5.2.1.

5.1.2.1 Operational Amplifier (OPA350)

- Component Configuration

**OPA350: P, D, and DGK Packages
8-Pin PDIP, SOIC, and VSSOP
Top View**

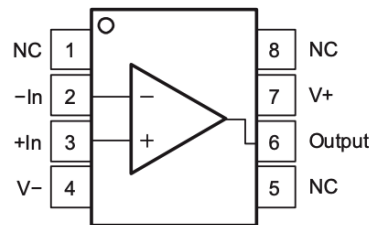


Figure 5.3: OPA350 Pinout

The OPA350 series of rail-to-rail CMOS operational amplifiers are optimised for low voltage, single-supply operation.

Name	Function
NC (1,5,8)	No Connection
-IN (2)	Inverting Input
+In (3)	Non Inverting Input
V- (4)	Negative power supply
V+ (7)	Positive power supply
Output (6)	Output

Table 5.2: OPA350 Pinout terminal description

- Control

In this case, the OPA350 series operational amplifiers are used and optimised for driving medium speed (up to 500 kHz) sampling Analog-to-Digital converters while offering excellent performance for higher speed converters. They also provide an effective means of buffering the input capacitance of the Analog-to-Digital converter and resulting charge injection while providing signal gain.

For optimum settling time and stability with high-impedance feedback networks, it may be necessary to add a feedback capacitor across the feedback resistor. This capacitor compensates for the zero created by the feedback network impedance and the input capacitance of the OPA350 (and any parasitic capacitance) [22].

5.1.2.2 Digital-to-Analog Converter (DAC7311)

- Component Configuration

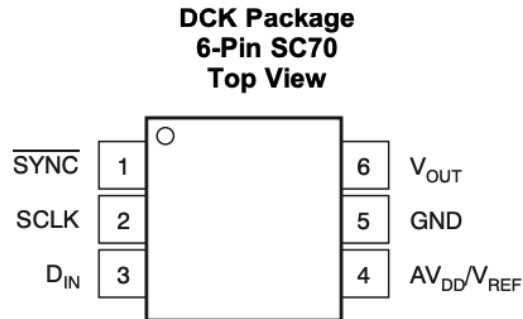


Figure 5.4: DAC7311 Pinout

Name	Function
\overline{SYNC} (1)	Positive power supply
SCLK (2)	Negative power supply
DIN (3)	Serial Data Input
AVDD/VREF (4)	Power supply input
GND (5)	Ground reference point
V_{OUT} (6)	Output

Table 5.3: DAC7311 Pinout terminal description

These converters are low-power 12-bit devices that have a single-channel output voltage.

- Control

The low power consumption of these devices in normal operation makes it ideally suited for portable, battery-operated applications such as a pacemaker. It uses a versatile, three-wire serial interface and is compatible with the serial peripheral interface (SPI) [23].

5.1.3 Stimulation Phase

The stimulation phase is accountable for the delivery of electrical stimuli to the heart.

When it comes to a pacemaker, the generation of precise and controlled voltage stimuli is essential and it is established in the industry that its resolution should be around 100mV.

5.1.3.1 Digital-to-Analog Converter (DAC8718)

- Component Configuration

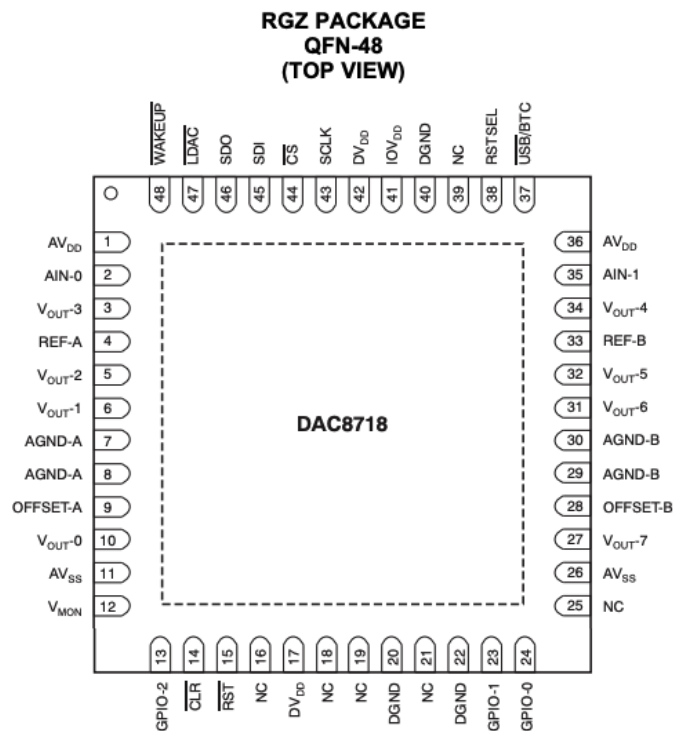


Figure 5.5: DAC8718 Pinout

The DAC8718 is a low-power, octal, 16-bit digital-to-analog converter produced by *Texas Instruments*.

This converter features a RGZ package QFN-48 (7x7mm).

This DAC provides low-power operation, good linearity, and low glitch over the specified temperature range of -40°C to $+105^{\circ}\text{C}$ [24].

- Control

This converter receives data from the ESP32 microcontroller through high-speed serial peripheral interface (SPI) and converts it into an analog signal.

5.1.3.2 CMOS Analog Switches (MAX4603EAE+)

- Component Configuration

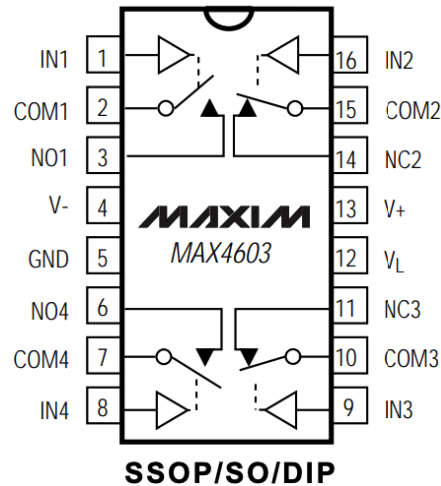


Figure 5.6: MAX4603 Pinout

This switch features a 16-pin SSOP package and operates from a single supply of +4.5V to +36V or from dual supplies of $\pm 4.5V$ to $\pm 20V$ and so, it is powered at 12V.

It is constituted of two NC (Normally Closed) and two NO (Normally Open) analog switches terminals. These analog switches are preferred over mechanical relays in applications where the current switching is required, being ideal in low-distortion applications [25].

Name	Function
IN1, IN2, IN3, IN4	Logic-Control Digital Inputs
COM1, COM2, COM3, COM4	Analog Switch Common Terminals
NO1, NO2, NO3, NO4	Analog Switch Normally Open Terminals
NC1, NC2, NC3, NC4	Analog Switch Normally Closed Terminals
GND	Ground
VL	Logic-Supply Input
V+	Positive Analog Supply Input
V-	Negative Analog Supply Voltage Input. Connect to GND for single-supply operation.

Table 5.4: MAX4603 Pinout terminal description

- Control

The switch is controlled through the microcontroller over the IN2-IN4 terminals whereas the COM1-COM4 terminals receive the signal from the digital-to-analog converter (DAC8718: V_{outA-D}) and the activation of both NO or NC switches is done with a digital input between 0 and 12V.

5.1.4 Power Supply

5.1.4.1 Battery: UBP053450/PCM

The battery chosen to power the circuit is UBP053450/PCM from *UltraLife*. It is a Li-ion battery with 3.7V of average voltage and 900mAh of capacity. Some features include small dimensions, weight, has a wide operating temperature range and can be assembled into packs.

It can be applied to portable electronics, medical equipment and tracking applications.

5.1.4.2 DC/DC Converter (LTC3400)

- Component Configuration

The LTC3400 is a synchronous, fixed frequency, step-up DC/DC converter. This converter is capable of outputting 2.5V to 5V and can work with a low start-up voltage (0.85V).

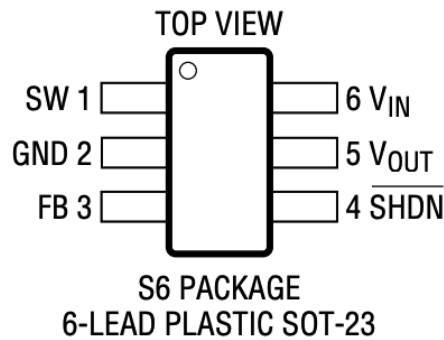


Figure 5.7: LTC3400 Pinout

Name	Function
SW	Switch Pin
GND	Signal and Power Ground
FB	Feedback Input to the Error Amplifier
\overline{SHDN}	Logic Controlled Shutdown Input
V_{out}	Output Voltage Sense Input and Drain of the Internal Synchronous Rectifier MOSFET
V_{in}	Battery Input Voltage

Table 5.5: LTC3400 Pinout terminal description

- Control

Current mode PWM control is used for exceptional line and load regulation. With its low $R_{DS(ON)}$ and gate charge internal MOSFET switches, the devices maintain high efficiency over a wide range of load current [26].

5.1.4.3 DC/DC Converter (LTC3436)

- Component Configuration

The LTC3436 is a monolithic boost switching regulator with a fast transient response and excellent loop stability. It also has high efficiency at high switching frequencies over a wide operating range maintaining consistent performance of Li-Ion batteries input systems.

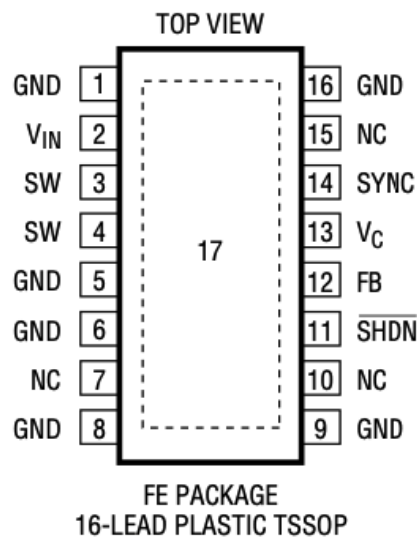


Figure 5.8: LTC3436 Pinout

Name	Function
SW	Switch Pin
GND	Signal and Power Ground
FB	Feedback Input to the Error Amplifier
\overline{SHDN}	Logic Controlled Shutdown Input
V_C	Output of the Error Amplifier
V_{in}	Power Input Voltage

Table 5.6: LTC3436 Pinout terminal description

- Control

The LTC3436 is a constant frequency, current-mode boost converter. This means that there is an internal clock and two feedback loops that control the duty cycle of the power switch. In addition to the normal error amplifier, there is a current sense amplifier that monitors switch current on a cycle-by-cycle basis. Also, output voltage control is obtained by using the output of the error amplifier to set the switch current trip point [27].

5.1.5 Detection Phase

5.1.5.1 Operational Amplifier (OPA350)

5.1.5.2 Optical Sensor

- SFH487

This infra-red emitter (photodiode) is a source of light energy in the infrared spectrum. It is a light emitting diode (LED) that is used in order to transmit infrared signals to be further detected by the photo transistor [28].

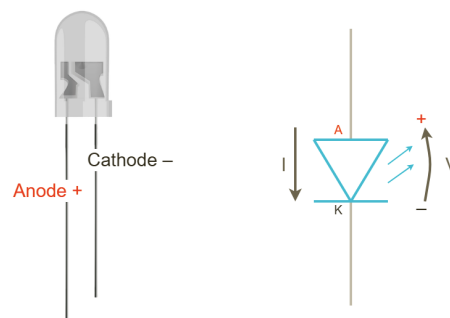


Figure 5.9: IR Emitter Diagram

- SFH309 FA

This component is a Silicon NPN phototransistor (receiver) with a spectral range of sensitivity 380 to 1100 nm and has high photosensitivity.

This transistor is a 2 or 3 terminal semiconductor device which converts the light energy into an electric current or voltage.

When the light is incident at the base of an NPN transistor, the base current develops and its magnitude depends on the intensity of the light incident over it.

The phototransistor amplifies the input light, and the output current is obtained from the collector of the transistor [29].

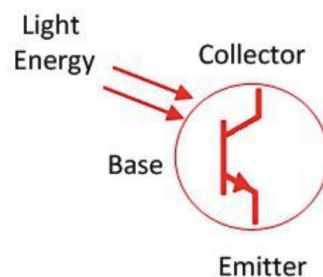


Figure 5.10: Photo Transistor Diagram

5.1.5.3 Analog-to-Digital Converter (ADS8661IPWR)

- Component Configuration

The ADS8661 is a 12-Bit ADC with Integrated analog-to-digital converter.

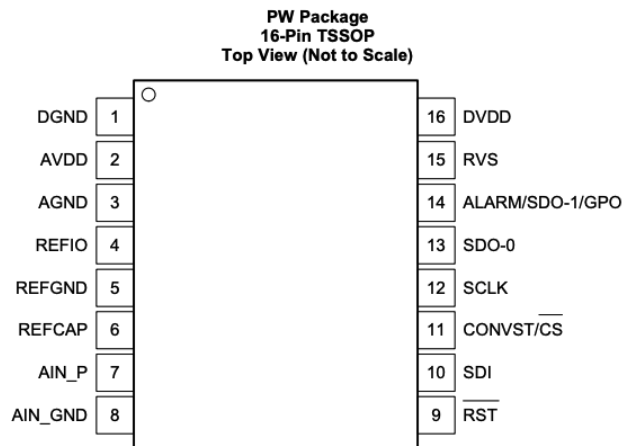


Figure 5.11: ADS8661 Pinout

Name	Function
DGND/AGND	Digital Ground pin/Analog Ground Pin
AVDD	Analog Supply Pin
REFIO	Internal Reference Output and External Reference Input Pin
REFGND	Reference Ground Pin
REFCAP	Reference Buffer Decoupling Capacitor Pin
AIN_P/ AIN_GND	Analog Input: Positive/Negative
DVDD/DGND	Digital Supply/Ground Pin
RVS	Multi-function Output Pin
ALARM/SDO-1/GPO	Multi-function Output Pin
SDO-0	Serial Communication: Data Output 0
SCLK	Serial Communication: Clock Input Pin
CONVST/ \overline{CS}	Dual-functionality Pin
SDI	Data input pin for serial communication and Chain Data Input
\overline{RST}	Active Low Logic Input to Reset Device

Table 5.7: ADS8661 Pinout terminal description

- Control

This device operates on a single 5V analog supply and support true bipolar input ranges of $\pm 12.288V$, $\pm 6.144V$, $\pm 10.24V$, $\pm 5.12V$ and $\pm 2.56V$ as well as unipolar input ranges of 0V to 12.288V, 0V to 10.24V, 0V to 6.144V and 0V to 5.12V.

The gain and offset errors are accurately calculated within the specified values for each input range to ensure high dc precision. The input range selection is done by software programming of the device internal registers.

The multiSPI digital interface is backward-compatible to the traditional SPI protocol [30].

5.2 Prototype Design

This section aims to analyse and clarify the stages of the assembly concerning to the model prototype through the use of the components depicted in section 5.1.

