Universidad Rey Juan Carlos

ESCUELA DE
INGENIERÍA DE FUENLABRADA

DEGREE IN BIOMEDICAL ENGINEERING

End of Degree Project

ELECTROCARDIOGRAPH BASED ON ARDUINO MICRO CONTROLLER: DEVICE AND SOFTWARE DESIGN AND IMPLEMENTATION

Author: Ferran García Torres

Supervisor: Antonio Consoli Barone

Academic Course 2023/2024

The first step in the diagnosis of disease consists in careful and accurate investigation of the symptoms.

René Laennec

Acknowledgments

Firstly, I want to thank my tutor, Antonio Consoli. The help, advice and knowledge that he provided were crucial to carry out this project. I am very grateful to have worked with you, first as a teacher and later as a tutor.

I want to thank my parents Manel and María Jose for providing me with an excellent education, encouraging me to pursue my dreams and giving me priceless advice through life while giving me total freedom to do what I consider the best. Also, thank my brother Guillem, it seemed like it was yesterday when we were just kids, time flies and I am so grateful that it is by your side. I listen to you and admire you as a little brother admires his older brother. I also want to dedicate some words to my late grandparents, I am deeply saddened that you cannot see me graduate, but I know that, wherever you are, you are taking care of me and are very proud of the man that I have become.

I also would like to thank my friend Álvaro, with whom I have been a flatmate for almost the whole degree. We did not know each other until both of us decided to study biomedical engineering. Thank you for your advice in the hard moments, and, above all, thank you for all the funny moments we spent together. Also thank Luis and Dani, the friends I went on Erasmus with. They have been key parts of my life these years. It is a shame that I cannot mention every friend I met in the degree, but I also want to thank all of them for making me want to go to the university every day.

I also want to mention a very important person, with whom I have been since the very beginning of this adventure. We have supported each other through all the ups and downs of all these years. Thank you for listening to me, giving me advice, and believing in me.

Finally, I want to extend my sincere thanks to all the teachers and all the employees of the Rey Juan Carlos University. Thank you for all the knowledge that you transmitted to me.

Abstract

This project focuses on the design and development of an electrocardiograph (ECG) utilizing a heart rate sensor and an Arduino microcontroller, this is accompanied by an intuitive and interactive Graphical User Interface (GUI). It fulfils the need for reliable and accessible alternatives for cardiac monitoring in domestic and medical contexts. The ECG is an essential diagnostic tool that provides valuable information regarding the electrical activity of the heart, allowing healthcare professionals to identify cardiac anomalies effectively. However, it is not only used for cardiac disease prognoses, like palpitations or chest pain, it is also used to evaluate symptoms like shortness of breath and dizziness, that might indicate underlying conditions like arrhythmia.

The main goal of this project is to create a portable and inexpensive ECG device for domestic use as a personal health monitoring tool and for educational purposes in biomedical degrees. The ECG system, based on the AD8232 sensor, includes a GUI designed to provide an intuitive and effective cardiac monitoring experience to users. By offering a versatile and cost-effective alternative, this project aims to enlarge access to cardiac monitoring, contributing to the cardiovascular health of the general population and also serving as an educational tool for biomedical engineering degree students.

The development of the ECG monitoring system encompasses multiple key aspects. The components utilized were chosen based on their compatibility and performance with the Arduino board. The hardware development includes the design of a Printed Circuit Board (PCB) and a 3D-printed enclosure to store the device securely. The software components include the development of Python code to create the connection with the AD8232 sensor in order to process the captured data and the GUI which displays the ECG data in real-time. Filters are also applied to the ECG to remove the noise from the registered data.

Multiple testing and validation of the ECG system were performed to ensure its efficacy and reliability. The tests include measuring the ECG of a healthy person under different circumstances to evaluate the device performance under multiple conditions. Furthermore, the device's resistance to external interferences is also covered in order to ensure a reliable performance in different environments.

Keywords: Electrocardiograph, Arduino, heart rate monitor, graphical user interface, cardiac monitoring, biomedical engineering, portable device, health monitoring, educational tool, cardiac anomalies.

Resumen

Este proyecto se centra en el diseño y desarrollo de un electrocardiógrafo (ECG) que utiliza un sensor de medición de frecuencia cardiaca y un microcontrolador de Arduino, acompañado de una interfaz gráfica de usuario (GUI) intuitiva e interactiva. Satisface la necesidad de alternativas fiables y accesibles para la monitorización cardiaca en contextos domésticos y médicos. El electrocardiograma es una herramienta de diagnóstico muy útil que proporciona valiosa información sobre la actividad eléctrica del corazón, lo que permite a los profesionales sanitarios identificar eficazmente las anomalías cardiacas. Sin embargo, no sólo se utiliza cuando los síntomas están estrechamente relacionados con enfermedades cardiacas, como palpitaciones o dolor torácico, sino también para evaluar síntomas como la falta de aire y los mareos, que podrían indicar afecciones subyacentes como arritmias.

El objetivo principal de este proyecto es crear un dispositivo de ECG portátil y barato para uso doméstico y con fines educativos en carreras biomédicas. El dispositivo, basado en el sensor AD8232, incluye una interfaz gráfica de usuario diseñada para proporcionar una experiencia de monitorización cardiaca intuitiva y eficaz a los usuarios. Al ofrecer una alternativa versátil y rentable, este proyecto pretende ampliar el acceso a la monitorización cardiaca para la población general, contribuyendo a la salud cardiovascular de la población y sirviendo también como herramienta educativa para estudiantes de titulaciones relacionadas con la salud.

El desarrollo del sistema de monitorización de ECG abarca múltiples aspectos clave. Los componentes utilizados se eligieron en función de su compatibilidad y rendimiento con la placa Arduino. El desarrollo del hardware incluye el diseño de una placa de circuito impreso (PCB) y una carcasa impresa en 3D para almacenar el dispositivo. Los componentes de software incluyen el desarrollo de código Python para crear la conexión con el sensor AD8232 para procesar los datos capturados y la interfaz gráfica de usuario (GUI) que muestra los datos de ECG en tiempo real. También se aplican filtros al ECG para eliminar el ruido de los datos.

Se realizaron múltiples pruebas y validaciones del sistema de ECG para garantizar su eficacia y fiabilidad. Las pruebas incluyen la medición del ECG de una persona sana en distintas circunstancias para evaluar el rendimiento del dispositivo en diferentes condiciones. Además, también se analiza la resistencia del dispositivo a las interferencias externas para garantizar un funcionamiento fiable en distintos entornos.

Palabras clave: Electrocardiógrafo, Arduino, monitor de frecuencia cardiaca, interfaz gráfica de usuario, monitorización cardiaca, ingeniería biomédica, dispositivo portátil, monitorización de la salud, herramienta educativa, anomalías cardiacas.

Contents

Acknowledgements

Abstract

Resumen

List of Figures **v**

List of abbreviations **vi**

1 Introduction **1**

1.1 Context and motivation 1

1.2 Objectives 2

2 State of art **3**

2.1 History of Electrocardiography 3

2.2 ECG concept and interpretation 4

2.2.1 ECG wave structure 4

2.2.2 Cardiac Cycle 5

2.3 Diseases detected with an ECG 7

2.3.1 Myocardial infarction 7

2.3.2 Abnormal heart rhythms 8

2.3.3 Enlargement of the heart 10
2.4 Sensors for Cardiac Monitoring 11
2.4.1 ADS1292R 11
2.4.2 MAX30003 12
2.5 Regulatory Considerations 13
2.6 Modern and future advancements in portable ECG monitoring devices 15

3 Materials and Methodology 17
3.1 Components Selection 17
3.1.1 Arduino Mega 2560 17
3.1.2 AD8232 Sensor 19
3.2 Code 21
3.2.1 Arduino Code 21
3.2.2 Python Libraries 22
3.2.3 Filters applied to the signal 24
3.2.4 Software for PCB design: Fritzing 26
3.2.5 Software for 3D Design: TinkerCAD 26

4 Design and Development 29
4.1 Hardware Design 29
4.1.1 PCB 29
4.1.2 3D printed enclosure 32
4.2 GUI Design 37
4.2.1 Python Code 39

5 Experiments and Results 45
5.1 Device Performance 46
5.1.1 PCB test 46
5.1.2 External interferences 47

5.2 Measurements 50

5.2.1 Rest test 50

5.2.2 Stress test 52

6 Conclusions **55**

6.1 Summary of achievements 56

6.2 Future Scope 56

Bibliography **59**

Additional Material **65**

List of Figures

2.1 Evolution of ECG through history [4]. 3
2.2 ECG trace of a healthy patient [51]. 5
2.3 ECG wave with the phases of the Cardiac Cycle [52]. 6
2.4 ECG with Myocardial Infarction [53]. 8
2.5 ECG with an Arrhythmia [54]. 9
2.6 ECG with left Ventricular Hypertrophy. 10

3.1 Schematic of the Arduino Mega 2560 [49] 19
3.2 AD8232 sensor [50] 20
3.3 Arduino IDE code 22
3.4 Libraries imported in Python 24

4.1 Schematic view of the connection between the AD8232 and the Arduino 31
4.2 PCB view of the connection between the AD8232 and the Arduino 32
4.3 Isometric view of the 3D design of the support. 33
4.4 Frontal view of the 3D design of the support. 34
4.5 Top view of the 3D design of the support. 34
4.6 Lateral view of the 3D design of the support. 34
4.7 Isometric view of the 3D design of the box. 35
4.8 Frontal view of the 3D design of the box. 35
4.9 Top view of the 3D design of the box. 36

4.10 Lateral view of the 3D design of the box. 36
4.11 Window of the GUI that shows how to place electrodes. 38
4.12 Window of the GUI in which the user selects the COM port. 39
4.13 Initial view of the GUI. 39

5.1 Updated design of the PCB. 47
5.2 ECG trace recorded when the PC was connected to the charger. 49
5.3 ECG of a healthy individual in the designed GUI. 51
5.4 ECG of a healthy individual after physical exercise. 53

List of abbreviations

ABS	Acrylonitrile Butadiene Styrene
AC	Alternating Current
AFE	Analog Front-End
BPM	Beats per Minute
CAD	Computer Aided Design
CSV	Comma Separated Values
DC	Direct Current
ECG	Electrocardiogram
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
GUI	Graphical User Interface
IDE	Integrated Development Environment
LED	Light Emitting Diode
LO	Lead-Off
MDD	Medical Device Directive
MDR	Medical Device Regulation
PCB	Printed Circuit Board

PIL	Python Imaging Library
PLA	Polylactic Acid
PSU	Power Supply Unit
PSU	Graphical User Interface
PWM	Pulse Width Modulation
RLD	Right-Leg Drive
SPI	Serial Peripheral Interface
SPS	Samples per Second
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

Chapter 1

Introduction

1.1 Context and motivation

In the intersection between technology and health, the progress in medical monitorization through history has demonstrated to be crucial in order to improve the diagnosis and treatment of different illnesses. Nowadays, cardiovascular diseases are the leading cause of death worldwide, an effective measure in order to tackle this alarming problem is to monitor the population's vitals which has been demonstrated to be a powerful action to prevent different cardiovascular illnesses or, at least, moderate the consequences in severe diseases.

This project is focused on the development of an electrocardiograph using a micro controller and a heart rate monitor, this is supported with an interactive and intuitive GUI. This project arises in response to the need of accessible and efficient solutions for cardiac monitorization in both medical and domestic environments.

The electrocardiogram (ECG) is an accessible diagnosis tool which provides information about the electrical activity of the heart, allowing the healthcare professionals to detect cardiac anomalies. But it is not only used in prognoses that are related to heart diseases like chest pain or palpitations, it is also used when a patient presents symptoms such as dizziness or shortness of breath, which could also lead to illnesses like arrhythmia. One of the goals of this project is to let the people have available in their homes a tool that can help them to get an initial diagnostic, obviously with the posterior assistance of a healthcare professional if it is required. However, the conventional equipment is very expensive, in the case of the usual 12 lead ECG machine the usual price goes from 1500 dollars to 4000 dollars, and it requires specific training for its correct use. On this basis, the usage of Arduino-based devices offers cheaper and more available solutions, reaching a wider and less specific audience.

1.2 Objectives

The main goal of this project is to develop a portable and affordable ECG device for use at home as a simple tool for personal health monitoring, and for academic purposes as an educational tool used in biomedical engineering courses. This device will consist of the Arduino AD8232 sensor, it will also feature a GUI designed in order to provide an efficient and intuitive cardiac monitoring tool. The objective is to expand the access to cardiac examination, offering a cheap and versatile solution which contributes to the cardiovascular health of the general population. This ECG monitoring system will be used by students from multiple degrees of the medical field at the Rey Juan Carlos University, allowing them to comprehend and gain experience with biomedical devices and their applications.

To achieve this, the following secondary goals have been established:

- **PCB Design** This objective implies the detailed design and manufacture of the electronic circuit using the Eagle PCB software. A complete circuit diagram will be developed knowing the technical specifications of the AD8232 sensor, as well as the other needed components for the desired functioning of the device.

- **3D printed enclosure** In addition to the above-mentioned objective, the other secondary objective regarding hardware, is the design of a 3D printed box which will group the hardware parts of the device. A CAD software will be used to design this piece. The goal is to have all the hardware components grouped in order to protect them of external harmful factors.

- **Graphical User Interface development.** The aim is to create an intuitive and easy-to-use Graphic User Interface. In this window, the ECG will be represented in real time and the user will be capable of recording the signal and save it on the local computer.

- **Development of a Stand-Alone executable (.exe) file in Windows to Access the ECG** Regarding the software, along with the GUI, the goal is to create a desktop application in .exe format. It will allow the user to access and use the ECG easily without the need of previous technical knowledge. It will be compatible with usual operative systems.

Chapter 2

State of art

2.1 History of Electrocardiography

The origins of the ECG can be traced back to the 1780s, in particular to 1786, that year, Luigi Galvani noticed that it was possible to capture electrical current from skeletal muscles [4]. Eventhough the heart is not considered a skeletal muscle, it served as a basis for Carlo Mateucci, who, in 1842, proved that each heartbeat of a frog is accompanied by an electrical impulse [4]. Nevertheless, it was not until 1887 that August Waller published the first prototype of an ECG in human beings. To accomplish this, he used a Lipmann capillary electrometer placing the electrodes in the chest and the back of the body [5]. Along with this publication, he also demonstrated that the electrical activity had its origin in the ventricular contraction [6]. All of this served as inspiration for Willem Einthoven who created the first clinical ECG.

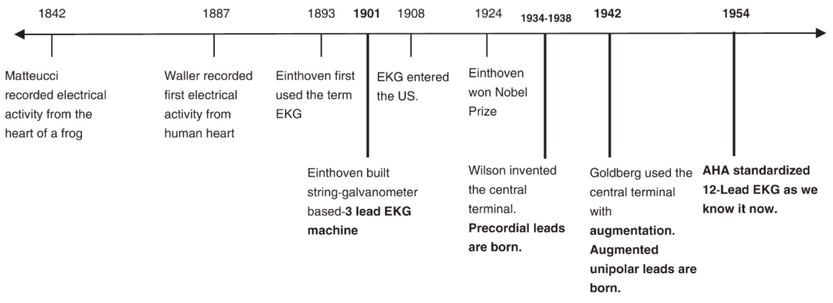

1842
Matteucci recorded electrical activity from the heart of a frog
1887
Waller recorded first electrical activity from human heart
1893
Einthoven first used the term EKG
1901
Einthoven built string-galvanometer based-3 lead EKG machine
1908
EKG entered the US.
1924
Einthoven won Nobel Prize
1934-1938
Wilson invented the central terminal. Precordial leads are born.
1942
Goldberg used the central terminal with augmentation. Augmented unipolar leads are born.
1954
AHA standardized 12-Lead EKG as we know it now.