When assembling the system modules, it results in a more general diagram of the schematics of the prototype as shown in figure 5.12.

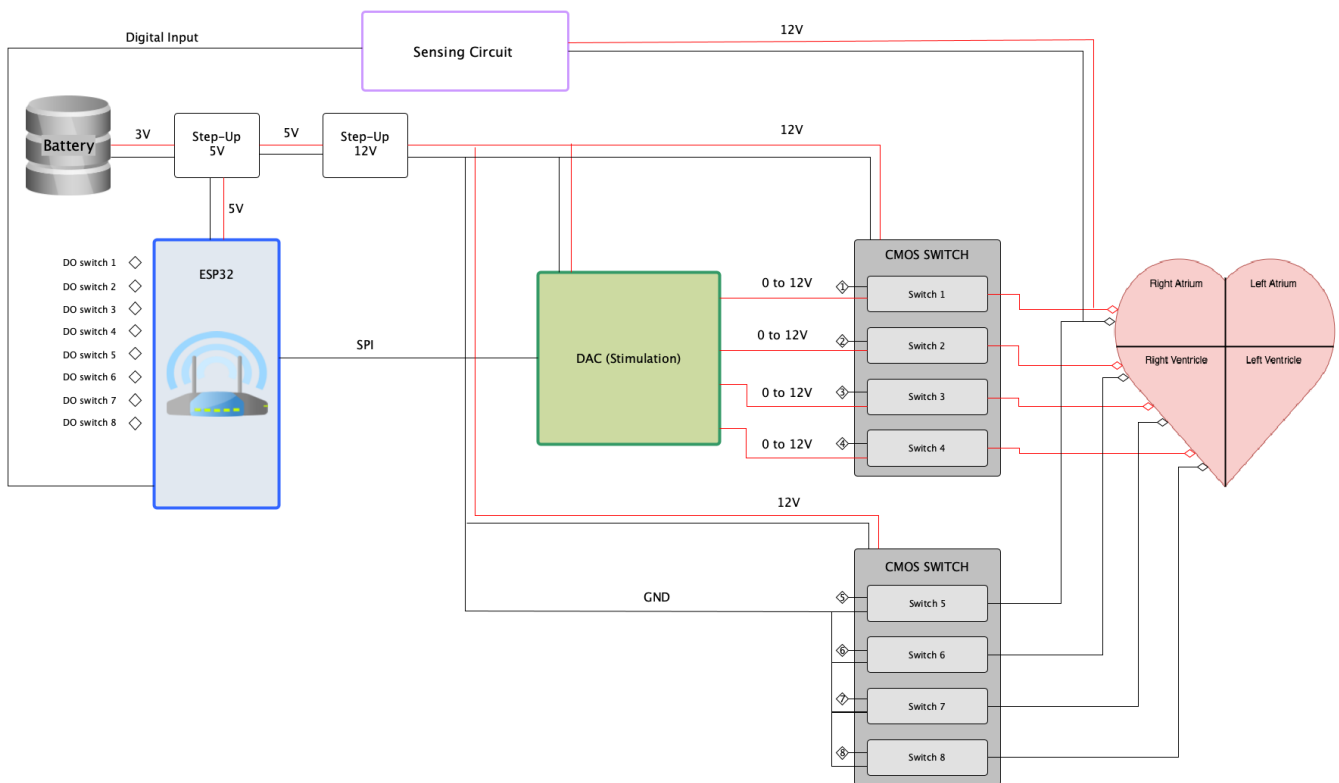


Figure 5.12: Circuit Diagram

The power supply voltage value is 3V and it is increased to 5V using a step-up converter. The step-up output voltage will power the microcontroller and will also be boosted to 12V. This output will power both the D/A Converter as well as the CMOS switches.

The control of the D/A Converter, as mentioned, is made through an SPI signal and the four outputs will connect to a single switch.

Note that the D/A Converter can have different voltage values for every output provided and is contained between 0 and 12V.

The output of the switch will be directed to the atria and ventricle depending on the intended signal coming from the CMOS and the purpose of the second CMOS switch is to connect each of the electrodes to their own ground in order to ensure that every stimulus is made in the correct electrode and not between different points.

5.2.1 Sensing Channel

In order for the pacemaker to stimulate based on a programmed timing interval, it must be able to sense intrinsic cardiac activity and can be inhibited from providing unnecessary/inappropriate stimuli to the heart.

Figure 5.13 shows the basic architecture for the Sensing Channel. It is extracted from "Low Power Analog CMOS for Cardiac Pacemakers: Design and Optimisation in Bulk and SOI Technologies" by Fernando Silveira and Denis Flandre [31].

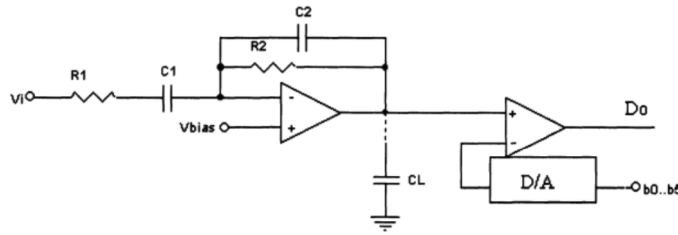


Figure 5.13: Sensing Circuit

The output of the filter (made by the first stage) is compared (in the second stage) with the programmable threshold set by the 5 bits DAC.

The C_L capacitor shown at the output of the first stage represents the parasitic capacitance. The V_{bias} voltage, which fixes the common mode voltage at the opamp input and the quiescent voltage at its output, is set at 5V.

The input to the DAC is fixed at 00111, setting the quiescent level at the input and output of the amplifier at reference voltage of the Digital-to-Analog converters.

One important parameter that characterises the influence of the opamp frequency response on the overall filter frequency response is the transition frequency f_t . It is set at $f_t = 160\text{kHz}$. For that the component values are: $R_1 = 22\text{k}$, $C_1 = 47\text{nF}$, $R_2 = 22\text{M}$, $C_2 = 15\text{p}$.

V_i is the signal that comes from the Atria through the lead selection blocks and D_o is the Digital Output that goes into the microcontroller to be processed.

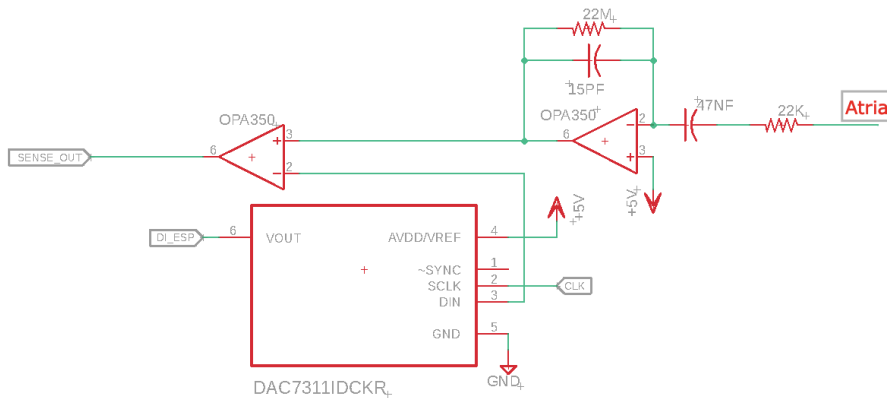


Figure 5.14: Sensing Schematics

5.2.2 Stimulation phase

A dual chamber pacemaker with electrodes in both the atrium and ventricle delivers the sequential atrial and ventricular signals to approximate the timing of the normal heartbeat. This device also senses intrinsic atrial and ventricular depolarisation, and delivers stimuli at the appropriate time to maintain proper chamber synchronisation.

The stimulation phase circuit is composed of the Digital-to-analog converter (subsection 5.1.3.1), CMOS switch (subsection 5.1.3.2) and the electrodes connected to the heart.

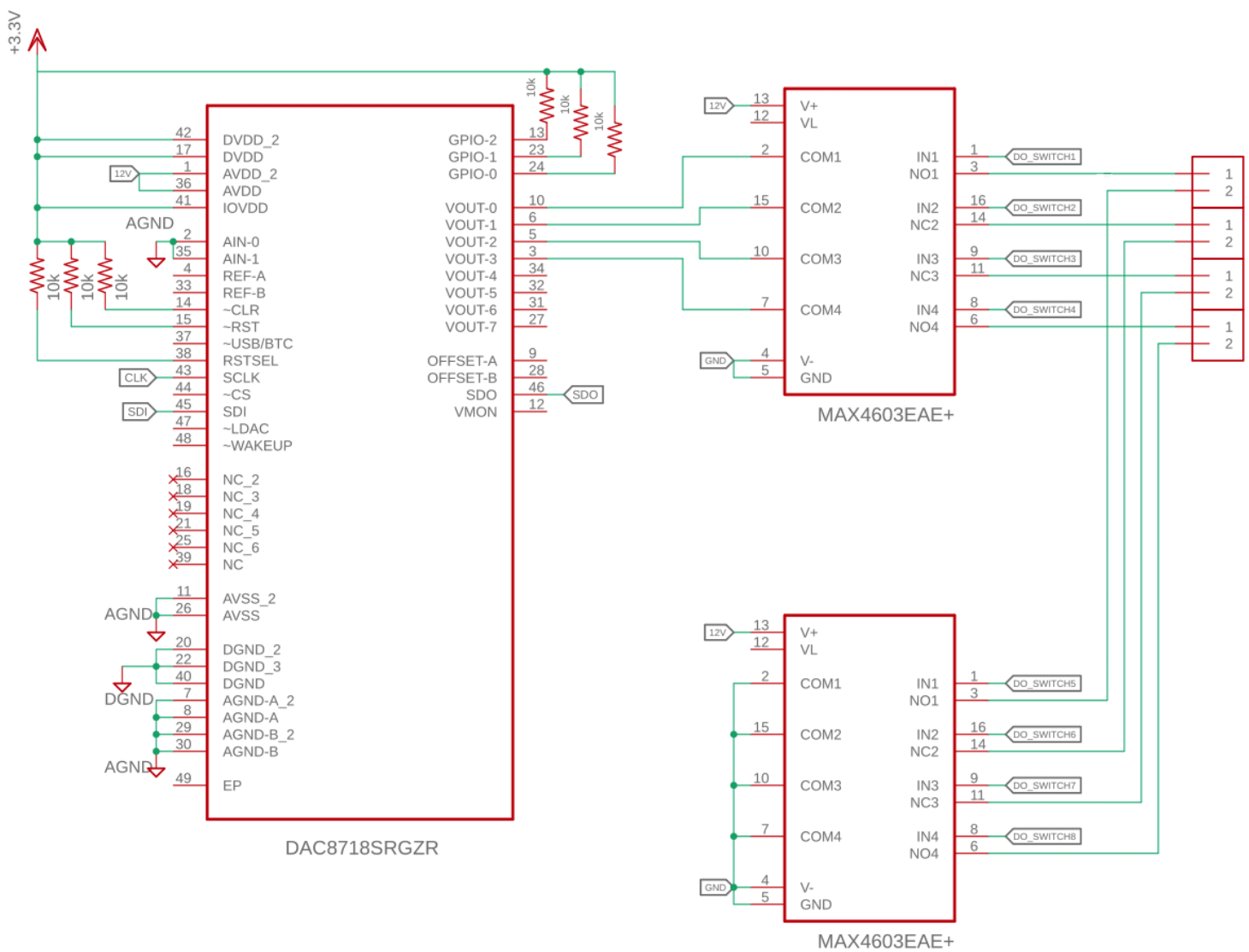


Figure 5.15: Stimulation Phase Schematics

5.2.3 Power Supply

Battery power systems often assemble cells in series to achieve higher voltage. However, when assembling sufficient cells is not possible in many high voltage applications due to lack of space. Boost converters can increase the voltage and reduce the number of cells.

In order to supply electric power to the electrical load (composed of the circuit and micro-controller), a battery is connected to a boost converter that steps up voltage from its input (3V) to its output load (5V) (subsection 5.1.4.2) to power the microcontroller. Another boost converter stepping up voltage from its input (5V) to its output (12V) is connected to the previous boost to power the D/A Converter and the CMOS switches (subsection 5.1.4.3).

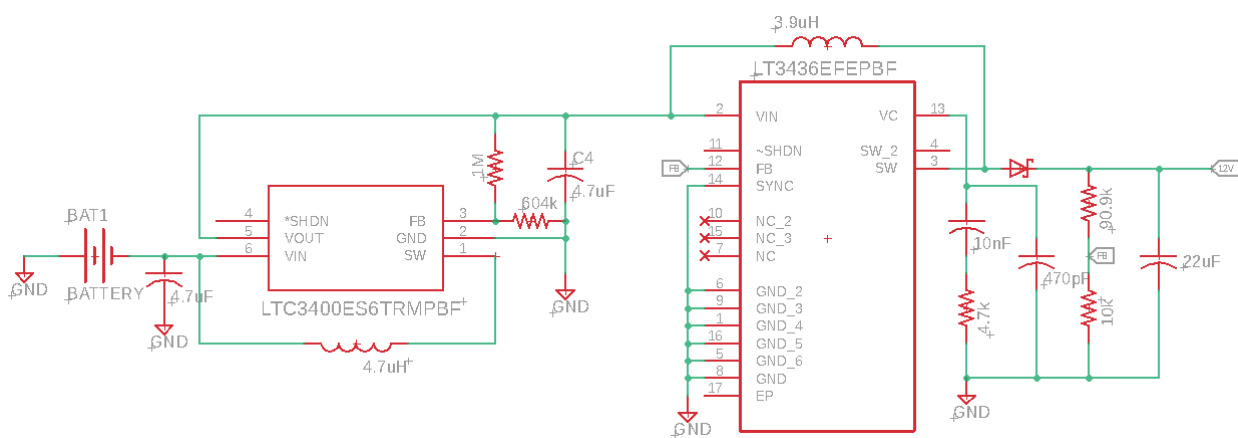


Figure 5.16: Power Supply Schematics

5.2.4 Detection Phase

The detection phase consists of the analog acquisition and treatment of the cardiac pulse signal in order to proceed to its visualisation in the form of pulsations per minute.

With the addition of the ECG circuit to the prototype, it is possible to add three more electrodes in the surface of the heart (subcutaneously) and it is responsible for the surface measurement of the electrical potential generated by electrical activity in cardiac tissue.

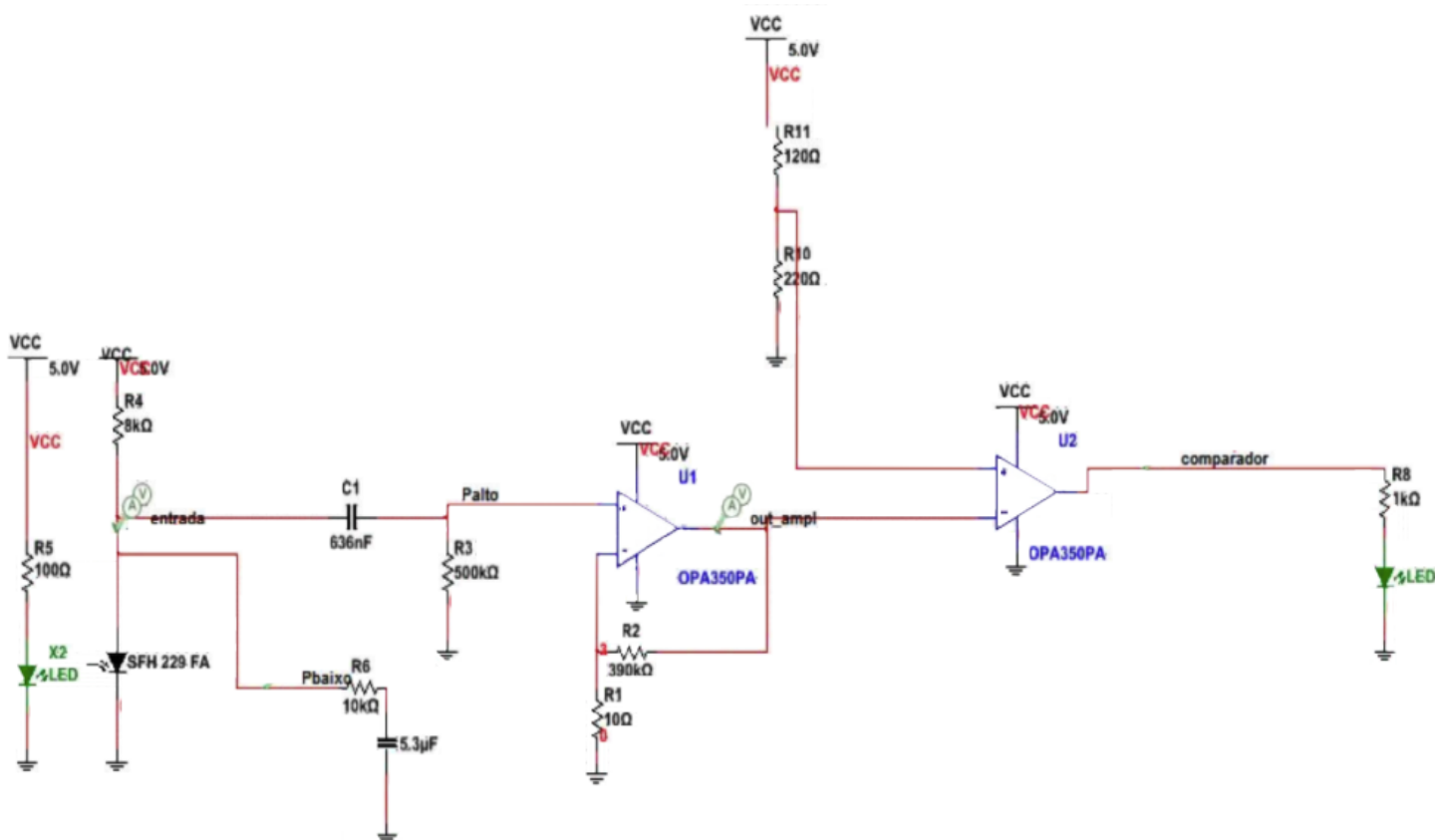


Figure 5.17: ECG Schematics

The implemented circuit consists of an optical sensor, a low pass filter, an active high pass filter with gain 40, a comparator (with 1.5V offset) and a led.

The optical sensor is composed of a diode for the emission of radiation and a photo transistor for receiving it, after reflected on the surface in question.

When acquiring, the received signal contains a continuous and an alternating component. It also contains atmospheric interference (for example, light) that translates into unwanted noise. All these variables must be eliminated if an accurate reproduction of the signal is desired.

5.2.5 Model Assembly

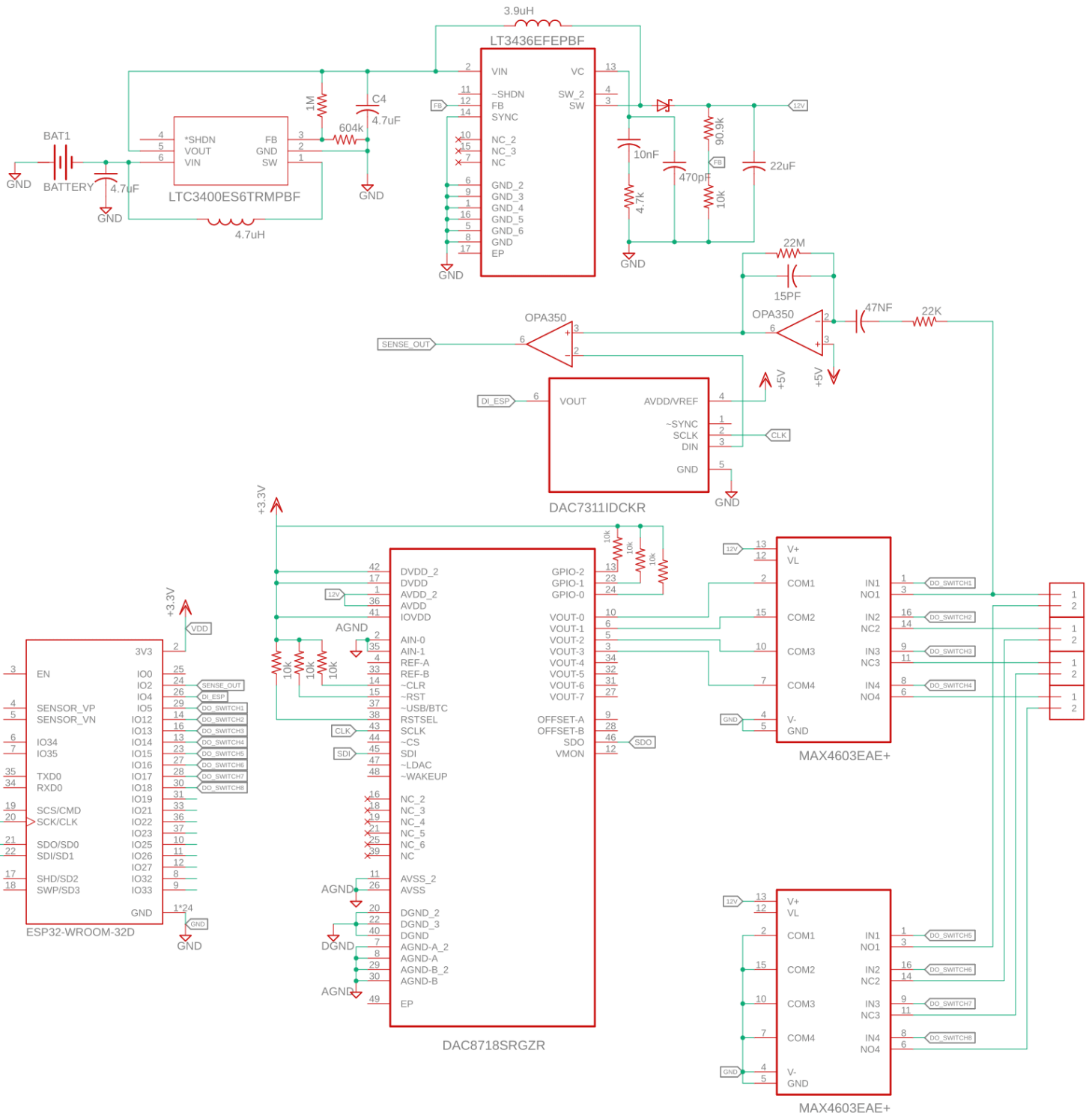


Figure 5.18: EAGLE Schematics

The circuit diagram of the assembly of the mentioned modules is completed in this section.

After processing the bioelectric signals present in the cardiac muscle, the sensing channel circuit will detect spontaneous cardiac activity and inform the microcontroller of the occurred.

It is important to note that in this diagram, the detection phase (ECG scheme) is not included because it would add higher complexity to the main circuit and so, it is tested separately.

The depicted capacitors are electrolytic, but in the real model prototype, they mustn't be electrolytic since this type of capacitors is polarised and its value will normally start from 1 μF whilst non electrolytic capacitors are non polarised and the value starts from ρF ranges.

Chapter 6

System Description: Software

A control program executed on a microcontroller embedded in a pacemaker is responsible for implementing the control functions of the device and thus this section aims to implement both the high-level and low-level control of the application.

6.1 Communication Protocol: TCP/IP

In order to allow a bidirectional data transmission between the control module and the graphical interface via Wi-Fi, a communication protocol must be set [32].

In this case, TCP/IP (*Transmission Control Protocol/Internet Protocol*) is a set of communication protocols used to interconnect network devices on the internet and can also be used as a communications protocol in a private computer network and is normally considered to be a four layer system:

- **Application Layer:**

This is the top layer of TCP/IP protocol suite. Provides applications with standardised data exchange (e.g. HTTP, FTP, SMTP);

- **Transport layer:**

Responsible for maintaining end-to-end communications across the network (e.g. TCP, UDP);

- **Network layer:**

Responsible for organising Data over the network (e.g. IP, ICMP);

- **Data link layer/Physical Layer:**

This is the lowest layer of TCP/IP protocol suite. Consists of protocols that operate only on a link (the network component that interconnects nodes or hosts in the network) (e.g. Ethernet, LAN, ARP);

6.1.1 How does TCP/IP protocol work?

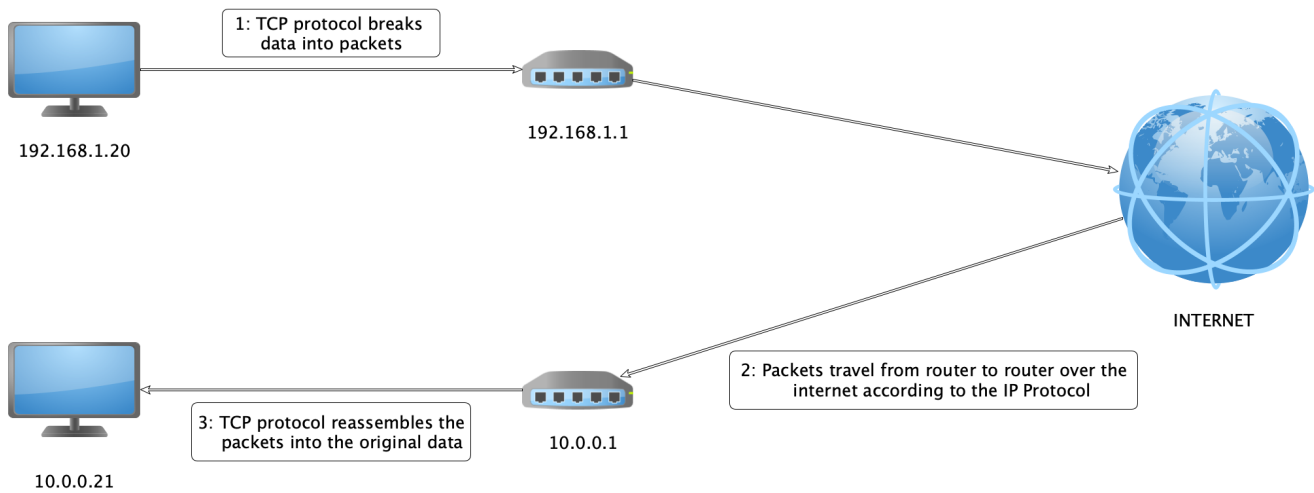


Figure 6.1: TCP/IP Protocol

The TCP protocol starts by breaking data into smaller packets where they, individually, take the quickest route and transfer it through the network.

To make sure the receiving computer can put the packets back together properly, TCP uses a header that contains instructions on what order to reassemble the packets as well as error checking information so that the receiving computer knows if the packet arrived without any missing data.

After this, the packets are pushed on to the internet layer which uses the IP to attach both the origin and the destination IP addresses. The data is then sent over the network layer which handles objects such as MAC addressing (so the packet goes into the correct physical machine) as well as converting the data in electrical impulses.

Packet switching makes the internet faster since it allows each packet to individually avoid congestion that would occur if all the data had to travel along the same route during each session.

TCP also deals with packets from all the computer applications meaning that the waiting times for an application to run are not too time consuming [33].

The same happens when using the ESP32 to communicate with the PC with the exception that the microcontroller can only be accessed locally. This means that objects within range of the ESP32's Wi-Fi can access it. However, if a router is added and the ESP32 is configured as a client, it is possible to broadcast a page to the wider internet.

With this, the application is only accessed locally (router not used) and it is possible to exchange messages between the ESP32 (client) and the website through the PC server which will then receive the data and store it in a database for further use.

This process will be explained in the following section.

6.2 Client and Server

After dwelling on the theoretical concepts analysed in the previous sections, the process in which the communication was achieved is depicted in this section.

When analysing the diagram of the chapter 4.2, there are bidirectional interactions between server and client that must be thoroughly explained. The following diagram shows an overview of the system concept with a deepened notion of this phase of the project.

The previous section was also added to the diagram in order to understand the connections between hardware and software.

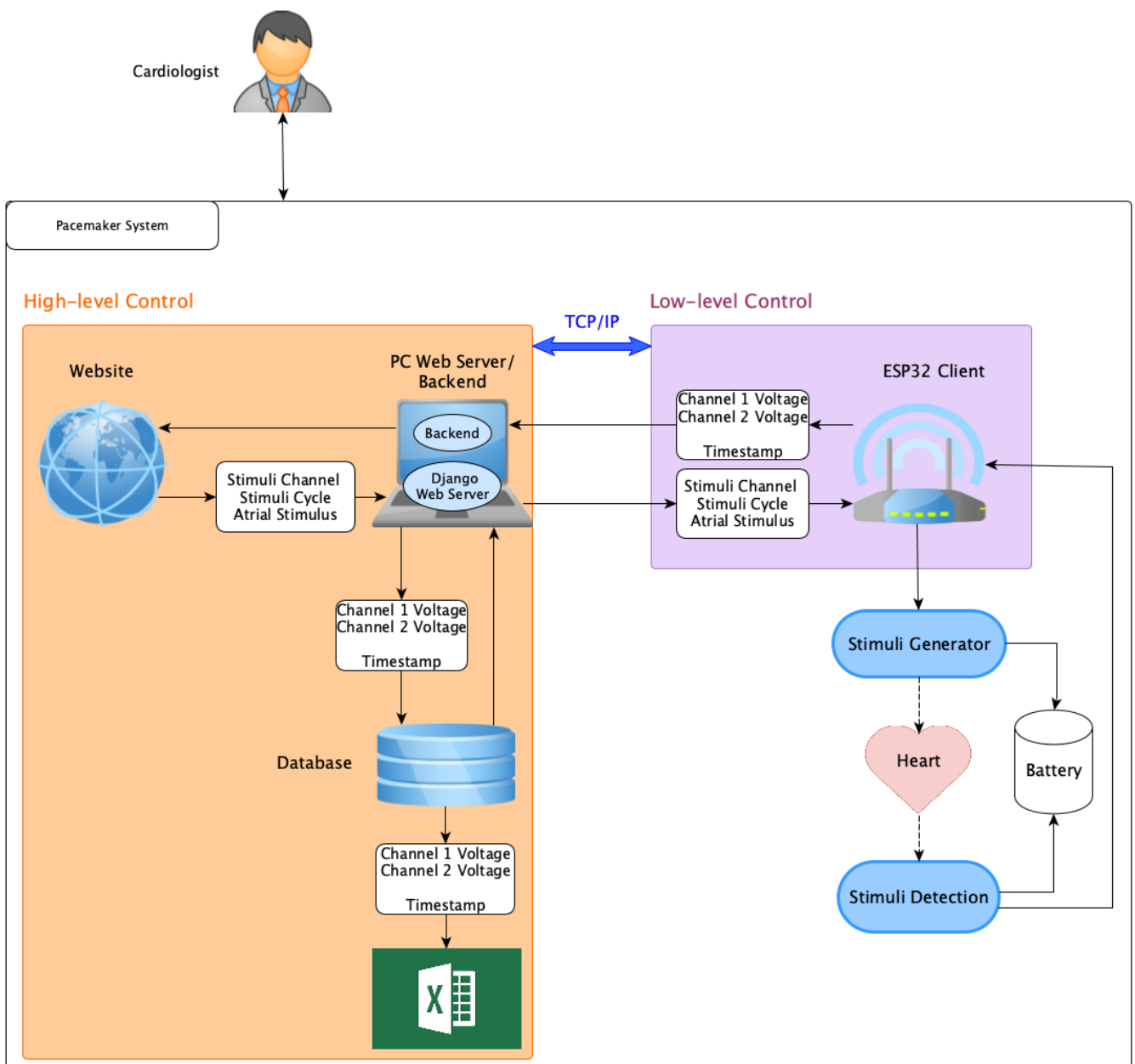


Figure 6.2: Concept Diagram: Bidirectional Communication

As visualised, the Low-level Control is responsible for the interactions between the ESP32 Client with the Backend (TCP/IP) and the High-Level Control is responsible for the interactions between the Website and the Webserver (HTTPS).