Figure 2.1: Evolution of ECG through history [4].

2.2 ECG concept and interpretation

The heart's electrical activity is represented as a set of waves in the ECG, each section of the waves correspond to a specific part of the cardiac circle. In this section, a general view of the ECG trace is provided, explaining the structure and interpretation of this type of physiological wave. [21].

2.2.1 ECG wave structure

Being familiar with the structure of the ECG will allow the user to detect abnormalities which may lead to a variety of diseases. Apart from a brief explanation of each of the parts, in Fig. 2.2 there is a schematic representation of the wave in order to see the parts graphically [22].

- **P waves.** This is the first event found in an ECG. This type of waves are defined by a small curvature which reflects the atrial repolarization, this marks the beginning of the cardiac electric activity. The morphology, duration and the relationship of the P waves with other events of the ECG offer insights about atrial conduction properties and possible anomalies.

- **QRS complexes.** This complexes are widely considered as the representation of ventricular depolarization, QRS complexes are represented as complex wave forms which group different deflections. The study of the morphology, duration of the QRS complexes helps identify ventricular conduction anomalies, it is also helpful in the localization of myocardial infarctions.

- **T waves.** This waves encompass the repolarization phase of ventricular myocardium, they embody the restoration of electrical equilibrium after depolarization. Alterations in the morphology, amplitude, and symmetry of the T waves may reflect myocardial ischemia, electrolyte imbalances, or drug-induced repolarization anomalies.

- **Intervals.** Intervals are defined as the time span between the events explained before, this intervals are essential metrics for characterizing conduction velocity and rhythm regularity. The interval between the P wave and the R peak (PR interval) defines atrioventricular conduction time, the QT interval, which goes from the Q wave to the T wave, represents ventricular depolarization and repolarization dynamics. Anomalies in the duration of QT intervals or PR segment deviation, may indicate the presence of arrythmogenic conditions. Finally, the ST segment is defined as the part of the ECG that goes from the end of the QRS complex to the start of the T wave. The ST segment represents the phase when the ventricles are fully depolarized. In a healthy patient, this part should be completely flat, elevation or depression of this segment might indicate the presence of myocardial infarction or ischemia.

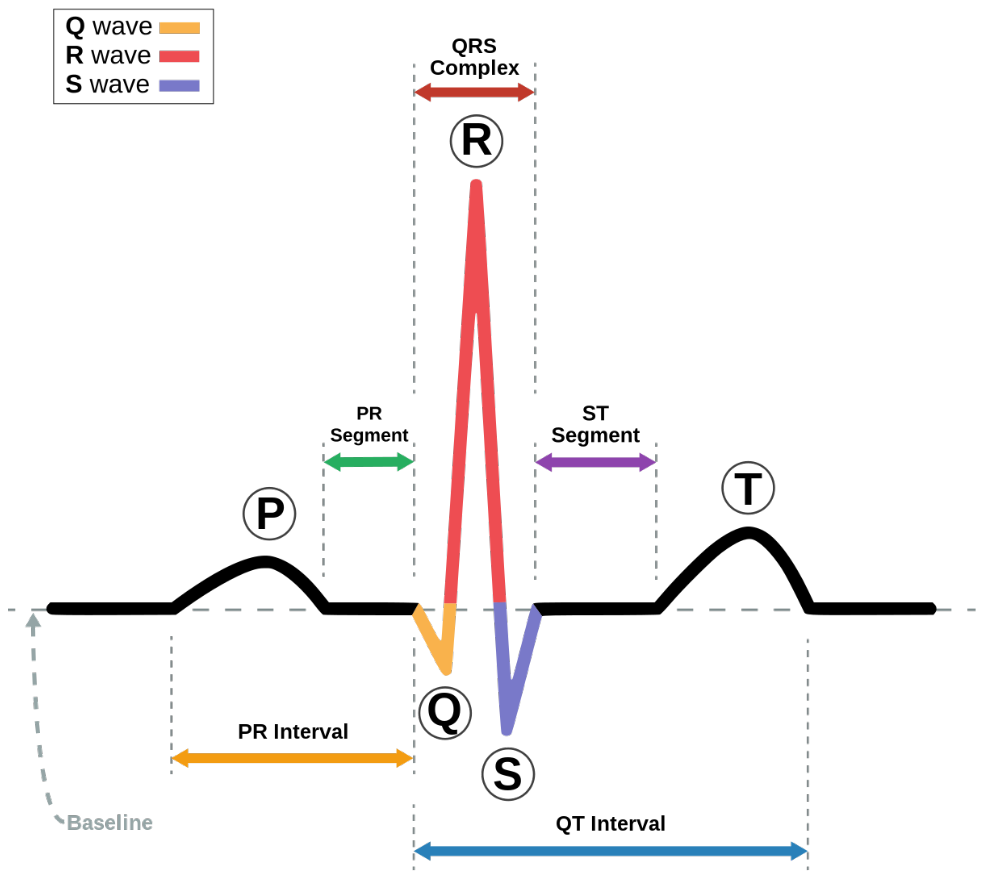


Figure 2.2: ECG trace of a healthy patient [51].

2.2.2 Cardiac Cycle

The cardiac cycle is the sequence of episodes that take place in the heart during a whole heart-beat. This sequence is characterized by the harmoniously work of both electrical and mechanical activities, it assures the correct pumping of the blood through the body.

Usually, the cardiac cycle is divided into two main parts, which can also be subdivided into more specific periods.

- **Systole.** In this period of the cardiac cycle the cardiac muscles contract, which leads to the pumping of the blood out of the chambers [23].
  - Atrial Systole. This phase is characterized by the contraction of the atria, which pushes blood into the ventricles. In the ECG, it is represented by the P wave, which indicates atrial depolarization.
  - Ventricular Systole. This event occurs right after atrial systole, it is defined by the contraction of the ventricles which pumps blood into the the pulmonary and aorta arteries. In the ECG, this phase is symbolized with the QRS complex, indicating ventricular depolarization.
- **Diastole.** In contrast with systole, in the diastolic phase, the cardiac muscles relax, letting the heart chambers be filled with blood once again [23].
  - Early Diastole. After the systole, the ventricles close which conduce to the closure of the aortic and pulmonary valves, and the opening of the mitral and tricuspid valves. This process allows the blood to circulate from the atria into the ventricles.
  - Late Diastole. This is the last phase of the cardiac cycle, in it the atria contracts until it completes the blood filling of the ventricles. This process prepares the heart for the next cardiac cycle.

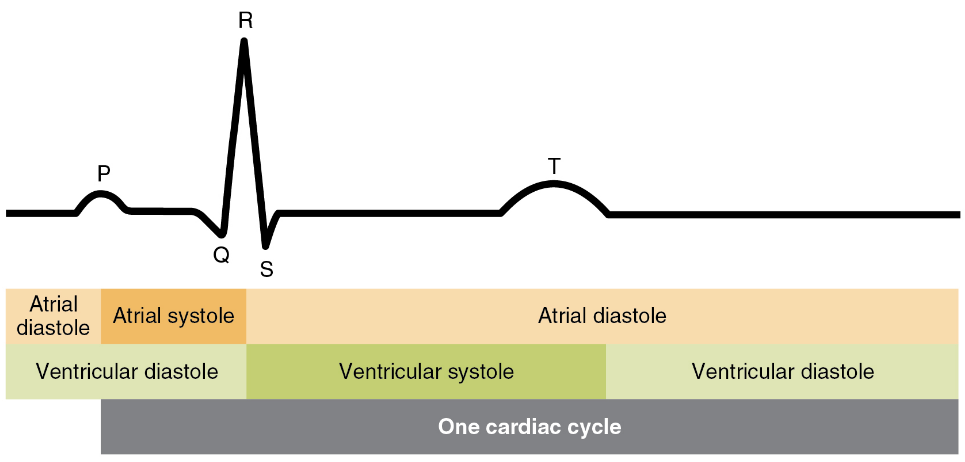


Figure 2.3: ECG wave with the phases of the Cardiac Cycle [52].

2.3 Diseases detected with an ECG

Cardiovascular diseases encompass an extensive variety of conditions that affect the heart and blood vessels. This group of illnesses is the leading group of morbidity and mortality worldwide. The ability to promptly detect these abnormal conditions is critical for effective intervention and treatment. In this context, electrocardiography offers helpful insights into the electrical activity of the heart helping in the identification of abnormalities that could lead to cardiac pathology [7]. This section will also include a brief description of the main diseases that can be detected in their early stages thanks to ECG.

2.3.1 Myocardial infarction

Myocardial infarction, also known as heart attack, is a very dangerous medical condition which consists of the abrupt interruption of the blood flow to the heart, this disease can lead to tissue damage or even necrosis [8]. The severity of the infarction depends on the size and location of the affected area, as well as the duration of the blood flow interruption. In several cases, the underlying origin of the myocardial infarction is the formation of a thrombus [8] in a coronary artery, which are the ones in charge of supplying the heart with oxygen-rich blood. This blood clots are caused by the rupture of atherosclerotic plaques, which are accumulations of cholesterol and fat over time. When the plaque breaks, it exposes the underlying tissue to the blood which triggers the formation of the thrombus.

Thankfully, the ECG, provides a precious information which is clutch in an early detection of the disease.

- **ST segment elevation.** There is a ST-Segment elevation of at least 1 millimeter in a standard ECG paper, which is equivalent to a 0.1 mV disturbance, which may indicate a trans mural myocardial injury, nevertheless, this elevation might be subtle, particularly in inferior leads, so a careful interpretation is required [9].

- **T wave inversion.** Following the above-mentioned ST-Segment elevation, a T-wave inversion may occur, this irregularity reflects repolarization abnormalities in the damaged myocardial area [10].

- **Pathological Q waves.** The formation of these pathological Q waves takes place in the later stages of the Myocardial infarction. The presence of these waves indicates a permanent damage in the cardiac muscles. Pathological Q waves may be found in up to 30 percent of patients suffering an infarction [11].

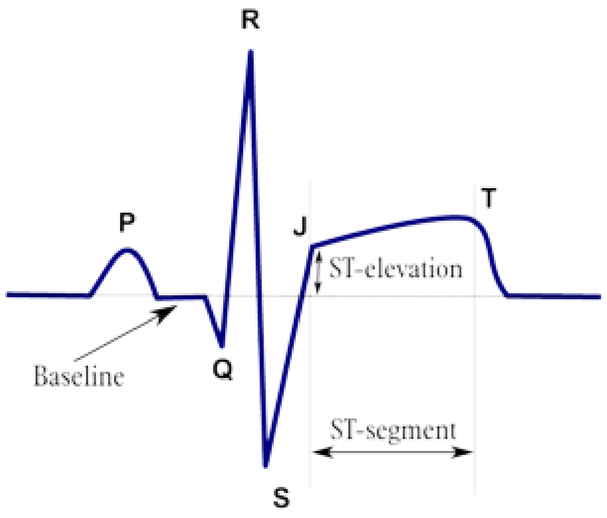


Figure 2.4: ECG with Myocardial Infarction [53].

2.3.2 Abnormal heart rhythms

Abnormal heart rhythms or commonly known as arryhtmias, are a set of cardiac disorders that are defined by an altered electrical activity of the heart, which may lead to an abnormal heart rate, it might be slower than usual (< 60 beats/min) or faster (> 100 beats/min) [12]. This disorders, can manifest as deviations of the systole and diastole, affecting the ability of the heart to pump blood. Arrythmias can have its origin in alterations in the initiation or conduction of electrical impulses, spontaneous firing of the cells or the reentry phenomena (abnormal circuits) within the heart tissue [13]. Any of these abnormalities can alter the normal electrical rhythm. The most common types of arryhtmias and their detection using ECG, which is an essential tool in order to detect them, are the following.

- **Atrial Fibrillation.** This is one of the most common type of arryhtmias, it is defined by a chaotic and accelerated electrical activity in the atria. The result of this is an irregular and usually rapid ventricular response. Nevertheless, it can be identified with an ECG, atrial fibrillation is characterized by the nonappearance of distinct P waves and an irregular R-R interval [14].

- **Atrial Flutter.** This condition is defined as a coordinated atrial tachycardia, it is accompanied by a rapid but regular ventricular rate. It can be detected using ECG because it presents 'sawtooth' patterns in the flutter waves [15].
- **Ventricular Tachycardia.** This type of arryhtmia is characterized by a rapid and altered heart rhythm which has its origin in the ventricles. On the ECG, it is defined by broad and curious QRS complexes at more than 100 bpm [16].

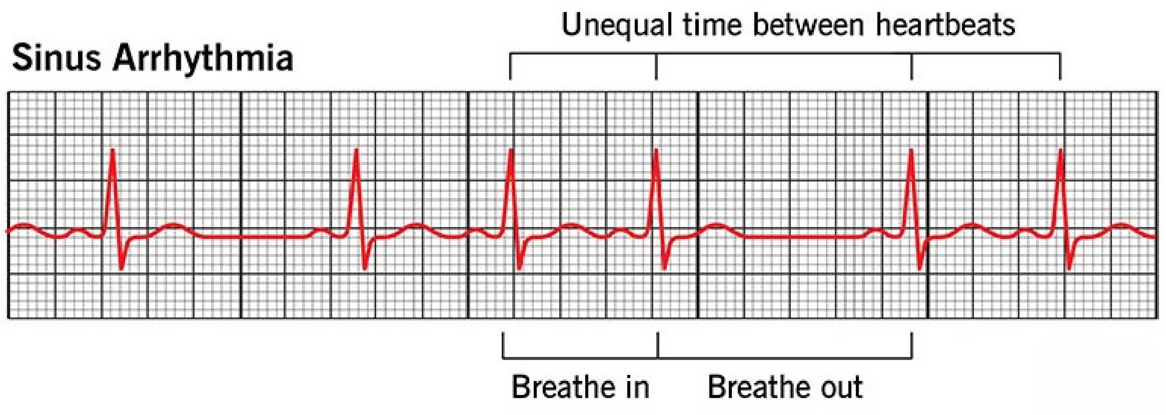


Figure 2.5: ECG with an Arrhythmia [54].

2.3.3 Enlargement of the heart

The enlargement of the heart or cardiac hypertrophy is characterized by the increase in mass and size of the myocardium as return to physiological or pathological stimuli. This phenomena is driven by the combination of hyperplasia (boost in the number of myocytes) and cellular hypertrophy (growth in the size of the individual myocytes). In the clinical field, cardiac hypertrophy can manifest as an enlargement in left ventricular wall thickness (Concentric hypertrophy) chamber size (Eccentric hypertrophy) or a combination of both of them. Another classification regarding this disease is between physiological hypertrophy and pathological hypertrophy [19]. Physiological hypertrophy takes place in response to a physiological stimuli such as physical activity or pregnancy, it is defined by the increase of the heart size and function but does not have negative effects on the cardiac function or structure. On the other hand, the pathological hypertrophy that occurs in response to pathological stimuli valvular heart disease, volume overload or chronic pressure, in contrast with the physiological hypertrophy, pathological hypertrophy is related to structural and functional anomalies which could predispose the patient's heart to arrythmias, heart failure or sudden cardiac death [18].

Regarding the hypertrophy detection using an ECG, this condition can be detected by paying attention to some changes in the ECG. First of all, we can find an increased QRS Voltage, which reflects augmented electrical activity of the hypertrophied heart muscle [19]. ECG findings in cardiac hypertrophy might include ST-segment and T-wave anomalies, which reveal alterations in ventricular repolarization caused by the structural changes in the heart muscles. Finally, another characteristic associated with cardiac hypertrophy in the ECG are the arryhtmias, which may manifest as abnormal heart rhythms on the ECG in the patients with this condition [20].