The cardiologist controls the pacemaker device through the monitor unit (web interface). The data is then transferred to the server and consequently received by the client. On the other hand, the client receives data from the hardware module and sends it to the server (backend). This data will be stored in a database which can then be exported to an excel file.

In order to send and receive data from the client to the server and vice-versa, a potentiometer (10k Ω) and a Led connected to a button were used to test these functionalities. The sole purpose of these components were to demonstrate the bidirectional communication assembled as depicted in figure 6.3.

When using the potentiometer, two voltage values are sent (Voltage_Channel1 and Voltage_Channel2) as well a Timestamp to save at what time those values were registered.

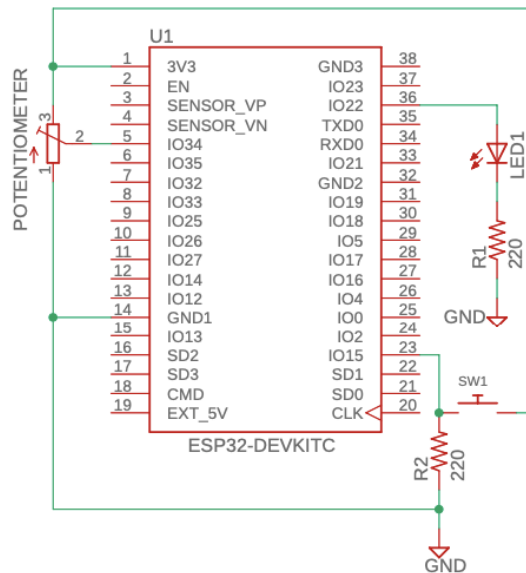


Figure 6.3: Bidirectional Communication: Testing Circuit Schematics

Before further explanation, the used technologies throughout the project must be cautiously designated. The phases of the developed code are also enumerated and explained in order to allow better comprehension on the subject. After this, a preview of the interface is demonstrated.

6.3 Used Technologies

6.3.1 Arduino Software

The Integrated Development Environment used is **Arduino Software (IDE)**. This platform is also open-source and free to use. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

In order to select the microcontroller in which the code is to be transferred, the ESP32 option must be chosen in "Tools > Board > DOIT ESP32 DEVKIT V1" as shown in Figure 6.4. It is important to note that this function may change according to the ESP module at hand.

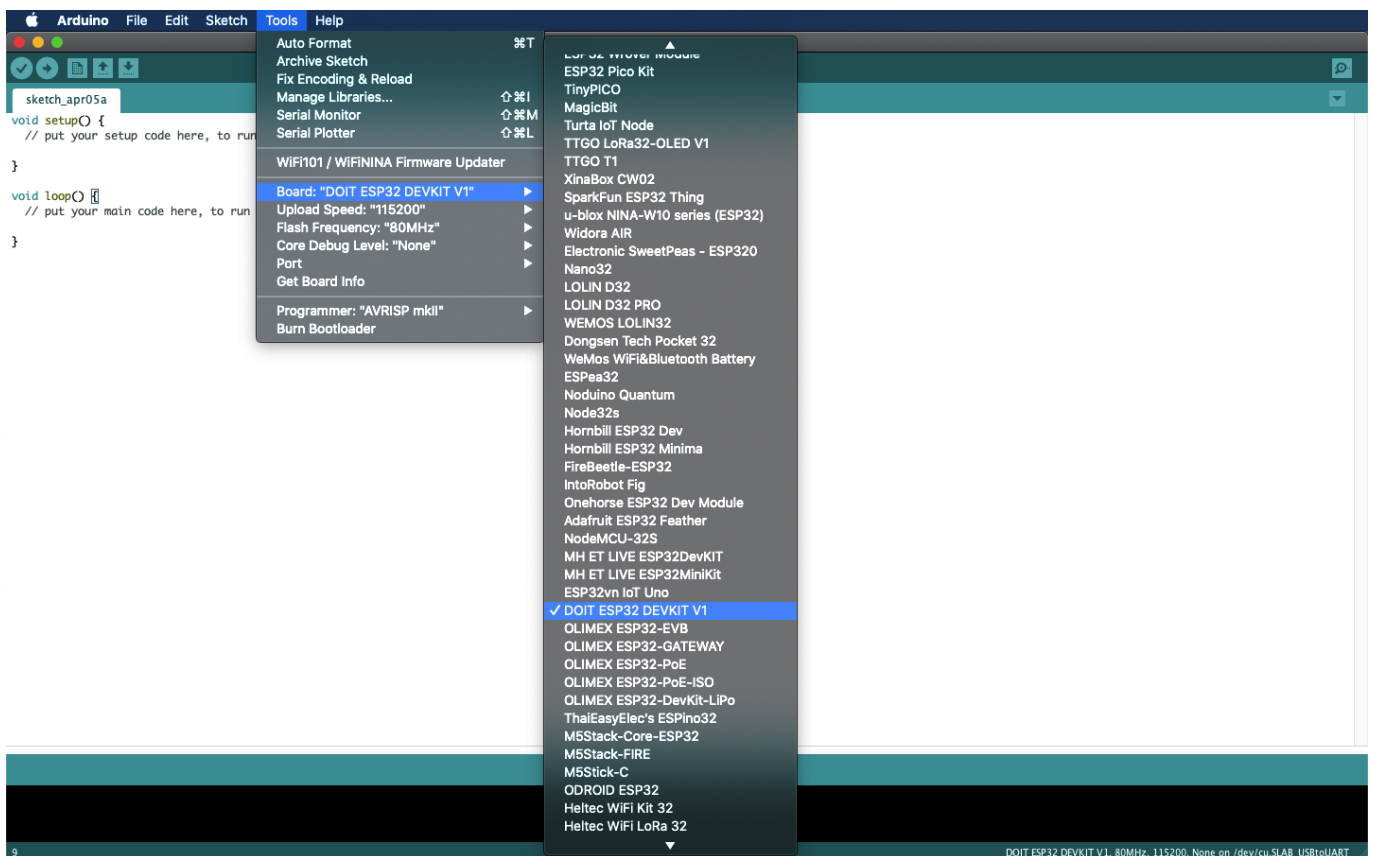


Figure 6.4: Arduino Software (IDE)

This Software will be used to create the ESP32 client.

6.3.2 Python Software

- **Python IDE:**

Similarly to Arduino IDE, **Pycharm** is an integrated development environment (IDE) used in computer programming, specifically for the **Python language**.

- **Web Framework:**

In order to respect the defined clients needs (chapter 4.1), in this case, N2, a web framework was defined being this the **Django Framework**.

Web frameworks are a key technology because they provide a standard way to build and deploy web applications on the *World Wide Web* which means that it can provide libraries for database access, templating frameworks, session management and promote reuse of code. Other example of a web framework is Flask.

The following table states some of the differences between Django and Flask.[34, 35, 36]

	Django	Flask
Performance	<ul style="list-style-type: none"> – good results – used by high-traffic websites 	<ul style="list-style-type: none"> – good results – used by high-traffic websites
Packages	<ul style="list-style-type: none"> – Over 4000 built-in packages (easier to find a package to build and run an application with less effort) 	<ul style="list-style-type: none"> – Minimalistic and with no restrictions (possibility to implement functionalities with external libraries) – Flexible and extensible
Community	<ul style="list-style-type: none"> – Huge and active developer Community – Easy to find answers online 	<ul style="list-style-type: none"> – Small Community – Takes longer to find answers online
Security	<ul style="list-style-type: none"> – Has options to protect applications from Cross-site scripting, Cross-site request forgery and SQL injection 	<ul style="list-style-type: none"> – Flask-Security library provides almost the same mechanisms as Django to prevent data leaks and other web attacks.
Use cases	<ul style="list-style-type: none"> – Designed for fast development of complex web apps – Allows developers to implement scalable and maintainable functionality 	<ul style="list-style-type: none"> – Allows developers to create smaller apps faster

Table 6.1: Django Vs Flask

After analysing the differences between both of these web frameworks, the Django Framework was chosen over Flask Framework due to its beginner friendlier environment and characteristics.

- **Database:**

The database used was **SQLite3** because it is already integrated in the Django Framework, though it is possible to use other type.

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. With it, it's possible to prototype an application and then port the code to a larger database such as PostgreSQL or Oracle.

- **Data Format:**

JavaScript Object Notation or **JSON** is a lightweight data-interchange format inspired by the object literals of JavaScript being also language independent. This data format was chosen as an alternative to using XML for transferring data between software components making it easier to transfer data from client to server or vice-versa.

- **Python Libraries:**

Description	Python Libraries
Export Data to Excel	py2xlsx
Read Data Format	json
Get Data Timestamp	time
Matrix Format / Mathematical Calculations	pandas
Information on the Python Interpreter	sys
Database	sqlite3

Table 6.2: Python Libraries

6.4 Software Development Modules

6.4.1 Client Side

1. Client Creation ("client.ino")

In order to Create the client, the arduino IDE is used.

Setting up web a server on ESP32 requires little code and is very straightforward. This is thanks to functionality provided by the versatile ESP32 WiFi library that is included in the beginning of the code.

With this, a connection must be established to the local network by setting the SSID and password credentials. A "host" and "port" number are also set being these referred to the ESP32 device credentials.

```
void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);
  pinMode(PB_PIN, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("...");
  }

  Serial.print("WiFi connected with IP: ");
  Serial.println(WiFi.localIP());

  wifiServer.begin();
}
```

"Serial.begin()" function sets the data rate in bits per second (baud) for serial data transmission and the testing led and potentiometer are set as output and input respectively.

The "setup()" function will initialise the WiFi library's network settings and provide current status.

Following this, the connection is checked and when connected, the WiFi IP is printed. This IP will later be used to set a path in order to send data to the esp as shown.

The complete "setup()" function is depicted in appendix [A.1.1](#).

2. Send Data from ESP32 to Server

In order to send data, a connection to the server, in this case to the backend.py, must be established first.

The data format used to send the data received from the potentiometer is JSON.

The JSON format allows the simple parsing between systems and is independent of programming languages.

This is depicted in "client.print()" functions as shown in the following code snippet.

```
//Send data to backend.py
Serial.println("----- Sending Data to backend.py -----");
Serial.print("voltage Channel1: ");
Serial.println(potValue);

Serial.print("voltage Channel2: ");
Serial.println(potValue/2);

client.print("{\"voltage_channel1\": ");
client.print(potValue);
client.print(", \"voltage_channel2\":");
client.print(potValue/2);
client.print("}");

Serial.println("----- Disconnecting from backend.py... -----");
client.stop();
```

The complete "send_data()" function is depicted in appendix [A.1.2](#).

3. Receive Data from Server

Similarly to [2](#), receiving data from the server may also occur.

In the "receive_data()" function, a vector ("buffer") is created. A connection to the server is verified and if there's an available client, data reception may proceed. In here, the received data is written into the buffer and it's possible to analyse it in the Arduino IDE serial monitor.

It is important to bind a server to the correct address so that clients can communicate with it. In the developed application, the localhost (127.0.0.1) was used as the IP address, which limits connections to clients running on the same server.

The complete "Receive_data()" function can be found in Appendix [A.1.3](#).

After completing these steps, a "loop()" function must call these functions:

```
void loop() {
  receive_data();
  send_data();
}
```

6.4.2 Server side

1. Server Creation ("backend.py"):

The primary socket API functions and methods used in this module are `socket()`, `bind()`, `listen()`, `accept()`, `connect()`, `recv()` and `close()`.

The following diagram demonstrates the data flow for TCP and sequence of socket API calls:

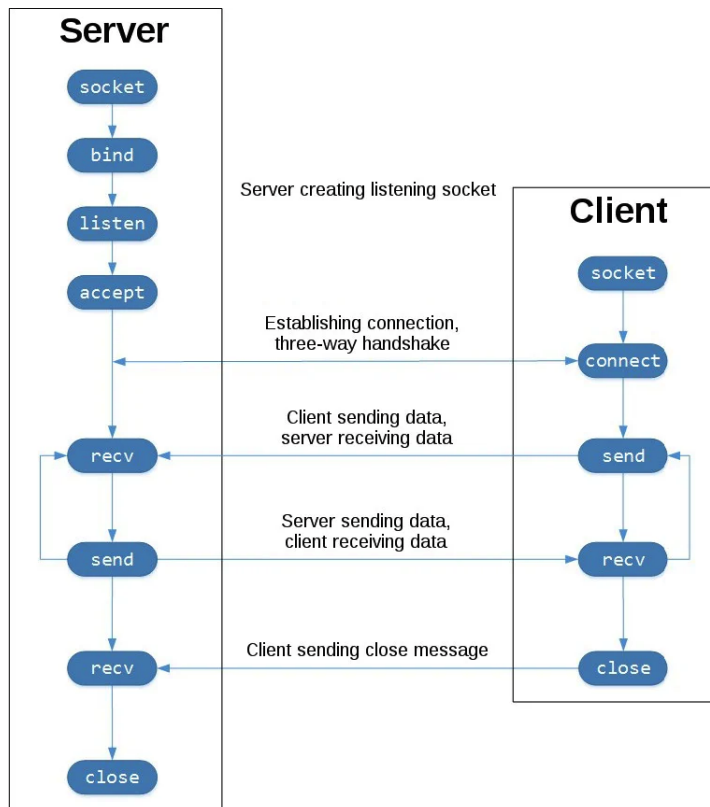


Figure 6.5: Server Side: Data flow for TCP

Looking at the server side, the first four calls represent the process of listening for connections from clients. When a client connects, the server calls `accept()` to accept, or complete, the connection.