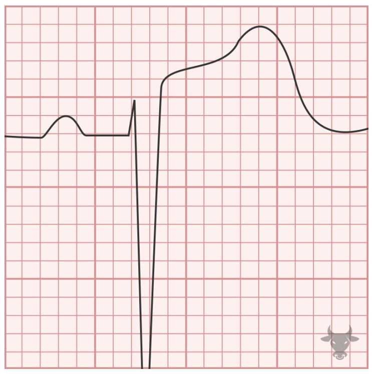


Figure 2.6: ECG with left Ventricular Hypertrophy.

2.4 Sensors for Cardiac Monitoring

One of the most important components of the ECG, is the sensor that will be used to register the heart's electrical activity. Manufacturers offer an extensive variety of sensors, each one designed with different features and benefits to tackle diverse needs, ranging from professional use sensors to general public devices.

In this section, the user can find a brief description of two commercial available sensors that were not selected as the final sensor for this project along with a comparison of them with the AD8232 sensor, which was the final choice.

2.4.1 ADS1292R

The ADS1292R sensor, which is manufactured by Analog Devices, is an Analog Front-End (AFE) device that is specifically developed for the measurement of ECG and other bio potential applications. An AFE device is a circuit that processes an analog signal detected through a sensor, in this case the signal is an ECG, and converts it into a digital signal before transmitting it to a digital system, in this project the Arduino Mega 2560. This sensor is fabricated with the goal of being versatile, making its use feasible in a wide range of monitoring scenarios. A brief description of the main properties of this sensor is given in the following bullets [25].

- High performance. Thanks to its advanced filtering capabilities, this sensor can effectively reduce noise and artifacts providing extraordinary signal quality with low noise, which can go as low as 4.5μV and a resolution of 24 bits.
- Low Power Consumption. The AD1292R sensor presents very low power consumption batteries, with an operating current of just 350 μA per channel at a 250 Samples per Second (SPS) data rate. This feature makes this sensor really valuable for portable and wearable monitoring systems.
- Complexity in configuration. The complexity in the configuration and integration might be the biggest drawback regarding this sensor. First of all, the mandatory need of a micro controller such as an Arduino board, forces the developers to ensure appropriate communication and synchronization between the sensor and the microcontroller. It is true that the chosen sensor also requires the use of a microcontroller, the AD1292R sensor requires a accurate SPI interface setting and time constraints management. Even a trivial misconfiguration in clock speed could lead to communication errors. Apart from this,

the AD1292R sensor gives a wide variety of configurable settings, this allows the user to customize the system as much as possible, but also adds difficulty since an incorrect configuration may lead to an erroneous performance of the sensor. This incorrect performance of the device could be caused by an erroneous reference voltage setting or incorrect bias current settings. This could make developers face a steep learning curve.

- Noise Sensitivity. Since this sensor is an AFE module, it is susceptible to electromagnetic interference and noise from other electronic elements. This sensor provides with high gain amplifiers because of the weak nature of biological signals, the drawback of this necessary property of the sensor is the amplification of the undesired noise signals. Maintaining a stable and good quality signal could be difficult in environments with important electrical noise and interference. Taking into account all of this, is essential to have a proper shielding to minimize the noise and have a good quality signal [24].

2.4.2 MAX30003

The MAX30003 sensor, designed by Maxim Integrated, is an integrated biopotential AFE device that was specifically conceived for the design of portable ECG systems. This sensor admits SPI communication, which allows it to adapt a diverse variety of microcontrollers. Here is a brief description of its main features [26].

- Integrated features. This sensor has some features which makes it a good choice when creating an ECG application. It has lead-on/lead-off detection that warns the user when the electrodes are not correctly attached to the skin. It also has good motion artifact reduction, which helps to filtrate the artifacts that are triggered by the patient's movement.
- High accuracy and resolution. The MAX30003 has an 18-bit resolution and a large sampling rate that provides highly precise measurements that facilitate the detection of anomalies.
- Complexity in configuration. The MAX30003 sensor, as it happens with the AD1292R sensor, requires a foreign microcontroller to register the signal, which adds complexity to the whole systems. This, added to the fact that it has a lot of customizable parameters, could lead inexperienced developers to misuse this sensor.

2.5 Regulatory Considerations

Since the ECG is a medical device, it is subjected to very scrupulous regulatory standards, which are necessary to ensure the safety, efficacy, and reliability of the devices. These standards are crucial in order to protect the patient's health and integrity and also to assure that the apparatus works correctly under diverse conditions. In this section, a brief description of the regulation standards in the European Union is given, since it is where the development of this device is made.

- **ISO 13485.** This international regulation ensures the compliance of requirements related to a comprehensive quality management system specific to the medical apparatus industry. Manufacturers have to demonstrate their capacity to provide the customer a non-dangerous device that fulfils multiple regulatory requirements. Here is a description of the crucial aspects covered by this standard [27].
  - Risk management. This standard focuses on the identification and control of the risks during the whole product life cycle, from the design of the device to the manufacture and commercialization of it. This ensures that the potential risks are minimized and the device is safe for patient use.
  - Quality assurance. This regulation emphasizes on the maintenance of the highest quality standards in the whole development process. This includes strict controls over the fabrication processes, supplier quality management and product testing to ensure that the medical device meets the excellence criteria.
- **IEC 60601-1.** This includes a set of standards for the safety and performance of an electrical medical equipment. It comprises general requirements for basic safety and optimal performance. Compliance with the IEC 60601-1 is compulsory in the bast majority of regions, including the European Union. The main characteristics of this standard are described below [28].
  - Electrical and mechanical safety. IEC 60601-1 ensures that the medical device works safely under diverse electrical conditions. To assure this, electrical insulation tests or leakage current tests are performed. On the other hand, regarding mechanical safety, this standard addresses possible hazards such as sharp edges or moving parts. These tests are performed in order to have a mechanically robust and safe device, preventing physical injuries to users and patients.

  - Electromagnetic Compatibility (EMC). This standard also ensures that the medical apparatus does not emit Electromagnetic Interference (EMI) which could alter the functioning of other devices and to ensure that the device is not affected by electromagnetic interference coming from other sources. This is fundamental for maintaining an effective performance of the ECG since places like hospitals have notable electromagnetic interference sources.

- **CE Marking.** In order to commercialize an ECG monitorization device in the European Union, it must have the CE mark, this label indicates the compliance with the European Medical Device Directive (MDD) or Medical Device Regulation (MDR). The CE marking is an essential step in proving that the device complies with the safety, health, and environmental protection requirements. Here are the principal characteristics of this labeling [29].

  - Conformity Assessment. This describes a meticulous examination of the design, fabrication process and performance of the device assuring that it complies with the standards. Depending on the risk classification of the apparatus, the assessment can be managed through self-assessment or requires engagement from a Notified Body. This organism is an independent certification organization designated by a member state of the EU. This organization will be in charge of reviewing the documentation and performing product testing among other actions.
  - Technical Documentation and Clinical Evaluation. Developers have to compile technical evidence that demonstrates the compliance with the MDR and MDD mentioned above. On the other hand, a clinical evaluation has to be carried out to verify the device safety and performance. This includes conducting clinical trials and compiling a clinical evaluation report.

2.6 Modern and future advancements in portable ECG monitoring devices

Recently, the cardiac monitoring industry has focused its efforts on the design and development of portable ECG devices. To achieve this goal, advancements have shifted towards developing devices that are compact, robust, and accessible for a wider and less specialized audience. This new direction in the ECG technology sector is due to an increasing demand for more flexible and easier cardiac monitoring options that can integrate into everyday routines [31].

In this context, the most relevant factors to fulfill the healthcare demands are summarized in the following section:

- **Miniaturization of Components.** The goal of manufacturing smaller components is to ensure practicality, making the device easier to store and manipulate. This can be accomplished due to advances in microelectromechanical systems technology, which includes the use of semiconductors in the fabrication of devices and creating sensors as powerful as possible while maintaining a small size. The long-term objective is to embed these sensors into patches or even clothing, making continuous monitoring practical less intrusive [32].

- **Improved User Experience.** The rise of these portable ECG devices has made cardiac monitoring accessible to a wider population [30]. This is a significant breakthrough since patients no longer have to go to the hospital or health center for routine ECG monitoring. However, this means that developers must also focus on making these devices as user-friendly as possible, designing intuitive graphical interfaces and data presentations that are easy to understand. These advances are especially useful for patients located in areas far from hospitals.

- **Early detection and preventing care.** The importance of being able to commercialize this type of device to the general public lies in providing data in real-time, which enables continuous monitoring. This continuous monitoring is crucial for the early detection of cardiac anomalies. Early detection can prevent the progression of various cardiac illnesses and reduce the risk of serious events such as strokes or ischemia. The commitment to preventive care through these devices can lead to improved patient welfare and reduced healthcare costs [32].

- **Integration with Telemedicine.** In recent times, telemedicine is acquiring a very important role in the healthcare field, therefore, devices that the patient is able to use at home without professional help are becoming very important. This type of devices allow medical professionals to monitor patients' cardiac health remotely, carry out virtual consultations effectively, and provide advice immediately due to the real-time data provided by this tools. This facilitates and enhances the ability to manage, for instance, chronic cardiac diseases eliminating the need to go to the hospital or healthcare center.

Chapter 3

Materials and Methodology

3.1 Components Selection

The design and manufacturing of an ECG device includes a cautious selection and integration of the components to assure effective and reliable cardiac monitoring. For this project, the AD8232 sensor and the Arduino Mega 2560 were selected as the central components that will form the ECG device and will ensure the correct performance of the cardiac monitoring device. The performance of these elements will be complemented by a 3D printed enclosure or supports and a custom PCB, which are essential in the design of a compact, and portable ECG device.

The AD8232 sensor provides high quality and low noise signal detection, which is essential for a reliable cardiac monitoring. On the other hand, the Arduino Mega 2560, offers the processing power for real time data management and system control. These two components along with the 3D supports and the PCB, create a user friendly ECG monitoring device which is suitable for domestic and professional use.

3.1.1 Arduino Mega 2560

The Arduino Mega 2560 was chosen as the microcontroller for this ECG monitoring project due to its capabilities and meeting with the project requirements. These capabilities include processing power, input/output competences or memory capacity [33], [34].

- **Processing power and performance.** The 'heart' of the Arduino Mega 2560 is a microcontroller, the ATMega 2560, which works at a clock speed of 16MHz. This processing speed is more than enough to deal with the real-time signal acquisition and processing labours required by the ECG monitoring system. The ATMega 2560 allows the user to manage multiple actions in parallel, such as reading analog signals, processing the signal data, and communicating with the GUI to send that signal. This microcontroller ensures an efficient performance without any lag or delay.

- **Input/Output capabilities.** The Arduino mega 2560 offers 54 digital Input/output pins and 16 analog input pins. Even though not all pins are being used in this project, it is very useful to have a large number of pins since it offers the possibility of adding extra gadgets in the future, such as additional sensors or wireless communication. On the other hand, there are 15 Pulse Width Modulation (PWM) outputs that allow an accurate control of elements such as LEDs or motor drivers, in this project they are not used but could be useful in future improvements of this device.

- **Memory Capacity.** This microcontroller has 256 kB of flash memory, even though 8 kb of it is used by the bootloader, the remaining 248 kB are more than enough to provide extensive space used for storing complex code and libraries that are mandatory for signal processing and communication. Alternatively, the Arduino Mega 2560 has 8 kB SRAM and 4 kB EEPROM which is sufficient to ensure that there is abundant volatile and non-volatile memory to deal with data storage, variables, and program execution without memory flow issues.

- **Communication Interfaces.** The Arduino Mega 2560 features four hardware UART ports which provide reliable serial communication. This is crucial for the data transfer between the micro controller and the GUI. It is also very important for future improvements in the device since these ports are key for wireless communication. This microcontroller includes two communication protocols, the 12C and the SPI. These protocols have to be taken into account in future improvements of the project since they allow the connection of additional sensors and modules like external memory units or wireless communication devices.

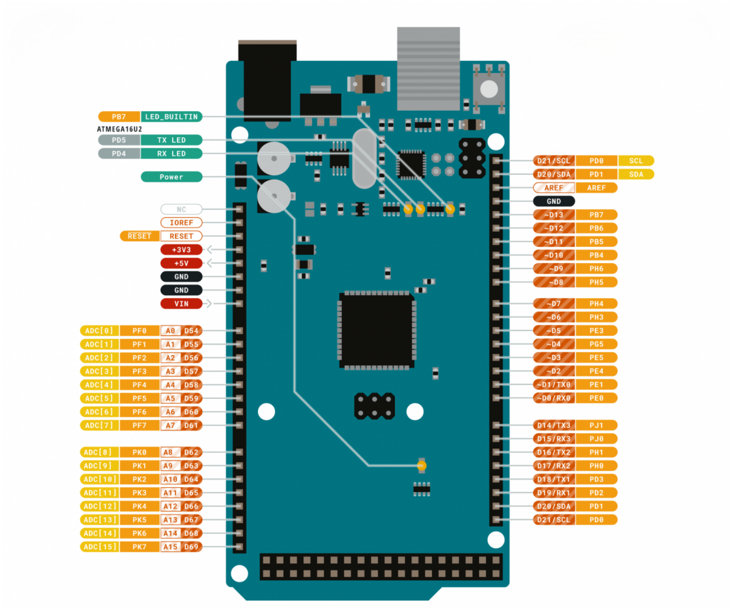


Figure 3.1: Schematic of the Arduino Mega 2560 [49]

3.1.2 AD8232 Sensor

The AD8232 sensor is an AFE component, which processes the analog signal detected through the sensor and converts it into a digital signal before further processing, created for the measurement of ECG signals. It is designed to extract, amplify, and filter weak bio potential signals in environments with important electrical noise like those created by the electrode placement. This sensor is commonly used in wearable health monitoring systems, fitness applications, or in portable ECG devices. This effectiveness is due to its low power consumption, small size, and high performance. Regarding the hardware of the sensor, it is a robust, surface-mount package, which makes it very adequate and easy to integrate into a PCB. Its small size and minimum need for external components simplify the design of the whole device, reducing the overall footprint of the ECG monitoring system.

The AD8232 sensor has a powerful amplifying instrumentation that provides great common-mode rejection, this is the property of the amplifier that rejects or ignores common-mode signals. In the case of ECG signal detection, this common-mode signal could be electrical noise or electromagnetic interference. This sensor also provides minimal offset voltage, which is a small voltage that might appear in the output of an amplifier even when there is no input signal. A small offset voltage guarantees that the amplified ECG signal is a reliable representation of the activity of the heart. These two properties assure a precise amplification of the ECG signal.

The gain of this amplifier could be regulated by the user if needed via external resistors. The sensor also integrates a Right-Leg Drive (RLD) amplifier. This adaptability allows the sensor to conceive a large range of signal levels and noise conditions. This piece is also essential in driving the common-mode voltage of the patient to a stable level, which is yet another improvement regarding the rejection of external electrical interference [35].

The AD8232 sensor is equipped with a fast restore function. This feature allows the sensor to recover the ECG signal after prolonged disorders, like, for example, the disturbances caused by defibrillation shocks. This results in continuous and accurate monitoring.

The AD8232 operates on a supply voltage range from 2.0V to 3.5V (in this project, the sensor is connected to the 3.3V supply) and usually consumes only 170µA. This can be very useful in future improvements of the device, making the system become wireless [35].

The AD8232 sensor is conceived to work with a three-lead ECG configuration, which consists of three electrodes placed in specific parts of the body to measure the electrical activity of the heart.

In summary, the AD8232 is a compact, low-power consuming, and easy-to-integrate option for biopotential analysis applications, such as a portable ECG monitoring device. Its design and features ensure great signal quality, power efficiency, and flexibility to operate together with other sensors and components, such as wearable health monitoring devices, portable monitoring systems, or fitness applications.

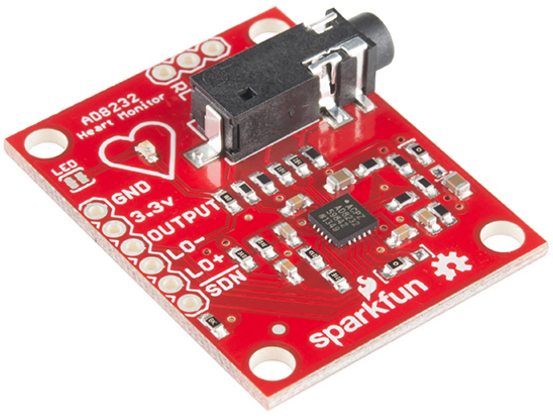


Figure 3.2: AD8232 sensor [50]

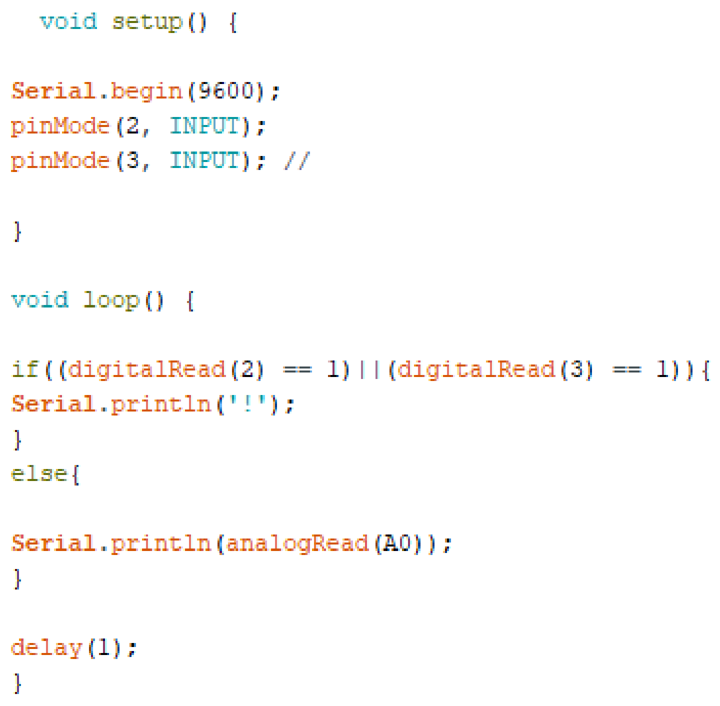
3.2 Code

3.2.1 Arduino Code

In this section, a description of the Arduino IDE code is given. This code creates the connection between the microcontroller and the AD8232 sensor. It prints the detected signal in the Serial Plotter tool of Arduino IDE. This code has been implemented in order to test the device principle of operation and it is useful to understand the connection between the two components and to check if the Python code is done correctly. The explanation of the code will be done following the order shown in Fig. 3.3.

The code can be found in 6.2. First of all, there is the setup function, this function will only be executed when the connection starts or when the Arduino board is restarted. Its role is to initialize all the settings. The first line of code starts the serial communication with a baud rate of 9600 bits per second. This line is essential since it authorizes the Arduino to send data to the computer through a USB connection. The next line establishes pin 2 as an input, this selection is arbitrary but the developer has to ensure that the pins are the same in the code as in the connection among the components. This pin will be used for lead-off detection (LO+). The last line of this function does the same as the above one, it sets the pin 3 as input and it will be used for lead-off detection too (LO-).

The second function is the loop function, which runs right after the 'setup' function. This function encompasses the main logic of the program. The first element found in this function, is an 'if' statement, if we go further into that statement we find two conditions, whose purpose is to check if the value received from each pin is 'HIGH' (which is the same as saying equal to one in digital logic). If the value received in any of the pins is equal to one a '!' symbol is going to be printed, indicating the presence of an error. When either of the pins receives 'HIGH', it means that the electrodes are not properly attached to the body of the patient or that there is some kind of issue with the serial connection. If neither of the pins sends a 'HIGH', the block of code inside of the 'else' statement starts running. In the next line, the 'analogRead(A0)' part, specifies that the value is being read from the analog pin A0, this pin is also arbitrary, but, once again, the pins specified in the code have to be aligned with the ones used between the components. This value ranges between 0 and 1023, it represents the analog voltage between 0V and 5V. Then this part is used in the next line to deliver the analog value to the serial monitor for it to be represented graphically. The final line of this code is in charge of pausing the program for one millisecond after each loop, the number of milliseconds is decided by the developer. This delay is essential to prevent the serial buffer from saturating because of the large amount of data received, this will ensure a more reliable and understanding output on the serial monitor.



```
  void setup() {

Serial.begin(9600);
pinMode(2, INPUT);
pinMode(3, INPUT); //

}

void loop() {

if((digitalRead(2) == 1)||(digitalRead(3) == 1)){
Serial.println('!');
}
else{

Serial.println(analogRead(A0));
}

delay(1);
}
```

Figure 3.3: Arduino IDE code

3.2.2 Python Libraries

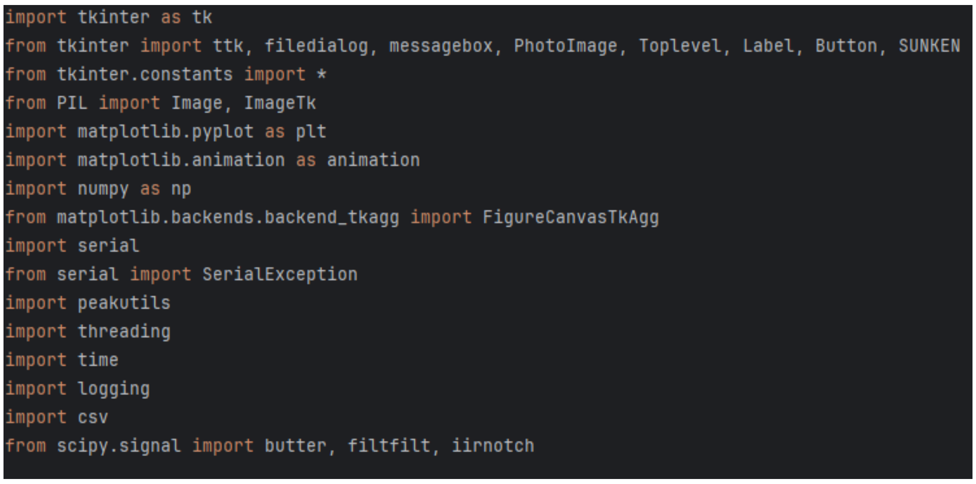
For the development of this project, a large variety of external modules were needed in order to satisfy all the needs of the device. In this section, a brief description of each of the modules imported is given. The order to describe these modules is going to be the one shown in Fig. 3.4.

- **Tkinter Modules.** The Tkinter module is mostly used for the creation and customization of the application window. It is imported as tk in order to avoid naming conflicts and brevity [36].
  - Ttk. This appends widgets that are more modern compared to classic Tkinter widgets
  - Filedialog. This enables the user to open file dialogs to allow the selection of a file or directory.

  - Messagebox. It is used to exhibit message boxes that are used for warnings, errors, or give information.
  - Photoimage. It is used to display images.
  - Toplevel. This is used to create additional windows apart from the main one.
  - Label and Button. These are basic GUI widgets for displaying text and interacting with the user.
  - SUNKEN. This is a purely aesthetic import. It only affects the appearance of the elements.

- **PIL Modules.** This import is divided into two sub-imports. The first, called 'Image,' is used for opening, saving, and interacting with images in a wide variety of formats. The second, 'ImageTk,' is primarily used to display PIL images in Tkinter applications.
- **Matplotlib Modules.** The Matplotlib module is a very powerful tool to create visualizations in Python. It is a very large module, so its input will be divided into three imports. The first import is 'import matplotlib.pyplot as plt' the pyplot library is used for creating fixed, animated, and interactive visualizations in Python. The next one with name 'import matplotlib.animation as animation' offers support in the creation of animations using Matplotlib. Finally, the import 'from matplotlib.backends.backend-tkagg import FigureCanvasTkAgg' is used to display the Matlplotlib creations in Tkinter applications, in this case, the GUI [37].
- **Numpy Module.** This module is imported as 'np' to avoid naming conflicts. The Numpy module offers support for large, multi-dimensional arrays and matrices, accompanied by a wide variety of mathematical functions to treat those arrays.
- **Scipy Modules.** This module is very useful for signal processing. From this module, we obtain the three imports used for signal filtering.
  - Butter. This is used to create the two Butterworth filters that were applied.
  - filtfilt. This module is mandatory when more than one filter is applied.
  - iirnotch. This library is used to create the Notch filter.
- **Serial Module.** This module is used to create the serial communication through the COM port. Specifically, the 'SerialException' is imported from the Serial module to handle exceptions related to serial communication.

- **Peakutils Module.** This is used for detecting peaks in data.
- **Threading Module.** This library is used to execute codes in multiple threads for current execution.
- **Time Module.** This adds different time-related functions.
- **Logging Module.** This is employed for logging messages to a file or console.
- **CSV Module.** This module is used to read and modify Comma Separated Values (CSV) files.



```python
import tkinter as tk
from tkinter import ttk, filedialog, messagebox, PhotoImage, Toplevel, Label, Button, SUNKEN
from tkinter.constants import *
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import serial
from serial import SerialException
import peakutils
import threading
import time
import logging
import csv
from scipy.signal import butter, filtfilt, iirnotch
```

Figure 3.4: Libraries imported in Python

3.2.3 Filters applied to the signal

The three filters that are being applied to the detected signal are the Butterworth high pass filter, the Butterworth low pass filter and, the Notch filter. Below there is a detailed explanation of each filter along with the reasoning behind this choice for the project.

The Butterworth high pass filter is a filter that is conceived to allow frequencies that are over a cutoff frequency which is decided by the developer, the frequencies that are below this cutoff will be attenuated. The Butterworth high pass filter is characterized by not presenting ripples, in consequence, it avoids distortion of the signal within the specified frequency range, in other words, it has a flat frequency response in the pass band.

The main reason to use the Butterworth high pass filter is to eliminate the characteristic baseline wander and low-frequency noise which mutates the signal completely. This baseline noise is normally caused by the respiration and movement of the patient. Apart from that, this filter offers a flat response in the pass band so the important ECG features, being the P and T waves and the QRS complex, are not affected by this filter preserving the nature of the ECG signal [38].

On the other hand, the Butterworth low pass filter is designed to allow frequencies below a cutoff frequency decided by the user, frequencies that are over this cutoff frequency will be attenuated. As it was explained above for the Butterworth high pass filter, this filter has a flat frequency response in the pass band, which is very useful to avoid distortion.

This filter is used to eliminate high-frequency noise from the ECG, this noise can have its origin in multiple sources, such as muscle contraction and electromagnetic interference. With the choice of a cutoff frequency, in the case of this project 40Hz, this filter will attenuate high frequencies. With this filter, the user ensures the preservation of crucial information for heart rate and rhythm analysis [39].

Finally the Notch filter, or band-stop or band-reject filter, is a filter that is developed to remove a specific narrow band of frequencies, but it allows other frequencies to pass through mostly straightforwardly. Regarding ECG signal processing, the Notch filter is widely used in order to eliminate the power line interference at 50Hz.

The power line interference, is usually at 50 Hz or 60 Hz varying with the region, for this project it is fixed at 50.0 Hz, this is a very usual origin of interference in ECG signals. If this power line interference is not filtered out, it will substantially degrade the quality of the ECG. The Notch filter is a safe option for ECG filtering, since this filter was developed to decrease the narrow band frequencies related to the power line interference. This will effectively eliminate the power line noise while leaving the rest of the frequencies untouched. Moreover, by removing the power line interference, the Notch filter improves the signal-to-noise ratio of the ECG signal, which provides clearer and more accurate readings [40].

In conclusion, the combined usage of the Butterworth high pass filter, the Butterworth low pass filter and the Notch filter, removes all the principal noise and interference present in the ECG. Each of the filters performs an essential role in assuring the clarity and veracity of the signal. These filters, with their specific properties, were considered the best options in order to ensure a reliable and accurate monitoring of the cardiac activity.

3.2.4 Software for PCB design: Fritzing

The selected software used for the creation and development of the PCB, is Fritzing [43]. Fritzing is an easy-to-use and open-source software that is used to create PCBs for simple projects conceived as a hobby and also for complex project that feature a diverse variety of components.

Fritzing is a powerful tool that encompasses multiple stages of electronic design in the same environment. The primary services that the software uses include schematic capture, breadboard layout and, the PCB design itself, making it a really useful tool. It has a very complete feature set, allowing the developer to upload additional libraries. Here is a brief description of the most important features of Fritzing.

- **User Interface.** Fritzing has an intuitive drag-and-drop interface that clarifies the designing process. Thanks to this, developers are able to place elements, create connections, and envision the circuit easily without the need of much previous experience designing PCBs. As already mentioned before, this software integrates the different phases of electronic design, schematic capture, breadboard prototyping and PCB layout. This integrated view helps the developer when it comes to minimizing errors that can take place when transitioning from one stage to another.

- **Extensive component library.** Fritzing includes a library with a wide variety of pre-designed elements, which facilitates the PCB design. The Arduino Mega 2560 was among the pre-designed elements of Fritzing, but the AD8232 sensor had to be uploaded externally.

- **Visualization tools.** The software provides with very complete visualization tools which allows the developer to inspect the design in 2D or even 3D. This allows the user to verify the component displaying and connections assuring that the design complies with the user's demands and necessities.

3.2.5 Software for 3D Design: TinkerCAD

TinkerCAD is the chosen tool for the 3D modeling of the two support pieces and the container housing all the components of the ECG monitoring system, including the AD8232 sensor, the Arduino Mega 2560, and the PCB. TinkerCAD is an online 3D design and CAD software that is user-friendly and simple, making it an excellent choice for designing and prototyping basic 3D models [55].

One of the standout features of TinkerCAD is its flexibility for cooperative work and remote access. As a web application, it requires no installation, allowing users to access it from any computer with an internet connection. This is particularly beneficial for collaborative projects, as team members can easily share and edit designs in real time, regardless of their location.

TinkerCAD's interface is another significant advantage. It is highly intuitive and designed to be easy to learn, ensuring that even users with minimal experience in 3D design can quickly become proficient. This ease of use promotes an efficient and effective design process, allowing developers to focus on creativity and problem-solving rather than learning the function of complex software tools.

In addition to its user-friendly design, TinkerCAD has a large and active user community. This community is a valuable resource, providing a wealth of tutorials, forums, and shared designs that can offer guidance and inspiration throughout the modeling process. Whether users need help with specific technical issues or are looking for creative ideas, the community support available through TinkerCAD is an asset to take into account.

Chapter 4

Design and Development

4.1 Hardware Design

4.1.1 PCB

The PCB serves as an element for interconnecting the electronic components of a device. The materials used for PCB manufacturing usually are non-conductive elements, such as fiberglass, with conductive pathways printed onto the surface of the board. The material often used for this pathways, is copper, since it has to be a conductive material. These pathways create the electrical connections among the components, easing the transmission of current enabling the functionality of the whole system.

In order to ensure a proper understanding of the concept of PCB and its benefits, here is a brief description of the elements that form a PCB:

- Substrate. This is the base material of the PCB, the material used is usually fiberglass (FR4). The function of this part is to provide mechanical support and insulation.
- Copper Layers. There are thin layers of copper printed in order to form the circuit patterns. These layers can be present on more than one layer separated by insulating material.
- Solder Mask. It is a protective coating applied over the copper layers in order to avoid short circuits and corrosion. It also helps in the soldering process.

The use of a PCB provides compactness and miniaturization of the developed device. It allows to connect a large amount of components in a reduced space, which is essential in nowadays electronic devices, since the user demands a small-sized device without compromising functionality. The use of PCB also boosts the durability and efficacy of the device, since it provides robust mechanical support and assures reliable connections between the elements.