The client calls `connect()` to establish a connection to the server and initiate the three-way handshake. The handshake step ensures that each side of the connection is reachable in the network.

The next step is the round-trip section, where data is exchanged between the client and server using calls to `send()` and `recv()` [37].

Finally, the client and server `close()` their respective sockets and so, the *backend.py* file can be found in Appendix A.2.1.

2. Database Creation:

Due to the need to generate a report (excel file) with data received from the microcontroller and also for further addition of certain functionalities such as security credentials, it is crucial to create a database. The database queries data, allows the management of stored data and also manipulates it.

The `sqlite3` module was written by Gerhard Häring and to use it, a connection object that represents the database must be created:

```
conn = sqlite3.connect('name.db')
```

Once the connection is achieved, a cursor object can be created and call its `execute()` method to perform SQL commands:

```
c = conn.cursor()
c.execute()
```

After these commands, the changes must be saved and the connection closed:

```
conn.commit()
conn.close()
```

The code for the creation of the database can be found in Appendix [A.2.2](#).

3. Receive data from client and write to Database

Since the Database is created, now data must be written to it.

First, a name to the database must be given and a connection to the database must be established similarly to "Database Creation".

Once achieved this connection, the server connects to the defined database and uses SQL statements to insert previously received data from the client (subsection [6.4.1](#)) into it.

After these phases, the changes must be saved and the connection closed as well.

The code can be found in Appendix [A.2.3](#).

6.4.3 Interface reads Data from Database

Finally, in order for the interface to read data from the Database, a `read_all()` function must be built.

The first steps are also very similar to the previous subsection: connection to designated database and creation of a cursor object to call the `execute` method in order to allow the performance of SQL commands. In this case, data must be selected and ordered by "timestamp".

An array called "row" is created so all set parameters (also arrays) can be added to it. When using the `append()` method, an element is added to the end of each list of the said parameters.

The code can be found in Appendix [A.3](#).

6.5 Graphical Interface

The high-level control is possible through the interaction between the user and the designed web interface which is described in this section.

6.5.1 Home Page

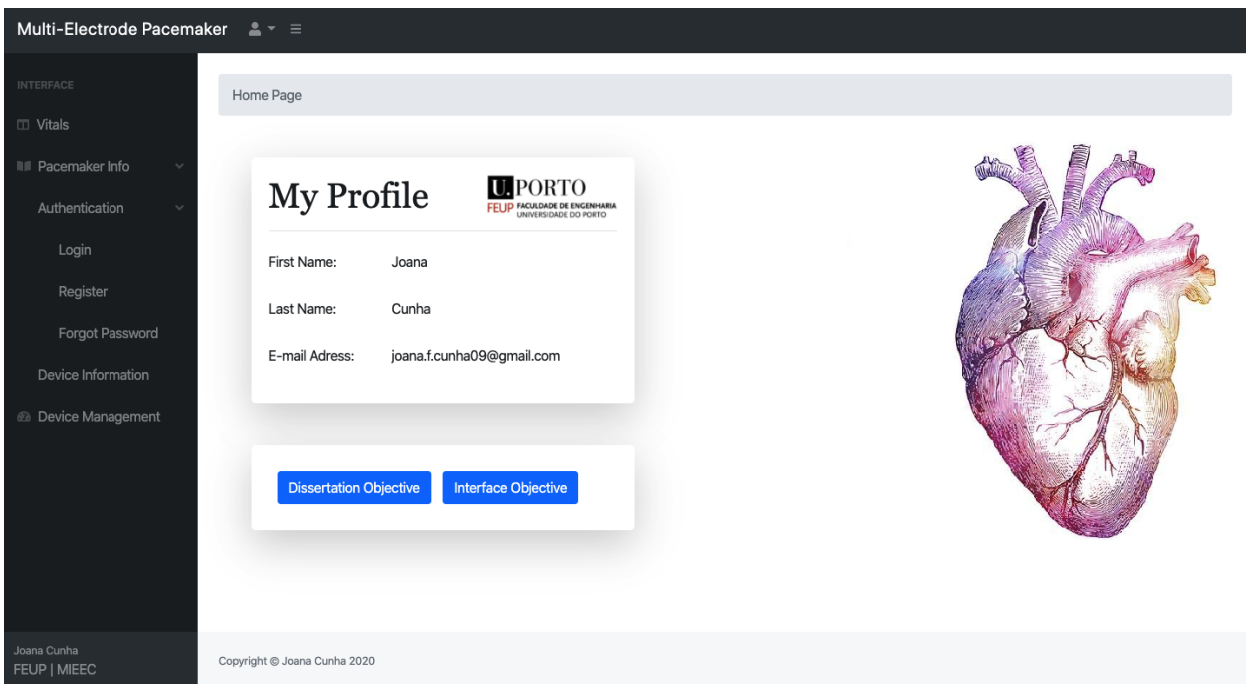


Figure 6.6: Home Page

The profile card shows the user credentials logged in, such as first name, last name and e-mail address and the sidebar displayed on the left of the page allows for the user to navigate between functionalities on the interface.

There are also two buttons in which a pop-up emerges stating both the Dissertation Objective as well as the Interface Objective.

With the addition of a user database, it would be possible to add different users through Login and Register pages (depicted in the Authentication section 6.5.3).

In this project, a user database was not fully implemented having only a super-user whose credentials are stated in the "My Profile" card.

6.5.2 Vitals

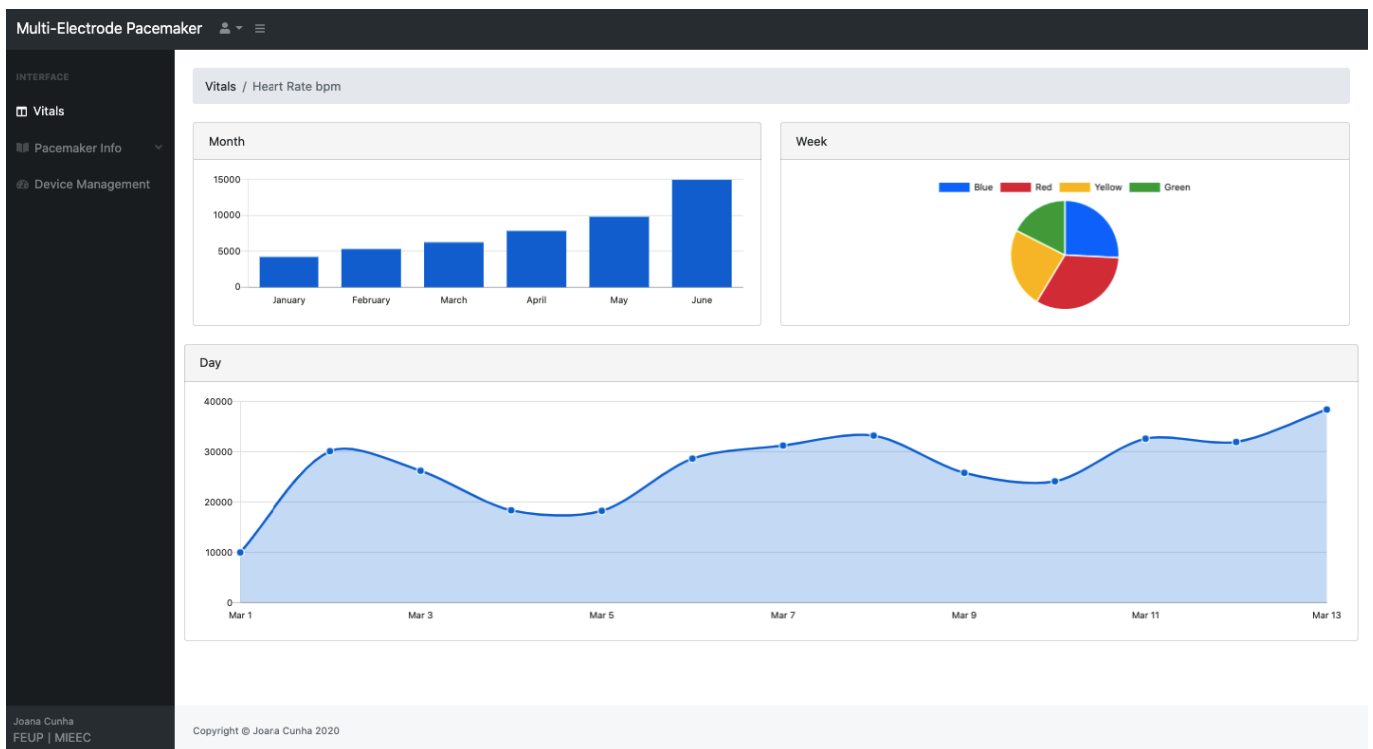


Figure 6.7: Vitals Page

The intent of the Vitals section is to show real-time data collected from the prototype depicting it in graphs of day, week and month.

This section was only designed and not completely implemented though it would be interesting to add in a future pacemaker module.

There are already some companies implementing this section in their android applications such as "Medtronic©: My Care Link Heart APP" which has a tracking system implemented and displays data in real-time.

6.5.3 Pacemaker Info

This subsection consists of two fields: Authentication and Device information.

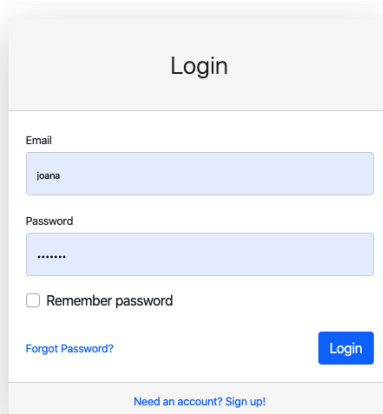
Referring to the Authentication field, there are three sections: "Login", "Register" and "Forgot Password."

The intent of these pages is to allow security credentials and also access to personal data without consulting a specialist and thus, the "Register" page allows a new user to register their credentials in the database, the "Login" page allows the previously registered user to access his personal data and finally, the "Forgot Password" page, allows the restoration of a user password.

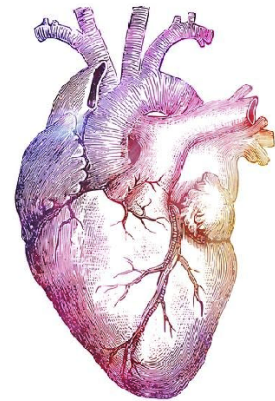
On the other hand, in the "Device Information" page, the prototype design and circuit schematics are presented.

– Authentication

- Login



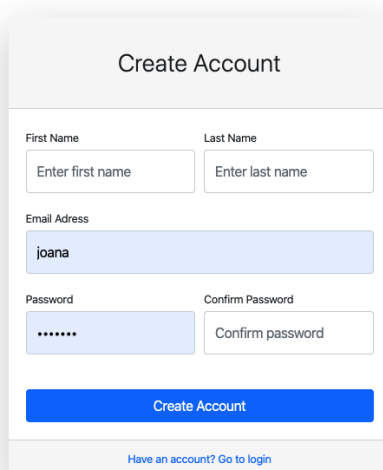
The screenshot shows a 'Login' form with a light gray header. Below the header, there are two input fields: 'Email' with the text 'joana' and 'Password' with masked characters '.....'. A checkbox labeled 'Remember password' is unchecked. A blue button labeled 'Login' is positioned to the right of the password field. Below the form, there is a link 'Forgot Password?' and a footer link 'Need an account? Sign up!'.



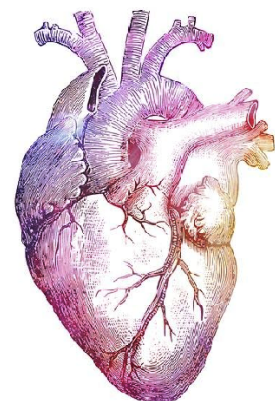
Copyright © Joana Cunha 2020

Figure 6.8: Login Page

- Register



The screenshot shows a 'Create Account' form with a light gray header. Below the header, there are four input fields: 'First Name' (placeholder 'Enter first name'), 'Last Name' (placeholder 'Enter last name'), 'Email Address' (text 'joana'), and 'Password' (masked '.....'). A 'Confirm Password' field (placeholder 'Confirm password') is located to the right of the password field. A large blue button labeled 'Create Account' is centered below the fields. At the bottom, there is a link 'Have an account? Go to login'.



Copyright © Joana Cunha 2020

Figure 6.9: Register Page

- Forgot Password

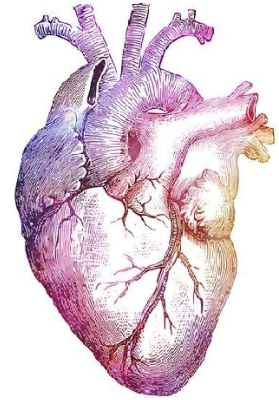
Password Recovery

Enter your email address and we will send you a link to reset your password.

Email

Return to login
Reset Password

Need an account? [Sign up!](#)



Copyright © Joana Cunha 2020

Figure 6.10: Forgot Password Page

– Device Information

Multi-Electrode Pacemaker

INTERFACE

- Vitals
- Pacemaker Info
- Authentication
- Device Information**
- Device Management

Joana Cunha
FEUP | MIEEC

Device Information

Prototype Design

Schematics

Power Supply
Sensing Channel
Stimulation Phase
Complete Scheme

Copyright © Joana Cunha 2020

Figure 6.11: Device Information Page

6.5.4 Device Management

Figure 6.12: Device Management Page

According to the concept diagram demonstrated in section 6.2, the Device Management page is where the interactions between the website and the server occur.

The first card is referring to the data to be sent from the website to the server (Stimuli Channel, Stimuli Cycle and Atrial Stimulus Reading Interval).

The second card is referring to the data received from the client that was previously stored in the database. In here the data coming from the client, can be exported to an excel file.

A third card is also used in order to give information to the user on how to send data to the server.

In order to pass input state through the system, Django uses request and response objects. This framework provides a range of tools and libraries to build forms to accept input from site visitors, and then process and respond to the data input. In this case, the APIs used were "HttpRequest" and "HttpResponse".

6.5.4.1 Data Input: HttpRequest and HttpResponse

When a page is requested, Django creates an `HttpRequest` object that contains metadata about it. Django then loads the appropriate view, passing the `HttpRequest` as the first argument to the view function. Each view is responsible for returning an `HttpResponse` object.

Since these requests were built using forms, the GET and POST are the only HTTP methods used.

Django's login form is returned using the POST method where the browser stacks the form data, encodes it for transmission, sends it to the server, and then receives back its response. On the other hand, GET stacks the submitted data into a string, and uses this to compose an URL that contains the address where the data must be sent, as well as the values ("management" in `views.py`). After this, the input data is sent to the client ("send2esp" in `views.py`) [38] and so with this, 3 types of files were created:

- "views.py"

Python function that takes a Web request and returns a Web response. It contains whatever arbitrary logic is necessary to return that response;

- "forms.py"

Python function where the elements of the form are defined;

- "management.html"

Where the site is built and the previous functions are called.