A PCB is designed with the help of a CAD software, for this project the chosen software was Fritzing. A very extensive and varied database of circuit elements and libraries is available for Fritzing software, the AD8232 sensor used in this project is included in this database. It has been downloaded from the Sparkfun Electronics website [42]. If the developer is not able to find the components being used on a project in Fritzing's library or on the internet neither, elements used in Fritzing can be created manually, but this was not the case for this project.

The first step in the creation of the PCB, is to create a new file to store any progress made by the user. Then, the AD8232 sensor that was previously downloaded from the manufacturer's website has to be uploaded.

After that, in the PROTOBOARD VIEW, the developer dragged the Arduino Mega 2560 (which was taken from Fritzing's library) and the AD8232 sensor. In that view, the user created the connections between the two elements. The LO+ pin from the AD8232 sensor is connected to the pin number two from the PWM pins from the Arduino. The LO- pin is connected to the PWM pin three from the microcontroller. The output pin from the sensor is associated with the A0 pin from the ANALOG IN pins from the Arduino board. The pins used of the Arduino board for the explained connections are decided by the developer, but it is crucial for the operation of the final device that the pins are the same than the ones specified in the code of the ECG system. The 3.3V pin from the AD8232 sensor, also called supply pin, is connected to the 3.3V pin from the POWER pins of the Arduino board. Finally, the Ground pin of the sensor is also connected to the Ground pin from the Arduino board.

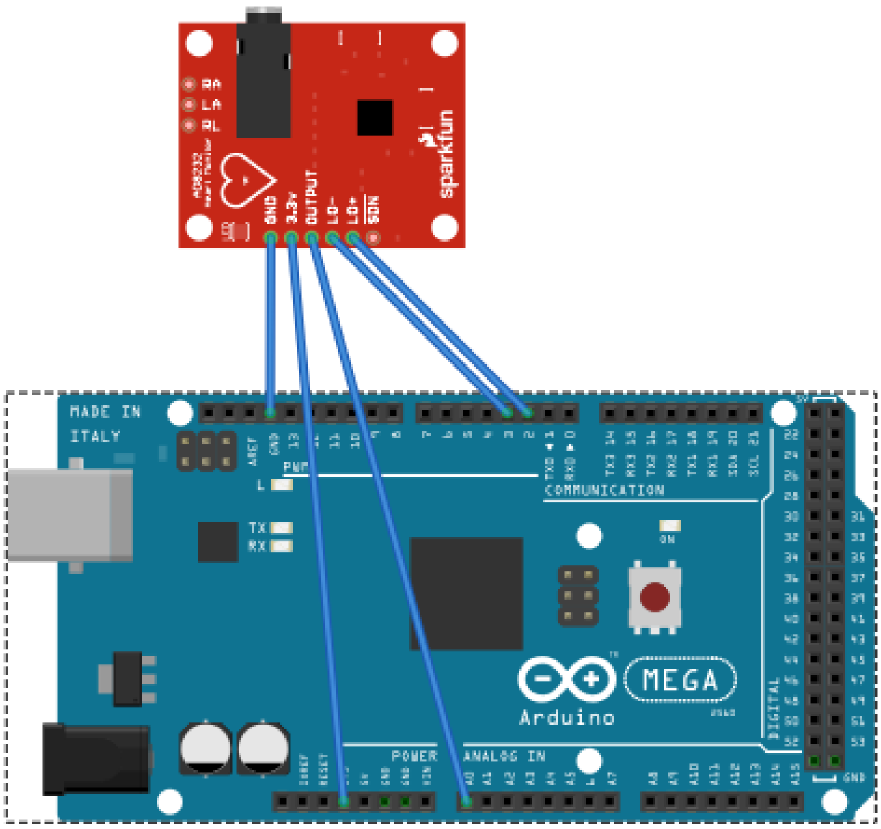


Figure 4.1: Schematic view of the connection between the AD8232 and the Arduino

Once all the pin connections are made in the SCHEMATIC VIEW, the view is switched to the PCB VIEW tab. This is the most relevant step, this is when the PCB is really created. The AD8232 sensor and the Arduino Mega are placed on the PCB in order to optimize the layout. The goal in this view, is to decrease the amount of traces that are created and to avoid crossing paths. These routes are created just by clicking on a component's pin and dragging it to the corresponding pin of the other component, this traces will correspond to the wire connections created on the PROTOBOARD VIEW. The first traces created were the power and the ground traces, this order is essential to ensure the stability of the circuit. The ground-to-ground connection between the sensor and the microcontroller is distinguished by its yellow color, while the rest of the connections were made using orange.

The next step after the creation of the traces, is to en-route the elements. Fritzing provides two options for routing, it can be made manually or automatically, since this project does not involve complex connections among the elements, the manual en-routing option was chosen. During this enrouting process it is crucial to take into account the design rules.

Finally, after ensuring that no mistake was made during the whole designing process, the developer proceeds to export the design file in Gerber file format and saves it in the desired destination. Then the Gerber file was uploaded to the AISLER [41] website in order to get the PCB printed. It is necessary to specify details about the PCB such as the size or the number of layers before submitting it. Once the PCB is received, multiple connection pins were soldered onto the PCB, the AD8232 sensor and the Arduino Mega 2560 were connected to those pins.

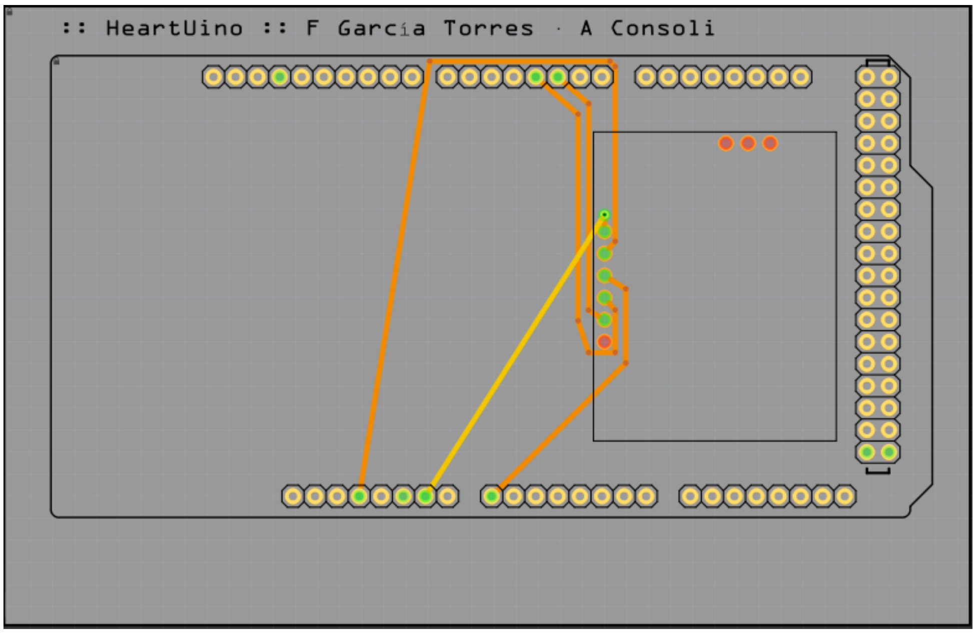


Figure 4.2: PCB view of the connection between the AD8232 and the Arduino

4.1.2 3D printed enclosure

One of the most important benefits of creating 3D designs instead of using pre-made elements is customization, by creating and manufacturing the components of the project, the developer is able to accommodate them specifically to the size and requirements of the other components of the device, in this case, the PCB along with the AD8232 sensor and the Arduino board, assuring an improved functionality. The other main advantage of 3D printing, is its cost-effectiveness, since it is notably cheaper to 3D print elements, especially if it is a unique or hard-to-find piece, than to buy a pre-made solution.

It is fair to say that possible commercial solutions also have remarkable benefits, such as the robustness of the piece. The industrial machinery used to print pieces tends to create stronger pieces, which eliminates the possibility of wobbling or even breaking in extreme cases. It is also true that surface finishing may be 3D printed pieces' major weakness, having parts with rough or uneven appearance, in the majority of cases this is just an aesthetic drawback. Finally, The developer has to be aware of the different factors surrounding the 3D printer itself, such as printer calibration, the quality of the filament used or printing settings.

The first 3D design created consists of two supports as the one shown in Fig. 4.3, this supports will hold the PCB horizontally straight, attaching the sensor and the micro controller on each of the sides of the surface of the board. These supports are very useful in terms of space efficiency, holding the PCB this way allows the user to save more space than leaning it on a flat surface. Apart from that, orientating the PCB like this improves the airflow among the elements of the device, which can help to cool down the components when a prolonged use of the device is made.

Nevertheless, this design also presents some drawbacks, such as its fixed size, this means that this design is specifically conceived for the dimensions of this device, if the user aims to make future improvements in the system these supports might be obsolete and not capable of holding the whole new system. Another problematic factor might be the contact points, since, because of the way that the device is being held, it may damage the parts of the PCB in direct contact with the supports, also limiting the airflow through those parts of the PCB included in the ECG monitoring system.

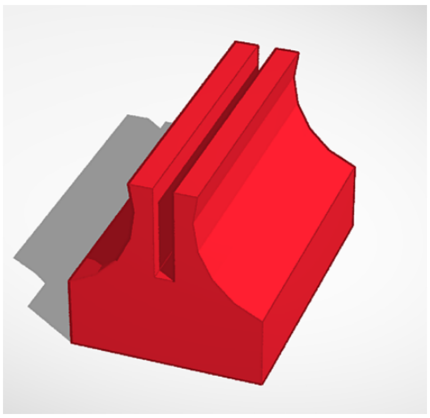


Figure 4.3: Isometric view of the 3D design of the support.

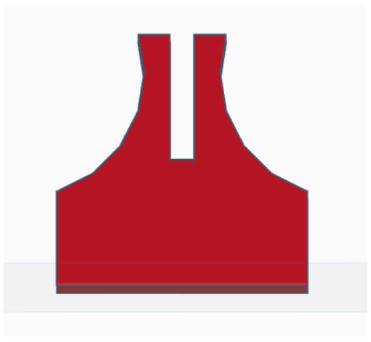


Figure 4.4: Frontal view of the 3D design of the support.

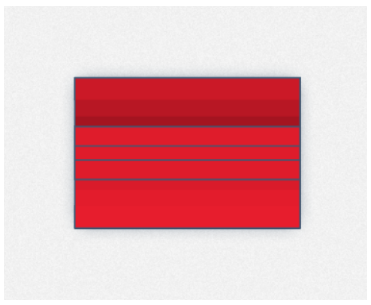


Figure 4.5: Top view of the 3D design of the support.

Figure 4.6: Lateral view of the 3D design of the support.

The next 3D model designed, is radically different from the first one as shown in Fig. 4.7. This design consists of a box in which the PCB, the AD8232 sensor and the Arduino board will be accommodated. This has an indentation in the upper part in which the user can slide in and out a cover also included in the design, the goal of this cover is to protect the fragile components that form the ECG monitoring system. This design also has two apertures on both sides of the container. The function of one of the openings is to connect the Arduino board to the user's computer, while the other one is used to connect the sensor to the electrodes that will be attached to the body.

This design would provide an enclosure that gives physical protection to the PCB, the sensor and the Arduino board, acting as a shield from dirt, dust, or even mechanical damage caused by the accidental dropping of the device. The openings designed at the ends of the box grant an efficient cable management, allowing the user to have an organized cable routing, making it simpler to connect the cables to their respective destination. Furthermore, an enclosed design usually gives a more professional appearance to the final product.

On the other hand, heat accumulation might be a problem in this model, while the container offers protection, it may retain the heat inside the box, which could result in overheating of the electronic components of the device due to the reduced ventilation. Another drawback of this design is the restricted accessibility, the apertures only allow the user to connect or disconnect the cables, but if some issue occurs in the connection of the elements among their selves the cover would have to be removed, which could be an inconvenient. In this box there may be size constraints, the container has a specific size ruled by the dimensions of the other elements, which could complicate the incorporation of extra elements in future improvements of this device.

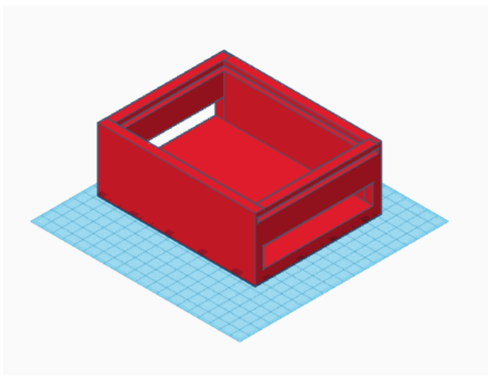


Figure 4.7: Isometric view of the 3D design of the box.

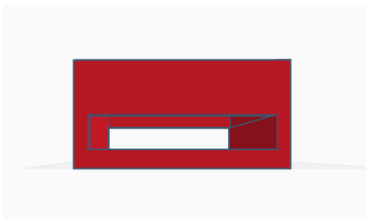


Figure 4.8: Frontal view of the 3D design of the box.

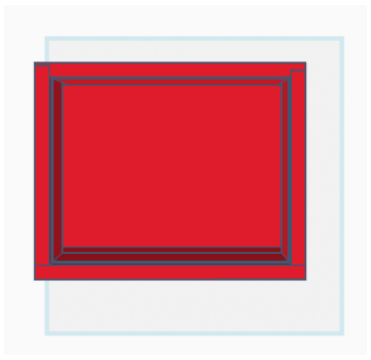


Figure 4.9: Top view of the 3D design of the box.

Figure 4.10: Lateral view of the 3D design of the box.

Material used for 3D printing

The selection of an appropriate material for 3D printing the enclosure or the supports of the ECG device is crucial for ensuring its functionality, durability, and overall performance. Among the various materials available for 3D printing, polymers stand out as the most suitable choice for this application. This justification outlines the reasons for selecting a polymer and providing some polymers that are well-suited for 3D printing the enclosure of the ECG monitoring system.

Polymers fulfill all the requirements of this project, making them an ideal choice for the ECG device enclosure and the supports. The key advantages of using polymers include:

- **Durability**. Polymers such as Acrylonitrile Butadiene Styrene (ABS) and Polylactic Acid (PLA) offer great impact resistance and strength, ensuring that the 3D-printed components can handle everyday use and accidental drops.
- **Electrical Insulation**. In general, polymers provide good insulating properties, which are essential for protecting the electronic components within the enclosure from electrical interference and short circuits.

- **Lightweight**. Polymers are intrinsically lightweight, which contributes to the overall portability of the ECG device. This is particularly important for a device intended for regular use and transport.
- **Cost-Effectiveness**. Polymers are generally cheaper than metals and other advanced materials. This makes them a cost-effective option for producing enclosures, especially for small-scale production runs or prototype printing.

4.2 GUI Design

The term GUI is used to refer to a visual representation of data that allows the users to interact with electronic devices or software applications. These representations usually include graphical components such as icons, buttons or drop-down menus to facilitate user interaction and comprehension, presenting the information in an intuitive manner.

In this project, the main goal of the GUI is to act as a bridge between the ECG monitoring device and the user. This application allows the users to visualize the ECG data in real-time, interpret it and record it. Thanks to the GUI, the user is able to control the monitoring process, customize the settings and, access information related to the ECG readings.

The functioning of this application is very simple. First of all, the user has to launch the application executable, and a window like the one shown in Fig. 4.13 appears. Then the user has to connect the device to their computer's serial port via USB. The user has to connect the three electrodes to the body after plugging it in the sensor. The GUI has a drop-down menu which includes an intuitive and graphic description of where and how to attach the electrodes.