Note: Snippets of the "views.py", "forms.py" and "management.html" code can be found in Appendix B.1;

6.5.4.2 Data Output: Export Data

As previously mentioned, when receiving data from the client, it gets stored in the database. In the Device Management page, an export button exists in order to download these values to the local computer into an excel file.

This becomes possible when using:

- "export_to_excel.py"

This file reads all data present in the server and frames it in a table similarly to an excel table ("`DataFrame()`"). An API is then called in order to create the file extension pretended ("`variable.to_excel()`").

- "views.py"

- "management.html"

Note: Snippets of the "views.py", "export_to_excel.py" and "management.html" code can be found in Appendix B.2;

Chapter 7

Prototype Testing and Validation

This chapters' objective is to show the tests performed during the development of the prototype. So, the next sections, depict the hardware and software tests carried in order to validate the content of the previous chapters.

7.1 Hardware Testing

As previously discussed, under System Description, ensuring the accuracy of the hardware is a crucial component of a well-functioning system.

Testing is needed to ensure that every component of a system is operating as it should, and that the system is performing exactly in accordance with the specific local requirements.

After hardware has been verified, tested and implemented, it must continue to be maintained.

Systems should be maintained to ensure that they continue to perform to the level demonstrated during the testing stage. Ongoing monitoring or testing plans may need to be put in place to ensure that maintenance needs are identified and met when necessary.

An important note is that, due to the global pandemic, many hardware components deliveries suffered a significant delay so the hardware could not be completely assembled and, consequently, could not be fully tested.

7.1.1 Cardiac Cell

The following figure represents the designed model schematics based on a cardiac cell developed by Y. Maeda, E. Yagi and H. Makino [12] and also tested by both my previous colleagues, Eng. Flávio Amorim [20] and Eng. Joana Salvador [39].

The software used was the NI Multisim[®] circuit design program.

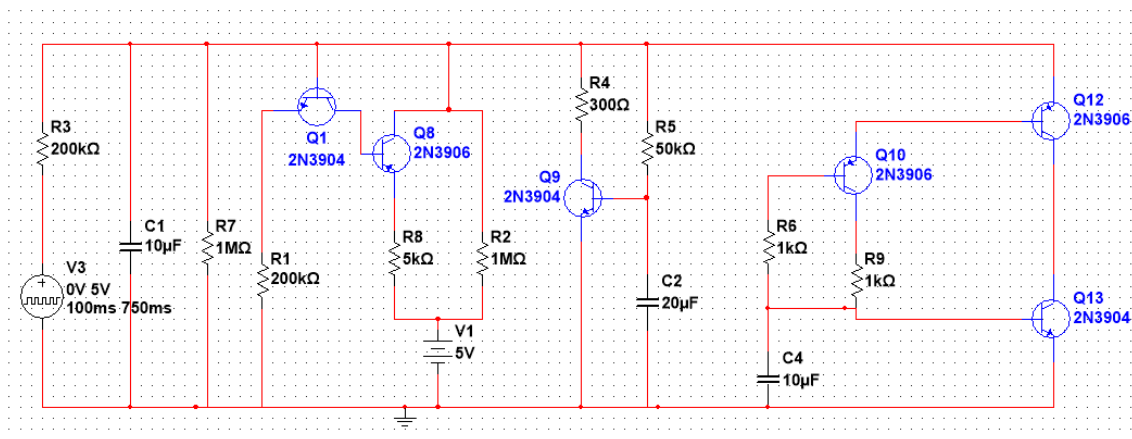


Figure 7.1: Cardiac Cell Model

This model uses a square wave in order to simulate the delivered stimulus to the cardiac cells.

Figure 7.2 represents the transient simulation of the action potential of cardiac cells presented by square waves with plateaus of different duration.

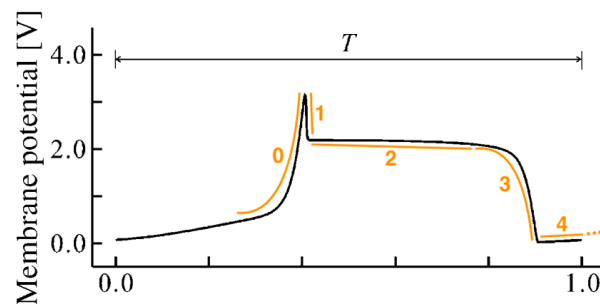


Figure 7.2: Transient Analysis of the Model. Extracted from [12]

In this figure, there are five phases represented:

- Rapid Depolarisation State (0)
- Rapid but Short Repolarisation (1)
- Plateau (2)
- Rapid Repolarisation (3)
- Slow Depolarisation (4)

Phase 0 represents the Rapid Depolarisation State. This up rise is caused by the opening of the sodium channels that allow Sodium influx of Sodium ions (Na^+) into the cytoplasm.

In phase 1, rapid polarisation occurs. This goes by the opening of Potassium channels which allows the efflux of positively charged Potassium ions (K^+) outside the cells. At the same time, sodium channels are suddenly closed making this phase the shortest one.

In phase 2, more or less plateau patterns are produced. The voltage-sensitive calcium channels are opened to facilitate the influx of positively charged Calcium ions (Ca^+) to balance the repolarising effects of potassium efflux.

In phase 3, there is rapid repolarisation of the cardiac cells. The calcium channels are suddenly closed which leaves the potassium efflux current unopposed.

Finally, in phase 4, sodium re-enters into the cardiac cells but now through slow spontaneous permeation through the cell membrane. This slow process prepares the cells for the next action potential cycle.

Following this, figure 7.3 represents the temporal waveforms of the hardware model for the cardiac cell obtained when simulating the circuit model in figure 7.1 indicating that it is correctly designed.

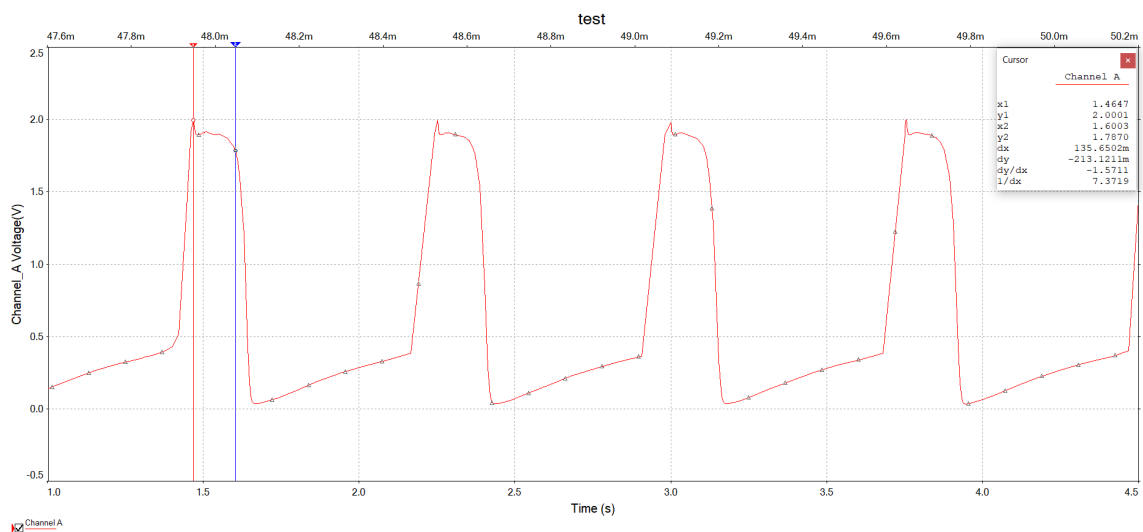


Figure 7.3: Transient Analysis of the Model: Obtained Result

7.1.2 Detection Phase

In order to acquire the signal received by the optical sensors, a first-order electronic low pass filter is added following an active high pass filter (combined passive RC filter network with an operational amplifier to produce a high pass filter with amplification).

The treated signal is then amplified using a comparator with an 1.5V offset and every time the heart pulses, the cardiac signal is detected in the optical sensor and the LED turns on.

An Analog-to-Digital converter could be used instead of the already integrated converter in the ESP32 microcontroller. The existing internal converter of the microcontroller does not have a precise resolution and the reason to add this new component (subsection 5.1.5.3) is due to its high performances and resolution as well as easy control. The signal can be measured by the ECG circuit and then sent to the microcontroller via SPI.

The complete circuit implementation is shown in figure 7.4 and the result obtained after filtering and amplifying the wave can be visualised in the following figure 7.5.

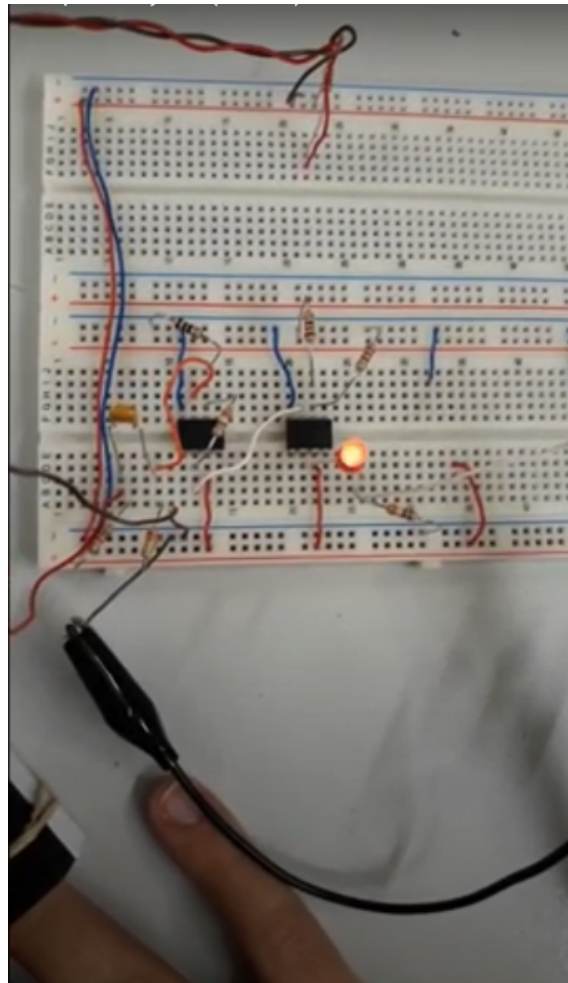


Figure 7.4: Implementation of ECG Scheme

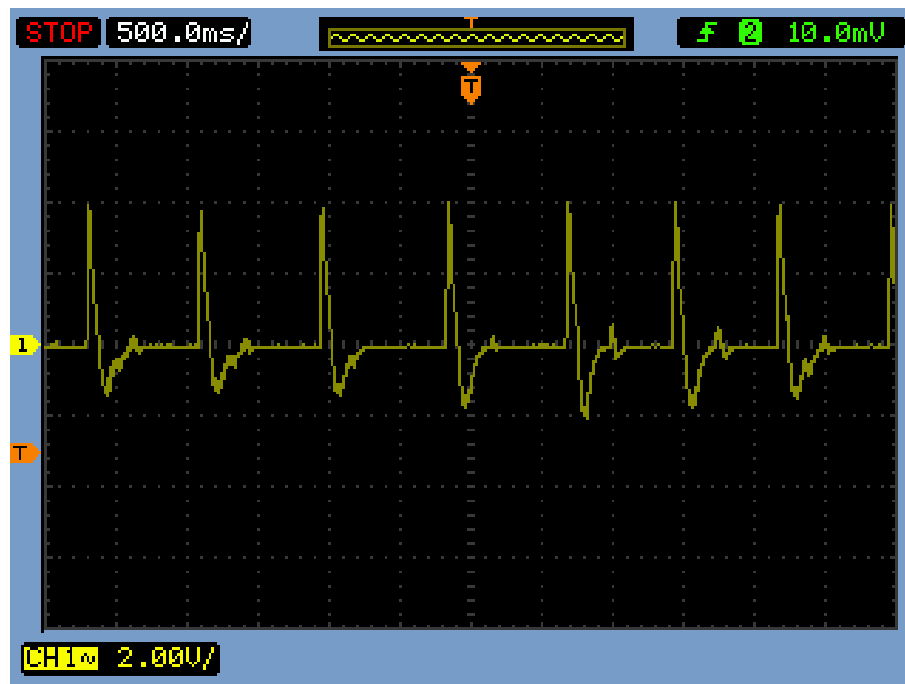


Figure 7.5: Filtered and Amplified Wave

As visualised, the LED turns ON when the cardiac signal is detected and the output wave obtained is in accordance with the section 2.2.2.1 referring to the electrocardiogram and so, in conclusion, the depicted circuit is correctly implemented.

7.2 Software Testing

In order to test the software part of the project, the scheme depicted in figure 6.3 is assembled, resulting in the following:

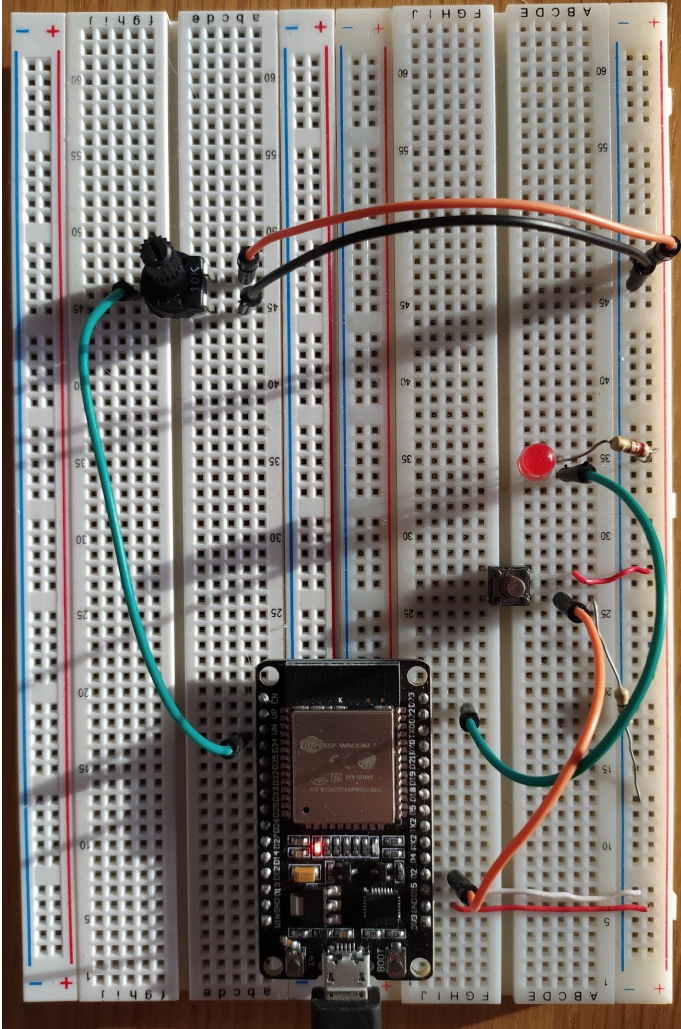


Figure 7.6: Bidirectional Communication: Testing Circuit Assembly

7.2.1 Client Side

In order to test the client side, the Arduino IDE is used. The code is uploaded and the serial monitor is opened.

The results obtained are further visualised in the following subsections.

7.2.1.1 Client Creation

After opening the serial monitor, a connection to the Wi-Fi is successfully established and client is created with assigned IP:

```
...  
...  
...  
WiFi connected with IP: 192.168.1.93
```

Figure 7.7: Client Successfully Created

7.2.1.2 Send data from Client to Server

- Assure connection to server (backend.py):

Since a connection to the server is non existent, a failed status occurs:

```
...  
...  
...  
WiFi connected with IP: 192.168.1.93  
Connection to backend.py failed
```

Figure 7.8: Connection to Server Failed

When connecting to the server (figure 7.17), a successful status occurs:

```
...  
...  
WiFi connected with IP: 192.168.1.93  
-----Connected to backend.py successfully!-----
```

Figure 7.9: Connection to Server Successful

- Basic test: Reading data from ESP32 Client

It is possible to perform a basic test on the connection using only a led and a button. If the button is pressed, the LED is turned on, else it is turned OFF by default.

```
...
...
WiFi connected with IP: 192.168.1.93
-----Connected to backend.py successfully!-----
LED OFF
```

Figure 7.10: Read data from client: Button not Ticked

```
WiFi connected with IP: 192.168.1.93
-----Connected to backend.py successfully!-----
LED ON
```

Figure 7.11: Read data from client: Button Ticked

- Send data to Server after connection established

After establishing a connection to the server, the potentiometer values can be visualised on serial monitor and on the server terminal. This indicates, the values are being successfully sent from the client (ESP32) to the server (backend.py).

```
----- Sending Data to backend.py -----
voltage Channel1: 2181
voltage Channel2: 1090
----- Disconnecting from backend.py... -----
```

Figure 7.12: Potentiometer Values on serial monitor

```
waiting for a connection
('connection from: ', ('192.168.1.93', 64936))
received: '{"voltage_channel1": 2181, "voltage_channel2":1090}'
('1589474713.66', '2181', '1090')
```

Figure 7.13: Potentiometer Values on backend terminal

7.2.1.3 Receive data from Server

When selecting values for both the Stimuli Channel, Stimuli Cycle and Atrial Stimulus Reading Interval, they are sent to the Django Web Server. Then, the server sends data to the client and is finally received on the client side.

The following figure, shows an example of data values selected. An important note is that, in a real situation of high-level control, the atria channel must be at all times selected in order to allow the correct stimulation of the cardiac tissue.

The screenshot shows a web form with the following sections and values:

- Stimuli Channel:** Radio buttons for Atria, Ventricle: 1, Ventricle: 2, and Ventricle: 3. Ventricle: 3 is selected.
- Stimuli Cycle:** Four input fields for delay after stimulation:
 - Auricle stimulation: 1.5 ms
 - Ventricle 1 stimulation: 0,0 ms
 - Ventricle 2 stimulation: 2.5 ms
 - Ventricle 3 stimulation: 0,0 ms
- Atrial stimulus Reading Interval:** Two input fields:
 - Min: 1 ms
 - Max: 2 ms
- Submit!** button at the bottom.

Figure 7.14: Send Values to Server

As observed, after ticking the submit button, the values are saved in a buffer and further visualised in the serial monitor (figure 7.15).

```
===== New connection from Django! =====
Received: {"Atria": null, "V1": null, "V2": null, "V3": "on", "Delay Atria": "1.5", "Delay V1": "0.0", "Delay V2": "2.5", "Delay V3": "0.0", "Min": "1", "Max": "2"}
===== Connection from Django closed =====
```

Figure 7.15: Receive data from Server

7.2.2 Server Side:

Similarly to the previous section (7.2.1), in order to test the server side, Pycharm (incorporated with the Django Framework) was used.

The code is executed and the (Pycharm) terminal is opened. The results obtained are further visualised in the following subsections

7.2.2.1 Server creation

After running "python manage.py runserver" command, the following message must be visualised.

The server is successfully created in the 127.0.0.1 domain (localhost) and is connected to port 8000. The designed web interface is tested in this server.

```
System check identified no issues (0 silenced).
May 20, 2020 - 14:17:04
Django version 3.0.3, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
Running
```

Figure 7.16: Server Creation: Pycharm Terminal

Secondly, a new terminal must be opened and the project folder must be browsed in here.

After this, "python backend.py" command must be called. In this phase, a connection is established to the server and is waiting for the client to connect.

```
Joanas-MacBook-Air:myproject joana$ python backend.py
starting up on 0.0.0.0 port 10000
waiting for a connection
```

Figure 7.17: Server Creation: Establishing connection to backend.py

7.2.2.2 Database creation

In order to create the Database, the command "python boards/DB/create_db.py" is called.

Figure 7.18 shows the project without the required database and figure 7.19 shows the created database ("tese.db") appearing on the project after running the said command.

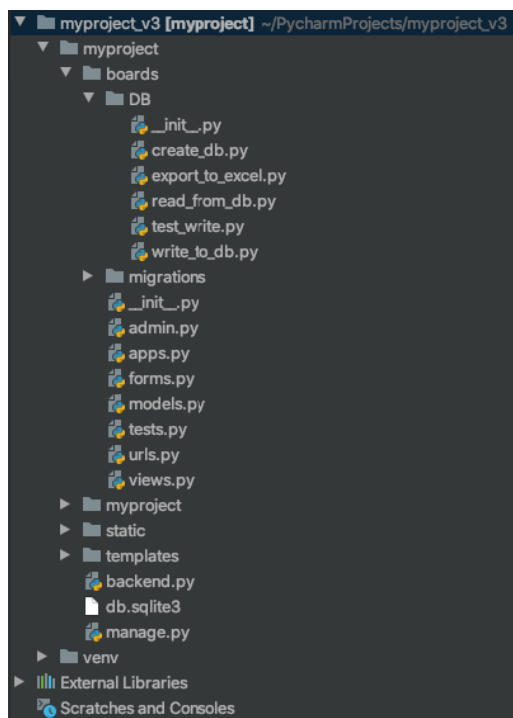


Figure 7.18: Project without database

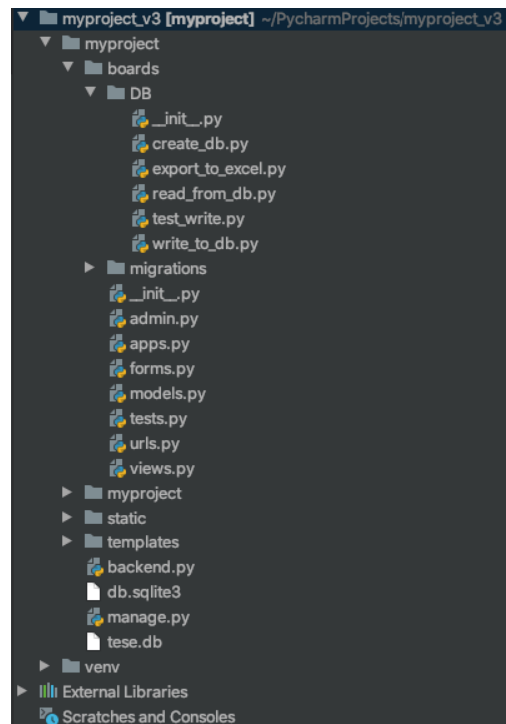


Figure 7.19: Database Created

7.2.2.3 Receive data from client and write to database

As already mentioned before, in this stage, SQL statements are used to insert data coming from the client in the previously created database.

This matter is not tested because it is already depicted in the following subsections as it is necessary to check the values in the database to compare with the set potentiometer values. In other words, this stage is successfully verified because it was possible to receive data and write it to the database (check figure 7.21 and 7.22).

7.2.3 Interface reads data from database

Similarly to the previous subsection, this section is also not directly tested because, in this stage, only data reading occurs and this can be visualised as an example in the terminal as already shown in figure 7.17 meaning that this phase is successfully completed.

7.2.4 Database export to Excel

In order to access data previously sent to the server from the microcontroller in an excel format, it is possible to do so by clicking in the button present in the Device Management Page.

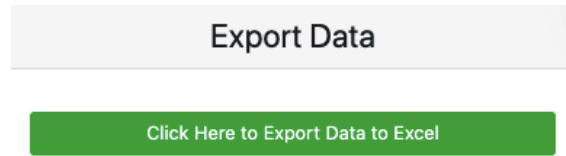


Figure 7.20: Export Data to an Excel file

An excel file appears on the project folder and the obtained values shown are as follows:

```
( '1591460095.21', 0.0, 0.0)
( '1591460096.43', 3005.0, 1502.0)
( '1591460097.67', 0.0, 0.0)
( '1591460098.69', 4005.0, 2002.0)
( '1591460100.01', 0.0, 0.0)
( '1591460101.26', 0.0, 0.0)
( '1591460102.47', 3863.0, 1931.0)
( '1591460103.7', 0.0, 0.0)
( '1591460104.93', 3903.0, 1951.0)
( '1591460106.26', 0.0, 0.0)
( '1591460107.49', 277.0, 138.0)
( '1591460108.72', 0.0, 0.0)
( '1591460109.74', 3334.0, 1667.0)
( '1591460110.77', 4095.0, 2047.0)
( '1591460112.0', 4095.0, 2047.0)
( '1591460113.23', 3099.0, 1549.0)
( '1591460114.45', 2410.0, 1205.0)
( '1591460115.68', 71.0, 35.0)
( '1591460116.91', 0.0, 0.0)
( '1591460118.14', 30.0, 15.0)
( '1591460119.37', 35.0, 17.0)
( '1591460120.4', 37.0, 18.0)
( '1591460121.67', 28.0, 14.0)
( '1591460122.94', 33.0, 16.0)
( '1591460124.11', 22.0, 11.0)
( '1591460125.1', 42.0, 21.0)
( '1591460126.13', 35.0, 17.0)
( '1591460127.36', 31.0, 15.0)
( '1591460128.58', 38.0, 19.0)
```

Figure 7.21: Values Sent to the Server

timestamp	voltage_channel1	voltage_channel2
1591460095.21	0	0
1591460096.43	3005	1502
1591460097.67	0	0
1591460098.69	4005	2002
1591460100.01	0	0
1591460101.26	0	0
1591460102.47	3863	1931
1591460103.7	0	0
1591460104.93	3903	1951
1591460106.26	0	0
1591460107.49	277	138
1591460108.72	0	0
1591460109.74	3334	1667
1591460110.77	4095	2047
1591460112.0	4095	2047
1591460113.23	3099	1549
1591460114.45	2410	1205
1591460115.68	71	35
1591460116.91	0	0
1591460118.14	30	15
1591460119.37	35	17
1591460120.4	37	18
1591460121.67	28	14
1591460122.94	33	16
1591460124.11	22	11
1591460125.1	42	21
1591460126.13	35	17
1591460127.36	31	15
1591460128.58	38	19

Figure 7.22: Values Saved in the Excel File

As shown, some values appear as null. This event occurs due to a faulty potentiometer which led to a misreading of some values.

Nevertheless, the values sent from the ESP32 to the server, and consequently exported from the interface to a local excel file, are correctly represented.

Conclusions and Future Work

Remote monitoring is likely to become a standard of care for patients with severe heart related diseases as they have the potential to improve the patient safety and support an efficient use of resources as well as cost reduction due to the fact that there's no need for surgery to check on the device state.

With pacemakers being safety-critical, bugs in the control program cannot be tolerated. Medical devices such as pacemakers are very prone to software errors due to the complex control algorithms that they use and so, an effort must be made to meet the quality and security of the device.

Pacemakers have a tendency to diminish in size and to better the life expectancy of the battery in the near future.

The development of a smaller printed circuit board (PCB), would allow the assembly of the developed circuit in a small device which is protected from the surrounding environment allowing a smaller incision when implantation occurs.

In order to avoid security breaches while accessing devices with wireless capabilities, there could be an addition of credentials to the projects' database (login, logout and forgot password functions).

Relatively to the graphical interface, the "Vitals" page was added but not completed. The intent of the page was to visualise received data in real-time through graphics.

Since the project is a smaller application directed to IoT, the database existent in the Django framework was sufficient although it could be replaced by PostgreSQL in order to increase its efficiency and to run over the network [40].

Another functionality that could be added would be the storage of data present in the database in an SD card using SPI. This could be proven useful when saving data in a external storage is needed.

Finally, the sensing scheme can be upgraded to a more efficient circuit. The actual circuit is completed with clamping diodes and serial switches, not shown, required to isolate the circuit from the stimulation pulses. These functionalities could be implemented in the future.

Some of the difficulties encountered were mainly in the hardware part such as the understanding of the circuit and delays in components deliveries.

It is important to note that delays occurred with the material deliveries due to the fact that the current situation with the world's health, closed all universities including the medicine university (FMUP) in which this work was developed.

So, the circuit is not complete nor properly tested.

Relatively to the software part, and because it depends on the parameters that the microcontroller receives from the hardware, the values depicted are not representing real data coming from the cardiac tissue even though the bidirectional communication via wireless is functioning correctly.