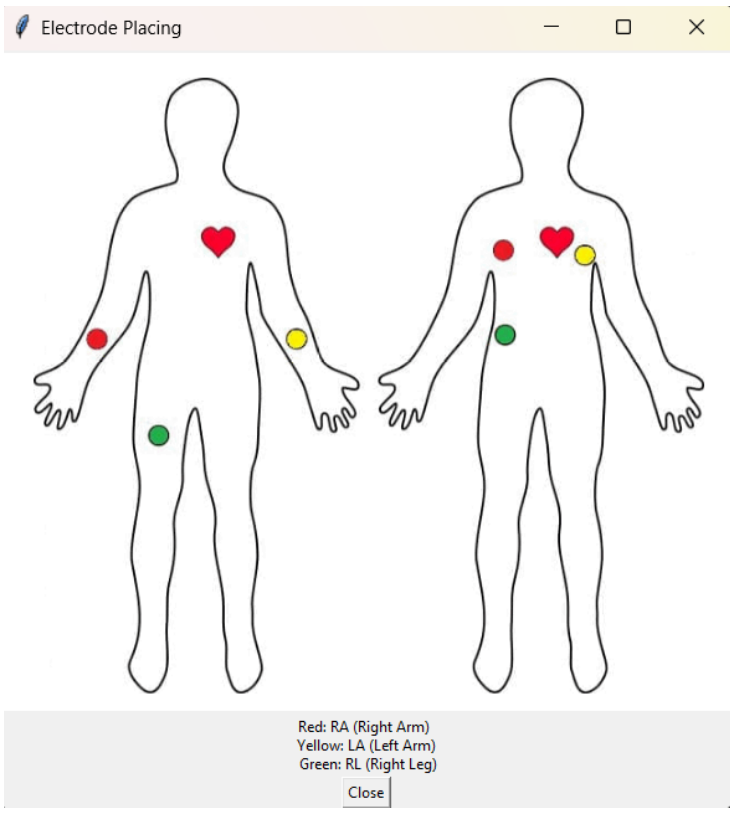


Figure 4.11: Window of the GUI that shows how to place electrodes.

After connecting everything, the user must specify the serial port to which the device is connected. The application includes a section that provides a step-by-step guide for identifying the port used by the ECG device. By following these steps correctly, the user can start the serial connection, and the ECG data will be displayed in real time on the interface.

The GUI shows three seconds of detected data at a time. After this period, the graph is cleared, and the next three seconds of data are displayed. With a sampling rate of 500 samples per second, the user will see 1500 samples in the application. The interface also allows the user to record the displayed ECG signal by clicking a button, with the option to stop the recording whenever necessary. Upon stopping the recording, a window will prompt the user to save the file to a specific location on the computer. The recording will be saved in CSV format.

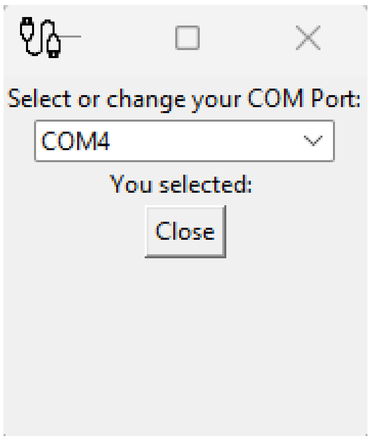


Figure 4.12: Window of the GUI in which the user selects the COM port.

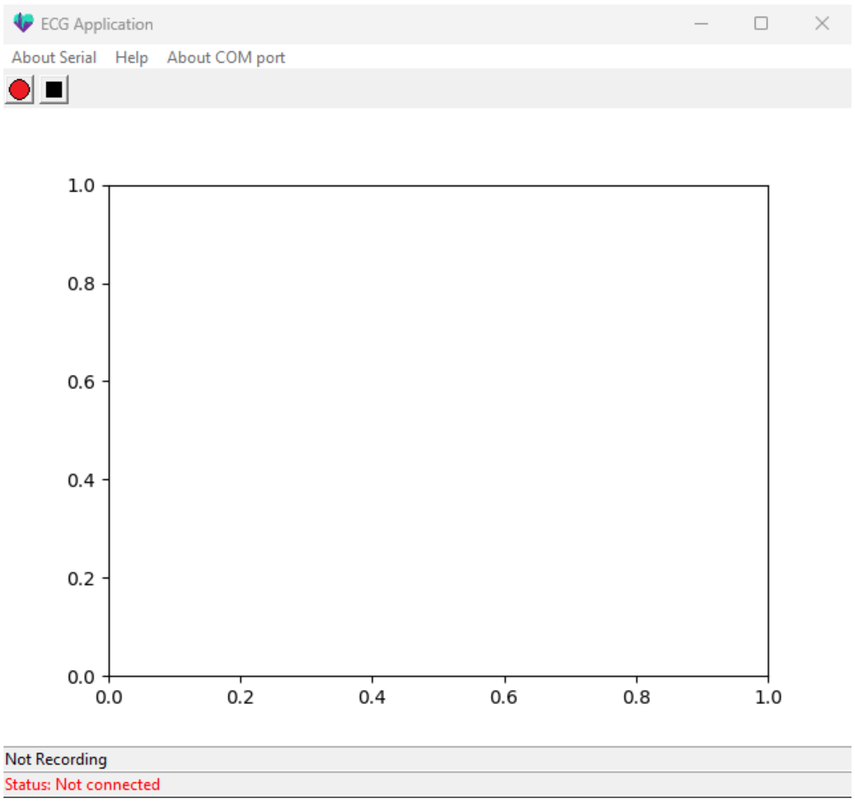


Figure 4.13: Initial view of the GUI.

4.2.1 Python Code

The structure of the code of the GUI is divided into three different files. The first one is named 'Application', this is a short program in which the main GUI window is created and the two other programs are called. Apart from that program, there is a file named 'SerialManager' which contains all the functions related to the serial connection between the ECG system and the computer ensuring an effective connection between the two devices. The last file is the one called 'GUI-ECG', this program is in charge of the creation of all the widgets that make up the GUI, as well as all the functions to filter the registered ECG signal and the ones that allow the user to record the signal. These last two programs will be explained in detail in the following section.

Serial Connection: Serial Manager

All the functions of this part of the code have been grouped in a class named 'SerialManager', this has been made in order to organize the functions used in this part of the project. Besides that, it is also very useful to gather the functions in a class taking into account the possibility of future improvements added to the project, since the developer would only need to extend this class without modifying the rest of the code.

The SerialManager class has been created to manage the serial connection made between the computer and the Arduino board, and therefore the AD8232 sensor. The purpose of this part of the code is to create and stabilize the communication with the ECG device, read the signal information transmitted by the sensor through the serial port, implement a moving average filter to the received data, and create the function that manages the start and stop of data recording.

- **Init Function.**

  This function is mandatory in every class, it is used to initialize multiple instance variables. Besides that, in this case, it also specifies the filter size for the moving average filter.

- **Read from port.**

  This method is in charge of reading data from the serial port repeatedly as long as the serial port is opened. It will convert the received information into a float if it is possible (if it is not possible it prints a warning message) and it will implement the moving average filter to the transformed values, finally, it will append this number to a list for it to be graphically represented in the GUI. If the record button has been pushed the data will be included in another list as well as in the list mentioned before.

- **Moving average.**

This function computes the moving average of the ECG data. It eliminates the most obsolete value from the list where the data is added, it appends the latest value, and calculates the average.

- **Start serial.**

  It is used to start the serial communication between the devices. It retrieves the port that the user selected in the GUI, closes the serial port in case it is already opened and establishes the serial connection taking into account the selected COM port and the specified baud rate. It also initiates an independent thread to read data from the serial port repeatedly.

- **Close serial.**

  This function is created to terminate the serial communication.

- **Start recording.**

  This is used to begin the recording of the registered data. If the record button is pressed, the list in which the recorded data is appended is cleaned and the recording begins.

- **Is recording.**

  This function is just used to see the status of the recording, which is used in the GUI.

- **Stop recording.**

  This is the last function of this class. It ends the recording process.

GUI description: ECGApp

On the other hand, the ECGApp encompasses all the functions related to the creation of the GUI and its widgets along with the signal processing functions. The GUI eases the serial communication with the ECG monitoring device, it applies different filters in order to clean the noise of the signal and saves the file that is generated when the user activates the recording function. The main module used to create the User Interface is Tkinter. This class also integrates the SerialManager class explained before for handling serial communication.

- **Init Function.**

  This function is common in all the classes. Apart from the initiation of the variables, in this case, the main window logo is loaded as well as the SerialManager class.

- **Create widgets.**

  This method creates and configures the drop-down menus in which the user can get advice on how to connect the electrodes or find the COM port. This function also configures the dynamic labels to indicate if the serial connection is successfully done and if the signal is being recorded.

- **Combobox window.**

  The following function creates an emerging window in which the user inserts the COM port in which the ECG monitoring system is connected, this selection can be modified as many times as the user needs.

- **Setup animation.**

  This is in charge of creating the canvas or plotting area in which the ECG signal will be represented in real-time, it is created using the Matplotlib module. This function also starts an independent thread that is used to update the plot periodically with the new ECG data registered.

- **Butter lowpass.**

  This function creates the Butterworth low pass filter that is later applied to the ECG. It has an order and a cutoff frequency which are established by the developer. The function returns the filter coefficients.

- **Butter highpass.**

  Similar to the last explained method, this function designs the Butterworth high pass filter, in this case, the developer has to specify the cutoff frequency and order of the filter. The returned values are also the filter coefficients.

- **Apply filters.**

  The filtering coefficients created with the above-explained functions are used in this method, which applies the Butterworth high pass filter and the Butterworth low pass filter along with the Notch filter to remove the power line interference present in all the ECG signals and other external noise. This function returns the data detected through the sensor filtered.

- **Calculate BPM.**

This function applies the filters to clean the signal, then it detects the R-peaks and calculates the Beats per Minute (BPM) based on the time intervals between those R-peaks.

- **Update plot thread.**

  This starts an independent thread and repeatedly updates the plot with the detected ECG data. Inside of this method, the user specifies the sample rate in which the signal is plotted as well as the number of seconds that are represented in the GUI at the same time. Apart from that, it uses the 'apply-filters' method along with a baseline noise reduction to the detected data before plotting it on the interface.

- **Update BPM.**

  This method retrieves the 'Calculate BPM' method explained before to compute the beats per minute and updates this number in the GUI. It is programmed to be updated every second.

- **Animate.**

  This function is in charge of clearing the old ECG representation which is shown in the plotting canvas and re-plots the newest data.

- **Show combobox and hide combobox.**

  These two functions configure the emerging window in which the user will specify the serial port that is being used.

- **Check COM.**

  this creates another window containing the instructions on how to check the COM port used in the device manager.

- **Electrodes.**

  this function has the purpose of showing the user how to attach the electrodes to the body.

- **order.**

  this will create a message box explaining step-by-step the instructions on how to switch on the ECG device and start using it.

- **Start recording.**

  the following function calls the 'stop-recording' method from the SerialManager class. It also disables the button that starts the recording and enables the stop recording button, by doing this the user will not be able to start another recording before stopping the current one.

- **Stop recording.**

  This method does the opposite of the last explained function, it will enable the button to start the recording and disables the stop recording button, while also calling the 'stop-recording' method from the SerialManager, but also calls the 'save-recorded-data' function that will allow the user to save the recorded data in the desired location.

- **Save recorded data.**

  This is in charge of saving the recorded data in a CSV file. It allows the user to save the recording in the desired PC location.

- **Process recording.**

  This last function will apply the Savitsky-Golay filter to smooth the data one last time and ensure that there is the minimum amount of noise in the ECG signal.

Chapter 5

Experiments and Results

The various experiments and tests conducted to evaluate the performance and accuracy of the developed ECG monitoring system yielded valuable insights into its effectiveness and potential future applications. This device consists of the Arduino Mega 2560, the AD8232 sensor, the PCB, and a 3D-designed enclosure. The primary objective of these experiments was to thoroughly assess the device's capabilities and performance in key areas such as signal acquisition, data processing, data recording, visualization, and the usability of both the ECG device and the GUI.

Given the importance of reliable and accurate ECG monitoring in healthcare, it was crucial to ensure that the designed device consistently provides high-quality ECG signals. Therefore, the experiments conducted were designed to test the system under various conditions to validate its performance. These tests aimed to determine the efficacy of the implemented filtering algorithms for noise reduction and the user-friendliness of the interface developed for real-time monitoring and data recording.

Additionally, the experiments evaluated the GUI in terms of user experience, specifically its ability to display real-time ECG waves, handle ECG data, and guide inexperienced users through the setup process.

Apart from the practical applications, the ECG system was designed taking into account a possible educational use. This device is intended to be used by students of the biomedical engineering degree at the URJC. The real-life experience of assembling, configuring, and using the ECG monitoring system gives the students a deeper understanding of biomedical instrumentation and signal processing. By using and getting familiarized with this device the students can close the gap between theoretical knowledge and practical application, enhancing their learning experience and preparing them for future professional roles in cardiac and biomedical fields in general.

This section details the results of these experiments, evaluating the overall device performance.

5.1 Device Performance

5.1.1 PCB test

The development of the ECG monitoring system included designing and manufacturing a custom PCB to seamlessly integrate the Arduino Mega 2560 board and the AD8232 sensor. The goal was to create a robust and reliable PCB to enhance the device's overall functionality. However, the tests performed on the device revealed a critical design flaw that significantly impacted the test results.

The initial PCB design aimed to optimize the layout of the ECG device components, minimizing residual interference and ensuring efficient routing. The AD8232 sensor was strategically positioned in the center of the PCB, with the connection paths carefully drawn. However, during testing, no signal was detected at the output, and the AD8232 was not functioning, as indicated by its inactive ON LED.

To diagnose the issue, a thorough inspection and analysis of the device were conducted. The problem was identified as an unintended short circuit between adjacent pins of the AD8232, caused by the close placement of the connection paths. This was verified using a multimeter, which confirmed the short circuit observed during the visual inspection.

After the identification of the flaw, the redesign of the PCB was promptly started, resulting in the design shown in Fig. 5.2. Even though the design is very similar, the focus was placed on rerouting the path that was causing the short circuit in the AD8232 sensor's pins. A re-evaluation of the entire layout was performed in order to ensure that the rest of the routes were optimally drawn to avoid future short circuits. This new design fulfils the best practices in PCB design, focusing on clear and unobstructed signal routes but taking into account efficiency and signal integrity.

The improved PCB design is expected to solve the issues that were encountered with the initial version, allowing the AD8232 sensor to function as it should. Nevertheless, due to timeline constraints, it was not possible to manufacture the redesigned PCB within the deadlines of the project. However, the implementation of the upgraded design will be deferred to future phases of this project.

In conclusion, the first PCB design for the ECG monitoring device encountered a critical bug that affected the functionality of the whole device. A meticulous redesign was performed to address this issue. This experience highlights the importance of iterative design and testing in engineering projects.

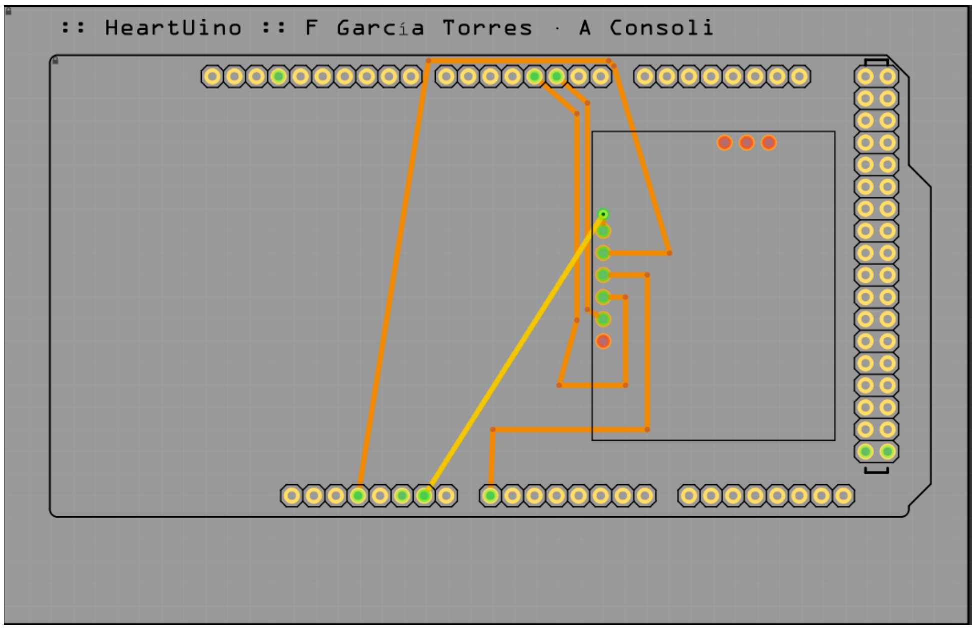


Figure 5.1: Updated design of the PCB.

5.1.2 External interferences

In the design and testing of the ECG monitoring system, it was clear that the quality of the ECG signals captured was affected by different sources of interference. A remarkable source of such interference was the computer used for data processing while it was connected for charging. In this subsection, a discussion will be made about the impact of this and other possible sources of interference that could deteriorate the ECG signal, along with multiple advice for minimizing the noise and ensuring an accurate signal acquisition [44].

When the computer used for interfacing the ECG monitoring system was charging, a notable amount of noise was observed in the ECG signal. This interference can be seen as a distortion of the ECG waveform, making it difficult to perform an accurate interpretation of the electrical activity of the heart. The main reason that causes this noise is the electromagnetic interference that is generated by the computer's Power Supply Unit (PSU) when it is connected to its charging adapter. The PSU converts the Alternating Current (AC) from the power plug into Direct Current (DC) needed by the computer's components. During this conversion from AC to DC, high-frequency noise might be generated. This noise can be propagated through the USB ports of the computer and reach connected devices, such as the ECG monitoring system. The ECG signals are in the microvolt range, meaning that this type of signal is sensitive to noise, which could lead to bad quality of the signal.

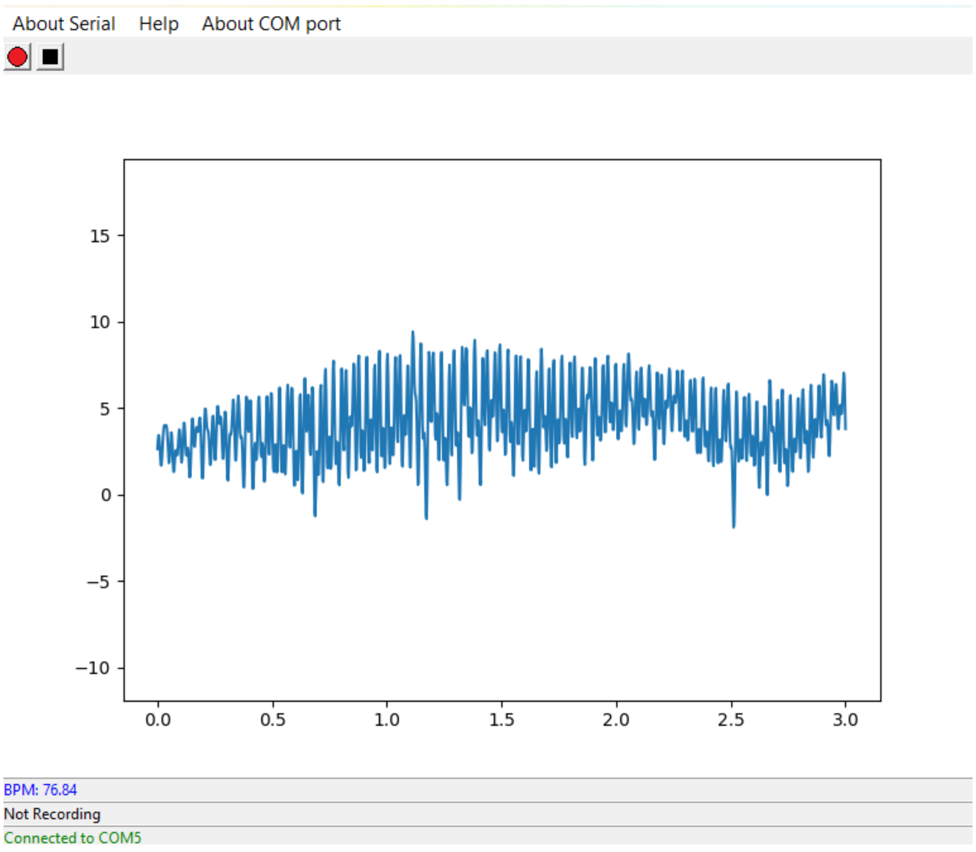
Here are some recommendations to avoid, or at least minimize the noise caused by charging the computer [45].

- **Use Battery Power.** The user should disconnect the charging adapter when the ECG monitoring device is being used, to prevent the PSU from generating electromagnetic interferences.
- **Isolate Power Supplies.** Avoid the use of external devices that require power, such as a monitor. This is essential in order to avoid possible ground loops.
- **Use Quality Power Adapters.** High-quality, well-shielded power adapters can reduce the amount of EMI generated.

Nevertheless, the charging adapter is not the only possible source of electromagnetic noise that can deteriorate the ECG signal. Here is a brief description of other possible noise sources [46].

- **Electromagnetic Interference from Electronic Devices.** Electronic devices such as mobile phones or tablets, can generate EMI. It is recommended to keep this kind of devices away from the ECG monitoring system while it is being used.
- **Electrode Placement and Quality.** Low-quality electrodes or inefficient electrode placement can introduce noise. Using high-quality electrodes and placing the electrodes correctly in the patient's body can decrease the presence of noise in the ECG signal.
- **Motion Artifacts.** The movement of the patient or the electrodes attached to the body can originate noise known as motion artifacts. By keeping the patient still and securing the electrodes the signal quality can notably increase.

The presence of noise and interference in ECG signals can significantly hinder the accurate monitoring and interpretation of cardiac activity. This section highlighted the specific issue of interference caused by using a computer while it is connected to its charging adapter, explaining the underlying causes and providing practical recommendations to mitigate this issue. Additionally, other common sources of interference were discussed, along with strategies to minimize their impact. Adhering to these best practices is crucial for obtaining high-quality ECG signals, thereby ensuring the reliability and effectiveness of the ECG monitoring system.



BPM: 76.84
Not Recording
Connected to COM5

Figure 5.2: ECG trace recorded when the PC was connected to the charger.

5.2 Measurements

The interpretation of ECGs provides decisive insights into the heart's electrical activity, serving as a crucial diagnostic tool in the clinical context. In this section, we present an analysis of ECG measurements registered from a healthy person, both at rest and after physical activity. These measurements help to understand the characteristic patterns and responses of a healthy heart, facilitating an efficient understanding of cardiac physiology.

5.2.1 Rest test

Fig. 5.3 represents a 3 second-long segment of the ECG trace acquired after 20 minutes of rest of the patient under study. The patient was lying on a bed, performing no activity for 20 minutes, and then electrodes were placed on his body and a 1 minute-long trace was acquired. The ECG waveform of a healthy individual at rest is normally characterized by a unique pattern which represents the depolarization and repolarization of the cardiac muscle fibers during each heartbeat [47].

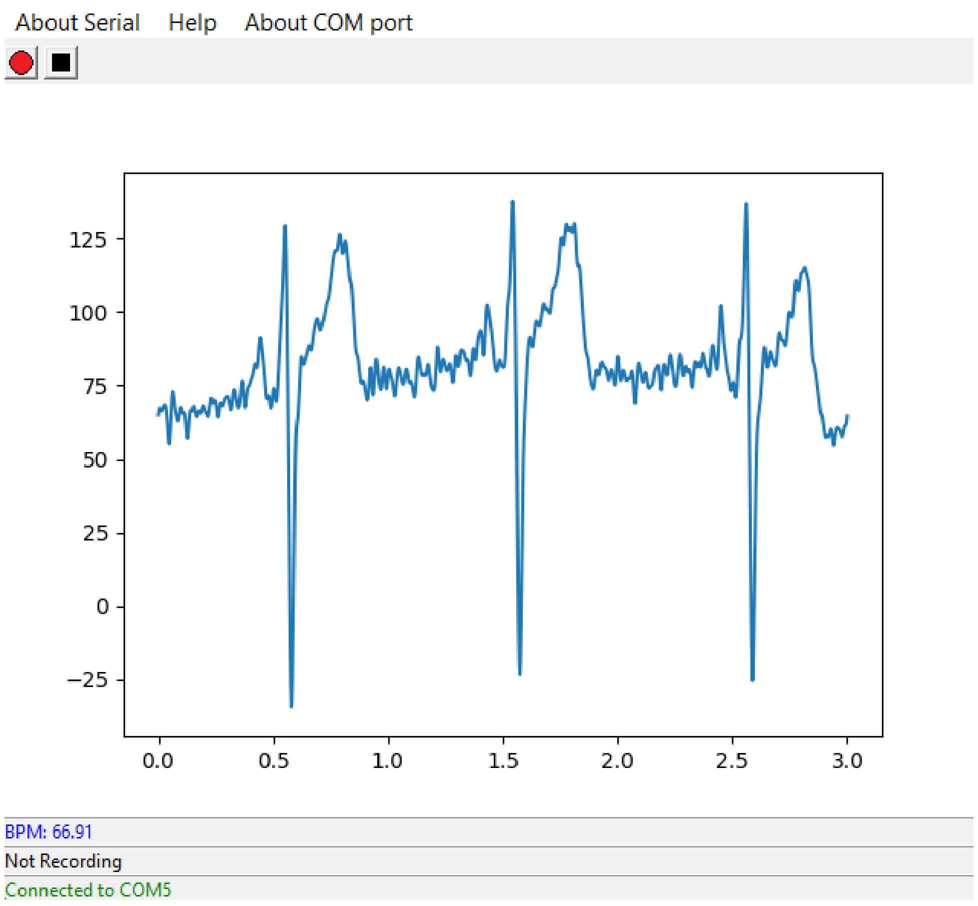
At rest, the ECG of a healthy patient shows a constant rhythm characterized by the sequence of multiple waves and intervals. The waveform itself starts with the P wave, which is the representation of the atrial depolarization as the electrical impulse spreads through the atria, which stimulates them to contract. Right after the P wave, there is the PR interval, which symbolizes the delay in the conduction at the atrioventricular node, which allows the coordinated contraction of the ventricles after the atria did so.

Afterward, the QRS complex takes place on the ECG, grouping the Q, R and S waves. The QRS complex represents ventricular depolarization, showing the fast spread of the electrical impulse through the bundle branches and Purkinje fibers, which ends up triggering ventricular contraction. The length and morphology of this complex provide very important information about the efficiency and normal operation of the ventricular conduction system and myocardial contraction [47].

The ST segment appears after the QRS complex, and it represents the early stages of ventricular repolarization. This part of the ECG wave, in ideal an scenario, should not show any important deviations from the baseline, which is a reliable indicator of normal cardiac function. If this is accomplished, this segment is said to be isoelectric. The last part of the ECG waveform is the T wave, which represents ventricular repolarization as the myocardium and all the components involved in the cardiac cycle prepare for the next one [47].

The heart rate is measured in beats per minute, which is, in a healthy person at rest, comprised between 60 and 100 [48]. From the trace shown in Fig. 5.3, the heart rate is calculated to be 67 BPM. This heart rate is a consequence of the balance between sympathetic and parasympathetic action on the heart's pacemaker cells, assuring reliable circulation and oxygen delivery throughout the body.

A correct interpretation of the ECG of a healthy individual includes an evaluation of the consistency and morphology of these ECG waveforms and the intervals that form them. A healthy patient's ECG waveform is characterized by a regular rhythm and morphology, which is interpreted as a normal cardiac electrical activity and operation. Variations in any of these aspects may indicate the presence of underlying cardiac pathology, requiring further medical evaluation.



BPM: 66.91
Not Recording
Connected to COM5

Figure 5.3: ECG of a healthy individual in the designed GUI.

5.2.2 Stress test

After physical exercise, the ECG of a healthy patient shows remarkable differences compared to the resting state. These changes are mainly characterized by an increased heart rate and enhanced sympathetic nervous system activity. Figure 5.4 represents a 3 second-long segment of the ECG trace acquired after an interval of short but intense physical activity done by the patient under study, then the electrodes were attached to his body and a 1 minute long trace was acquired.

Following physical activity, the BPMs usually rise notably in order to accommodate the metabolic demands of the body. In a healthy person, the heart rate can range from 120 to 180 BPM or higher approximately, this heart rate will vary depending on the duration and intensity of the physical activity. From the trace shown in Fig. 5.4, the heart rate is calculated to be 155 BPM. This elevation of the cardiac rhythm, named tachycardia, manifests the adaptive mechanism of the heart, which is able to increase the cardiac output, ensuring proper oxygen delivery to the tissues.

Here is a description of the different parts of the ECG waveform after physical activity, which presents certain anomalies concerning the ECG of a healthy resting individual.

- **P Wave.** The P wave represents the atrial depolarization. In a post-exercise ECG waveform, the P wave might appear more often and slightly steeper because of the increased atrial rate. In general terms, the morphology of the P wave is very similar in comparison with the resting state ECG, which indicates proper atrial function.

- **PR Interval.** This interval represents the conduction time from the atria to the ventricles through the atrioventricular node. The PR interval usually is reduced with exercise. This shortening denotes an increased sympathetic tone and a decreased vagal tone. This enhances a faster conduction necessary to fulfil the increased circulatory demand.

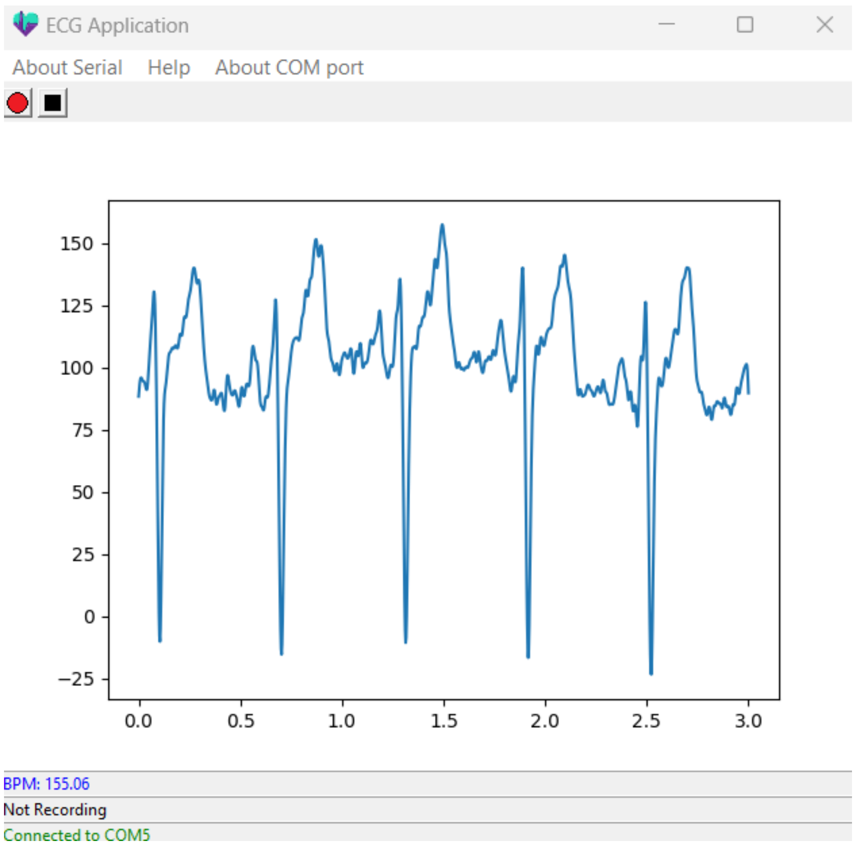
- **QRS Complex.** This complex represents ventricular depolarization. In a post-exercise ECG, the QRS complex might be slightly narrower in comparison with the QRS complex of a resting state ECG waveform. This difference is caused by the augmented conduction velocity within the ventricles during physical activity. Nevertheless, the morphology of this complex remains regular, which reflects an efficient ventricular function.