Appendix A

Software Development Modules

A.1 Client Side

A.1.1 Client Creation

```
#include <WiFi.h>
#define PB_PIN 15
#define LEDPIN 22
#define POTPIN 34

// Network credentials
const char* ssid = " ";
const char* password = " ";

const uint16_t port = 10000;
const char * host = "192.168.1.2";

WiFiServer wifiServer(port);

int potValue = 0;

void setup() {
  Serial.begin(115200);
  pinMode(LEDPIN, OUTPUT);
  pinMode(PB_PIN, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("...");
  }

  Serial.print("WiFi connected with IP: ");
  Serial.println(WiFi.localIP());

  wifiServer.begin();
}
```

A.1.2 Send Data from Server to ESP32

```
void send_data()
{
  WiFiClient client;
  int Push_button_state = digitalRead(PB_PIN);
  int potValue = analogRead(POTPIN);

  if (!client.connect(host, port)) {
    Serial.println("Connection to backend.py failed");
    delay(1000);
    return;
  }

  Serial.println("-----Connected to backend.py successfully!-----");

  //Read data from ESP32
  if (Push_button_state == HIGH ){
    digitalWrite(LEDPIN, HIGH);
    Serial.print("LED ON \n");
  }
  else {
    digitalWrite(LEDPIN, LOW);
    Serial.print("LED OFF \n");
  }

  //Send data to backend.py
  Serial.println("----- Sending Data to backend.py -----");
  Serial.print("voltage Channel1: ");
  Serial.println(potValue);

  Serial.print("voltage Channel2: ");
  Serial.println(potValue/2);

  client.print("{\"voltage_channel1\": ");
  client.print(potValue);
  client.print(", \"voltage_channel2\":");
  client.print(potValue/2);
  client.print("}");

  Serial.println("----- Disconnecting from backend.py... -----");
  client.stop();
}
```

A.1.3 Receive Data from Server

```
void receive_data()
{
    WiFiClient client = wifiServer.available();

    char buffer[800];
    int index = 0;

    if (client) {
        Serial.println("===== New connection from Django!! =====");

        while (client.connected())
        {
            while (client.available() > 0) {
                // Receive from backend.py
                char c = client.read();
                if(index < 799)
                {
                    buffer[index] = c;
                    index++;
                    buffer[index] = '\0';
                }

            }

            delay(10);
        }

        Serial.print("Received: ");
        Serial.println(buffer);
        client.stop();
        Serial.println("===== Connection from Django closed =====");
    }
}
```


A.2 Server Side

A.2.1 Server Creation

```
import socket
import sys
import json
import time

from boards.DB.write_to_db import write

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('0.0.0.0', 10000)
print('starting up on %s port %s' % server_address)
sock.bind(server_address)

# Listen for 1 incoming connections
sock.listen(1)

while True:
    # Wait for a connection
    print('waiting for a connection')
    connection, client_address = sock.accept()

    try:
        print('connection from: ', client_address)
        data = ""
        # Receive the data in small chunks
        while True:
            received = connection.recv(16)
            if not received: break
            data += str(received.decode('utf-8'))

        print('received: "%s"' % data)

        try:
            data_json_str = json.dumps(str(data))
            value_received = json.loads((json.loads(data_json_str)))
            voltage_channel1 = str(value_received['voltage_channel1'])
            voltage_channel2 = str(value_received['voltage_channel2'])
        except Exception as e:
            print(e)
            print("Error in data format. Check client data format")

        timestamp=str(time.time()) # Timestamp

        print(timestamp, voltage_channel1, voltage_channel2)
        if timestamp and voltage_channel1 and voltage_channel2:
            write('pacemaker', [timestamp, voltage_channel1, voltage_channel2])

    except:
        # If error exit
        pass

# Clean up the connection
connection.close()
```

A.2.2 Database Creation

```
import sqlite3

conn = sqlite3.connect('tese.db')

c = conn.cursor()

# Create table
c.execute('''CREATE TABLE pacemaker
            (timestamp text, voltage_channel1 real, voltage_channel2 real)''')

# Save (commit) changes
conn.commit()

# Close the connection
conn.close()
```

A.2.3 Receive data from client and write to Database

```
import sqlite3

# Database Schema
DB_NAME = 'tese.db'

def write(table, value=[]):
    conn = sqlite3.connect(DB_NAME)
    c = conn.cursor()

    value_str = ', '.join(value)
    query_str = "INSERT INTO " + table + " VALUES (" + value_str + ")"
    print("Running: " + query_str)

    c.execute(query_str)
    conn.commit()
    conn.close()
```

A.3 Interface reads Data from Database

```
import sqlite3

# Database Schema
DB_NAME = 'tese.db'

def read_all():
    conn = sqlite3.connect(DB_NAME)
    c = conn.cursor()

    timestamp_list = []
    voltage1_list = []
    voltage2_list = []

    query_str = "SELECT * FROM pacemaker ORDER BY timestamp"
    for row in c.execute(query_str):
        print(row)
        timestamp_list.append(row[0])
        voltage1_list.append(row[1])
        voltage2_list.append(row[2])

    return [timestamp_list, voltage1_list, voltage2_list]
```


Appendix B

Graphical Interface

B.1 Device Management Page: Data Input

B.1.1 Management Function (Views.py)

```
def management(request):
    print("Running")
    print(request.POST)

    if request.method == "POST":
        print("test POST")
        my_form = MyForm(request.POST)
        if my_form.is_valid():
            data = request.POST.copy()
            print('----- Stimuli Channel -----')
            print('Atria:', data.get('checkbox'))
            print('V1:', data.get('checkbox2'))
            print('V2:', data.get('checkbox3'))
            print('V3:', data.get('checkbox4'))
            print('----- Stimuli Cycle -----')
            print("Delay Atria:", data.get('sti_value1'))
            print("Delay V1:", data.get('sti_value2'))
            print("Delay V2:", data.get('sti_value3'))
            print("Delay V3", data.get('sti_value4'))
            print('--- Stimuli Reading Interval ----')
            print("Min:", data.get('min'))
            print("Max", data.get('max'))

            data_esp = {
                'Atria': data.get('checkbox1'),
                'V1': data.get('checkbox2'),
                'V2': data.get('checkbox3'),
                'V3': data.get('checkbox4'),
                'Delay Atria': data.get('sti_value1'),
                'Delay V1': data.get('sti_value2'),
                'Delay V2': data.get('sti_value3'),
                'Delay V3': data.get('sti_value4'),
                'Min': data.get('min'),
                'Max': data.get('max')
            }
            send2esp(data_esp)

        else:
            my_form = MyForm()

    return render(request, "management.html", {'form': my_form})
```

B.1.2 Send to ESP32 Function (Views.py)

```
def send2esp(data_esp):
    sock = socket.socket()
    sock.connect(("192.168.1.6", 10000))
    sock.send(json.dumps(data_esp).encode('utf-8'))
    sock.close()
```

B.1.3 Forms.py

```

from django import forms

class MyForm(forms.Form):
    checkbox = forms.BooleanField(required=False)
    checkbox2 = forms.BooleanField(required=False)
    checkbox3 = forms.BooleanField(required=False)
    checkbox4 = forms.BooleanField(required=False)

    sti_value1 = forms.FloatField(required=False, initial=0.0)
    sti_value2 = forms.FloatField(required=False, initial=0.0)
    sti_value3 = forms.FloatField(required=False, initial=0.0)
    sti_value4 = forms.FloatField(required=False, initial=0.0)

    min = forms.FloatField(required=False, initial=0.0)
    max = forms.FloatField(required=False, initial=0.0)

```

B.1.4 Management.html: Input Data Form

```

<form action="" method="POST">
  <div class="card shadow-lg border-0 rounded-lg mt-0">
    <div class="card-header">
      <h3 class="text-center font-weight-light my-0">Stimuli Channel</h3></div>
    <div class="card-body">
      &nbsp; &nbsp; &nbsp; &nbsp;
      {{ form.checkbox }}
      <label> Atria </label>

      {{ form.checkbox2 }}
      <label>Ventricle: 1 </label>

      {{ form.checkbox3 }}
      <label>Ventricle: 2 </label>

      {{ form.checkbox4 }}
      <label>Ventricle: 3 </label>
    </div>
    <div class="card-header">
      <h3 class="text-center font-weight-light my-0">Stimuli Cycle </h3>
    </div>
    <div class="card-body " >
      <p>Delay after:</p>
      <p><label > - Auricle stimulation </label> {{ form.sti_value1 }}<label> ms</label></p>
      <p><label> - Ventricle 1 stimulation </label> {{ form.sti_value2 }}<label> ms </label></p>
      <p><label> - Ventricle 2 stimulation </label> {{ form.sti_value3 }}<label> ms</label></p>
      <p><label> - Ventricle 3 stimulation </label> {{ form.sti_value4 }}<label> ms</label></p>
    </div>
    <div class="card-header">
      <h3 class="text-center font-weight-light my-0">Atrial stimulus Reading Interval</h3>
    </div>
    <div class="card-body">
      <center><p><label> Min &nbsp;&nbsp;&nbsp; </label> {{ form.min }}<label> &nbsp;&nbsp;&nbsp; ms </label></p>
      <p><label> Max &nbsp;&nbsp;&nbsp; </label> {{ form.max }}<label> &nbsp;&nbsp;&nbsp; ms </label></p></center>
    </div>
    {% csrf_token %}
    <input type="submit" value="Submit!"/>
  </div>
</form>

```

B.2 Device Management Page: Data Output

B.2.1 Data Export to Excel

```
from pandas import DataFrame

from .read_from_db import read_all

def export(filename=""):
    # If no filename provided don't run this function
    if not filename: return

    print("hello")
    db_records = read_all()
    df = DataFrame({'timestamp': db_records[0], 'voltage_channel1':
                    db_records[1], 'voltage_channel2': db_records[2]})
    df.to_excel(filename + '.xlsx', sheet_name='sheet1', index=False)
```

B.2.2 Download Excel (Views.py)

```
def downloadexcel(request):
    export('teste')
    return render(request, 'management.html')
```

B.2.3 Management.html: Output Data Button

```
<div class="card-header">
  <h3 class="text-center font-weight-light my-0"> Export Data</h3>
</div>
<div class="card-body">
  <button class="mx-auto col-md-12 btn btn-success mt-3" type="button" onclick="location.href={% url 'downloadexcel'%}" >
    Click Here to Export Data to Excel</button>
</div>
```


References

- [1] Giuseppe Lippi and Mario Plebani. *Biomarker research and leading causes of death worldwide: a rather feeble relationship*. DE GRUYTER, 2013.
- [2] Rafael Dal-Ré. *Worldwide Clinical Interventional Studies on Leading Causes of Death: A Descriptive Analysis*. ELSEVIER, 2011.
- [3] Blood flow: Blood supply of the heart. <https://sites.google.com/site/theheartanatomyproject2012/blood-flow?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>.
- [4] Queensland Cardiovascular Group. Anatomy of the heart. <https://qcg.com.au/patients/anatomy-heart>.
- [5] Central Georgia Heart Center. What you need to know about an implantable cardioverter defibrillator. Pearson Education, Inc. Available in <https://www.thinglink.com/scene/794660186202570753>, 2011.
- [6] Thinglink. Electrophysiology of the heart. Available in <https://centralgaheart.com/implanting-a-defibrillator/>, 2017.
- [7] James R. Edgerton Mani Arsalan, J. Michael DiMaio. Pacemaker and icd insertions. Thoracic Key. Available in <https://thoracickey.com/pacemaker-and-icd-insertions/>.
- [8] Joshua M. Cooper Haris M. Haqqani, Laurence M. Epstein. Engineering and construction of pacemaker and icd leads. Thoracic Key. Available in <https://thoracickey.com/engineering-and-construction-of-pacemaker-and-icd-leads/>.
- [9] Pacemaker system. Mayo Clinic: Cardiology Heart Surgery Hospital. Available in <https://www.mayoclinic.org/tests-procedures/pacemaker/about/pac-20384689>.
- [10] Who needs an implantable cardioverter-defibrillator? Harvard Health Publishing: Harvard Medical School. Available in <https://www.health.harvard.edu/drugs-and-medications/who-needs-an-implantable-cardioverter-defibrillator>, 2011.
- [11] Pacemakers: Leadless pacemaker. Cleveland Clinic. Available in <https://my.clevelandclinic.org/health/treatments/17166-pacemakers-leadless-pacemaker>.
- [12] Hideo Makino Yoshinobu Maeda, Eiji Yagi. *Synchronization with low power consumption of hardware models of cardiac cells*. ELSEVIER, 2005.

- [13] Sophie White. The conducting system of the heart. <https://teachmeanatomy.info/thorax/organs/heart/conducting-system/>.
- [14] Bernardo Manuel de Sousa Pinto. Morfologia do sistema cardiovascular e do sangue. Faculdade de Medicina da Universidade do Porto.
- [15] Johns Hopkins Medicine. Pacemaker insertion. <https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/pacemaker-insertion/>.
- [16] Niebauer J. Ashley EA. Cardiology explained, 2004. Chapter 3, Conquering the ECG.
- [17] Neal Bhatia and Mikhael El-Chami. Leadless pacemakers: a contemporary review, April 2018. Journal of Geriatric Cardiology available in <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5997619/>.
- [18] Neuzil P Sogaard P. Butter C. Seifert M. Delnoy PP van Erven L Schalji M Boersma LVA Riahi S Reddy VY, Miller MA. Cardiac resynchronization therapy with wireless left ventricular endocardial pacing: The select-lv study. Journal of Geriatric Cardiology available in <https://www.ncbi.nlm.nih.gov/pubmed/28449772>, MONTH=May, YEAR=2017.
- [19] Rice University. Battery-less pacemaker. Journal of Geriatric Cardiology available in <https://healthmanagement.org/c/cardio/news/battery-less-pacemaker>, MONTH=Jun, YEAR=2017.
- [20] Flávio R. Amorim. *Pacemaker Multielétrodo*. FEUP, 2016.
- [21] Components101. Esp32 - devkitc. Available in <https://components101.com/microcontrollers/esp32-devkitc>.
- [22] Texas Instruments. *OPAx350 High-Speed, Single-Supply, Rail-to-Rail Operational Amplifiers MicroAmplifier Series*. 2020.
- [23] Texas Instruments. *DACx311 2-V to 5.5-V, 80-A, 8-, 10-, and 12-Bit, Low-Power, Single-Channel, Digital-to-Analog Converters in SC70 Package*. 2020.
- [24] Texas Instruments. *Octal, 16-Bit, Low-Power, High-Voltage Output, Serial Input DIGITAL-TO-ANALOG CONVERTER*. 2020.
- [25] Maxim. *Quad, SPST, CMOS Analog Switches*. 2020.
- [26] Linear Technology. *600mA, 1.2MHz Micropower Synchronous Boost Converter in ThinSOT*. 2020.
- [27] Linear Technology. *1.2MHz Step-Up DC/DC Converter in SOT-23*. 2020.
- [28] Siemens. *GaAIAs-IR-Lumineszenzdiode (880nm), GaAIAs Infrared Emitter (880 nm)*. 2020.
- [29] Siemens. *NPN-Silizium-Fototransistor Silicon NPN Phototransistor*. 2020.
- [30] Texas Instruments. *ADS868x 16-Bit, High-Speed, Single-Supply, SAR ADC Data Acquisition System With Programmable, Bipolar Input Ranges*. 2020.
- [31] Denis Flandre Fernando Silveira. *Low Power Analog CMOS for Cardiac Pacemakers: Design and Optimization in Bulk and SOI Technologies*. SPRINGER SCIENCE+BUSINESS MEDIA, LLC, 2004.

- [32] Margaret Rouse. Tcp/ip (transmission control protocol/internet protocol). Available in <https://searchnetworking.techtarget.com/definition/TCP-IP?amp=1>.
- [33] Tcp communication. Available in <https://wiki.python.org/moin/TcpCommunication>.
- [34] Gareth Dwyer. Flask vs. django. Available in <https://www.codementor.io/@garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v>.
- [35] Merixstudio. Flask vs django: choosing python framework. Available in <https://www.merixstudio.com/blog/flask-vs-django-choosing-python-framework/>.
- [36] Nathan Jennings. Flask vs django: How to understand whether you need a hammer or a toolbox. Available in <https://steelkiwi.com/blog/flask-vs-django/>.
- [37] Nathan Jennings. Socket programming in python (guide). Available in <https://realpython.com/python-sockets/>.
- [38] Django documentation:working with forms. Available in <https://docs.djangoproject.com/en/3.0/ref/request-response/#django.http.HttpRequest>.
- [39] Joana P. Salvador. *Pacemaker Multieléctrodos*. FEUP, 2018.
- [40] TablePlus. Sqlite vs postgresql. Available in <https://tableplus.com/blog/2018/08/sqlite-vs-postgresql-which-database-to-use-and-why.html>.