- **ST Segment.** This segment of the ECG waveform reflects the early phase of ventricular repolarization. In healthy patients, the ST segment should stay very similar in exercise and resting state. ST segment deviations could indicate myocardial ischemia.

- **T Wave.** This wave represents ventricular repolarization. T wave might become more prominent after exercise, this is caused by the enhanced repolarization currents and the elevated heart rate. In healthy patients, T wave should remain almost identical in both situations.

- **QT Interval.** This interval includes the total total duration of the depolarization and repolarization. In post-exercise ECG, the QT interval tends to shorten due to the increased heart rate. This adaptation assures the completion of the cardiac cycle, enabling rapid and consecutive contractions.

The analysis of a post-exercise ECG requires a detailed assessment of the waveforms and intervals to differentiate between normal physiological adaptions from potential pathological anomalies. A regular rhythm with correct waveform morphology and interval adjustments reflects a healthy cardiac function. Variations from the expected patterns might indicate exercise-inducted deviations or latent cardiovascular diseases, which may require further clinical evaluation.



BPM: 155.06
Not Recording
Connected to COM5

Figure 5.4: ECG of a healthy individual after physical exercise.

Chapter 6

Conclusions

The design and implementation of the ECG monitoring system detailed in this project represent significant progress in making cardiac monitoring more accessible and educationally useful. The main objectives were to create an economical and user-friendly device for initial cardiac diagnostics and educational purposes. These objectives have been successfully met, though some challenges and areas for future improvement remain.

In the healthcare industry, ECG machines are often designed with a focus on innovation and reliability, sometimes at the expense of user-friendliness and affordability. While traditional ECG machines are highly accurate and reliable, they are often expensive and inaccessible to a broad population due to the need for specialized medical training. This project aims to address these issues by creating an open-source, low cost ECG system using readily available components.

This ECG monitoring system aims to provide a practical tool for healthcare students to enhance their learning experiences, as well as a user-friendly preliminary diagnostic tool. The combination of affordability, intuitive use, and educational value makes this project a notable contribution to both diagnostics and the medical field.

6.1 Summary of achievements

- **Device Development.** The project successfully designed and developed a functional prototype of an ECG monitoring system using an Arduino microcontroller and the AD8232 sensor. The software components were meticulously designed to facilitate user interaction, including features for real-time waveform visualization, signal recording, and user-friendly instructions for novices. The hardware development involved careful integration of the AD8232 sensor and the Arduino board with other components, ensuring reliable signal acquisition and processing. This approach made the ECG device both functional and user-friendly, suitable for educational and diagnostic purposes.
- **Cost-Effective Solution.** The project successfully developed an affordable alternative to traditional ECG machines. By utilizing components like the Arduino Mega 2560 board, which is both cost-effective and widely available, the project achieved its goal of creating a cheap yet robust and reliable device. This affordability makes the device accessible to a wider audience, including schools and educational centers with limited budgets, community health centers, and individuals seeking to monitor their cardiac activity at home.

6.2 Future Scope

- **PCB Design Issues.** The most crucial problem encountered in this project was a flaw in the PCB design. One trace was mistakenly routed too close to the pins of the AD8232 sensor, causing the device to short circuit and malfunction. This design bug was discovered during the testing phase when the sensor failed to provide any data. This incident highlighted the importance of meticulous design verification before PCB manufacturing. The design has been updated to correct this issue by rerouting the defective trace to ensure it does not interfere with the sensor pins. However, due to time constraints, it was not possible to manufacture the revised PCB within the project timeline.
- **Interference from External Power Sources.** Another notable challenge identified was the interference caused when the ECG device was connected to a charging computer. This connection resulted in the sensor detecting noisy, unusable signals. It is recommended that the ECG system not be used while connected to a charging computer. Additionally, other potential sources of EMI, such as WiFi routers or mobile phones, should be avoided. To minimize these effects, future improvements to the device could incorporate shielding and filtering techniques.

- **Educational Integration.** The ECG monitoring device is expected to be used by students of the biomedical engineering degree of the URJC. This will provide a hands-on learning experience, helping the students to comprehend practically cardiac monitoring and signal analysis, shortening the gap between theoretical knowledge and practical application. With this device the students will be able to observe real-time ECG waveforms, understand the correlation between the heart operation and electrical activity, and learn to detect and interpret different cardiac occurrences. This practical exposure is intended to boost their understanding of cardiology, preparing them for clinical practice.

- **Future Improvements.** Further iterations could focus on boosting the device's robustness against interference and incorporating additional features for a more comprehensive ECG monitoring device use. One alternative might be to test different sensor configurations that offer better noise isolation hence improved signal quality. Furthermore, the GUI can be improved in order to facilitate user interaction and data interpretation. A more intuitive and rich GUI can enhance user experience, facilitating the device operation and ECG data analysis. Another likely improvement is the integration of wireless technology, such as WiFi or Bluetooth, allowing remote monitoring and data transmission. Also, the incorporation of more advanced signal processing algorithms can boost the accuracy of the ECG signals by accurately filtering out the noise.

Bibliography

[1] Prieto-Avalos, G., Cruz-Ramos, N. A., Alor-Hernández, G., Sánchez-Cervantes, J. L., Rodríguez-Mazahua, L., Guarneros-Nolasco, L. R. (2022). Wearable devices for physical monitoring of heart: A review. Biosensors, 12(5), 292. https://doi.org/10.3390/bios12050292

[2] NIH: National Institute of Diabetes and Digestive and Kidney Diseases. *Hematuria (Blood in the Urine)* (NIHDDK Website Link)

[3] Patel, J. V., Chambers, C. V. & Gomella, L. G. (2008) *Hematuria: etiology and evaluation for the primary care physician.* The Canadian journal of urology, 15 Suppl 1, 54–62.

[4] AlGhatrif, M., Lindsay, J. (2012). A brief review: history to understand fundamentals of electrocardiography. Journal of Community Hospital Internal Medicine Perspectives, 2(1), 14383. https://doi.org/10.3402/jchimp.v2i1.14383

[5] Waller, A. D. (1887). A demonstration on man of electromotive changes accompanying the heart's beat. The Journal of Physiology, 8(5), 229–234. https://doi.org/10.1113/jphysiol.1887.sp000257

[6] Vincent, R., Department of General Medicine, Amala Institute of Medical Sciences, Thrissur, Kerala, Indi. (2022). From a laboratory to the wearables: a review on history and evolution of electrocardiogram. Iberoamerican Journal of Medicine, 4(4), 248–255. https://doi.org/10.53986/ibjm.2022.0038

[7] Zipes, D. P. (2000). Clinical application of the electrocardiogramEditorials published in the Journal of the American College of Cardiology reflect the views of the authors and do not necessarily represent the views of JACC or the American College of Cardiology. Journal of the American College of Cardiology, 36(6), 1746–1748. https://doi.org/10.1016/s0735-1097(00)00971-2

[8] Rasuli, B., Kuok, Y.-J. (2011). Myocardial infarction. En Radiopaedia.org. Radiopaedia.org.

[9] Sweis, R. N. (s/f). Acute Myocardial Infarction (MI). MSD Manual Professional Edition. https://www.msdmanuals.com/professional/cardiovascular-disorders/coronary-artery-disease/acute-myocardial-infarction-mi

[10] Ranjbar, A., Sohrabi, B., Sadat-Ebrahimi, S.-R., Ghaffari, S., Kazemi, B., Aslanabadi, N., Seyvani, B., Hajizadeh, R. (2021). The association between T wave inversion in leads with ST-elevation and patency of the infarct-related artery. BMC Cardiovascular Disorders, 21(1). https://doi.org/10.1186/s12872-021-01851-8

[11] Rawshani, A., PhD, M. D. (2016, septiembre 3). ECG signs of myocardial infarction: pathological Q-waves pathological R-waves –. Cardiovascular Education; Cardiovascular Medicine. https://ecgwaves.com/topic/ecg-criteria-myocardial-infarction-pathological-q-waves-r-waves/

[12] Fu, D.-G. (2015). Cardiac arrhythmias: Diagnosis, symptoms, and treatments. Cell Biochemistry and Biophysics, 73(2), 291–296. https://doi.org/10.1007/s12013-015-0626-4

[13] Allessie, M. A., Bonke, F. I., Schopman, F. J. (1977). Circus movement in rabbit atrial muscle as a mechanism of tachycardia. III. The “leading circle” concept: a new model of circus movement in cardiac tissue without the involvement of an anatomical obstacle. Circulation Research, 41(1), 9–18. https://doi.org/10.1161/01.res.41.1.9

[14] Manuals, M. S. D. (2017, agosto 9). Overview of atrial fibrillation.

[15] Cascino, T. (s/f). Electrocardiography. MSD Manual Professional Edition. https://www.msdmanuals.com/professional/cardiovascular-disorders/cardiovascular-tests-and-procedures/electrocardiography

[16] Desai, D. S., Hajouli, S. (2023). Arrhythmias. StatPearls Publishing.

[17] Hypertrophic cardiomyopathy (HCM). (s/f). Www.heart.org. https://www.heart.org/en/health-topics/cardiomyopathy/what-is-cardiomyopathy-in-adults/hypertrophic-cardiomyopathy

[18] Enlarged heart (cardiomegaly). (s/f). Cleveland Clinic. https://my.clevelandclinic.org/health/diseases/21490-enlarged-heart-cardiomyopathy

[19] Frey, N., Katus, H. A., Olson, E. N., Hill, J. A. (2004). Hypertrophy of the heart: A new therapeutic target? Circulation, 109(13), 1580–1589. https://doi.org/10.1161/01.cir.0000120390.68287.bb

[20] Hypertrophic cardiomyopathy (HCM). (s/f). Www.heart.org. https://www.heart.org/en/health-topics/cardiomyopathy/what-is-cardiomyopathy-in-adults/hypertrophic-cardiomyopathy

[21] Zhang, T., Liu, N., Xu, J., Liu, Z., Zhou, Y., Yang, Y., Li, S., Huang, Y., Jiang, S. (2023). Flexible electronics for cardiovascular healthcare monitoring. Innovation (Cambridge (Mass.)), 4(5), 100485. https://doi.org/10.1016/j.xinn.2023.100485

[22] Ashley, E. A., Niebauer, J. (2004). Understanding the echocardiogram. REMEDICA.

[23] Cardiac cycle. (n.d.). Utmb.edu. https://www.utmb.edu/Pedi_Ed/CoreV2/cardiology/Cardiology3.html

[24] Sodhro, A., Sangaiah, A., Sodhro, G., Lohano, S., Pirbhulal, S. (2018). An energy-efficient algorithm for wearable electrocardiogram signal processing in ubiquitous healthcare applications. Sensors (Basel, Switzerland), 18(3), 923. https://doi.org/10.3390/s18030923

[25] Features, 1., Description, 3. (s/f). ADS129x low-power, 2-channel, 24-bit analog front-end for biopotential measurements. Www.ti.com. https://www.ti.com/lit/ds/symlink/ads1292.pdf

[26] MAX32620HSP. (n.d.). Mbed.com. https://os.mbed.com/platforms/MAX32620HSP/

[27] ISO 13485. (2020). ISO. https://www.iso.org/iso-13485-medical-devices.html

[28] What is 60601 - all you need to know. (s/f). Compliance Navigator. https://compliancenavigator.bsigroup.com/en/medicaldeviceblog/what-is-60601-all-you-need-to-know/

[29] Us, W. W. (2020, February 12). Product and service. Ente Certificazione Macchine. https://www.entecerma.it/en/activities/certifications/product-and-service/

[30] Guo, S.-L., Han, L.-N., Liu, H.-W., Si, Q.-J., Kong, D.-F., Guo, F.-S. (2016). The future of remote ECG monitoring systems. Journal of Geriatric Cardiology: JGC, 13(6), 528. https://doi.org/10.11909/j.issn.1671-5411.2016.06.015

[31] Exploring the latest breakthroughs advancements in ECG tech. Executive Electrocardiogram Education. https://www.ecgedu.com/latest-advancements-in-ecg-tech/

[32] Mobile ECG devices market size and share report, 2030. (2023). [Data set].

[33] Github.com. https://github.com/khoazero123/arduino-mega-and-esp8266

[34] Arduino.Cc. https://www.arduino.cc/en/Guide/ArduinoMega2560

[35] FUNCTIONAL BLOCK DIAGRAM. (s/f). Data Sheet AD8232. Analog.com. https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf

[36] Bezerra Beniz, Douglas Espíndola, Alexey. (2016). USING TKINTER OF PYTHON TO CREATE GRAPHICAL USER INTERFACE (GUI) FOR SCRIPTS IN LNLS.

[37] matplotlib: Python plotting package. PyPI, https://matplotlib.org.

[38] Tanner, D., Morgan-Short, K., Luck, S. J. (2015). How inappropriate high-pass filters can produce artifactual effects and incorrect conclusions in ERP studies of language and cognition. Psychophysiology, 52(8), 997–1009. https://doi.org/10.1111/psyp.12437

[39] Feng, H., Sun, L. (2024). Design and simulation of Butterworth Low-pass filter. Academic Journal of Science and Technology, 10(3), 106–110. https://doi.org/10.54097/jpnnb366

[40] Kasim, Ö., Tosun, M. (2022). Effective removal of eye-blink artifacts in EEG signals with semantic segmentation. Signal, Image and Video Processing, 16(5), 1289–1295. https://doi.org/10.1007/s11760-021-02080-4

[41] AISLER - Quick and affordable manufacturing for your Electronic Project. (s/f). AISLER - Quick and Affordable Manufacturing for Your Electronic Project. https://aisler.net/users/signin

[42] SparkFun electronics. (s/f). Sparkfun.com. https://www.sparkfun.com/

[43] Fritzing.org. (s/f). https://fritzing.org/

[44] Chatterjee, S., Thakur, R. S., Yadav, R. N., Gupta, L., Raghuvanshi, D. K. (2020). Review of noise removal techniques in ECG signals. IET Signal Processing, 14(9), 569–590. https://doi.org/10.1049/iet-spr.2020.0104

[45] Medical devices: sources of electromagnetic interference. (s/f). Gov.uk. https://www.gov.uk/government/publications/electromagnetic-interference-sources/electromagnetic-interference-sources

[46] Thalkar, S., Upasani, D., Associate Professor. (s/f). Various techniques for removal of power line interference from ECG signal. Ijser.org. https://www.ijser.org/researchpaper/Various-Techniques-for-Removal-of-Power-Line-Interference-From-ECG-Signal.pdf

[47] Rawshani, A., PhD, M. D. (2018, mayo 14). Overview of the ECG waves, deflections, intervals, durations –. Cardiovascular Education; Cardiovascular Medicine. https://ecgwaves.com/overview-of-the-ecg-waves-deflections-intervals-durations/

[48] Jillian Kubala, R. D. (2023, febrero 27). Average heart rate: What it should be and how to measure it. Health. https://www.health.com/average-heart-rate-7106508

[49] Arduino Mega 2560 R3 (A000067). (s/f). PTSolns. https://ptsolns.com/es/products/arduino-mega-r3

[50] SparkFun Heart Rate Monitor - AD8232 - Mikroelectron MikroElectron is an online electronics store in Amman. (s/f). Mikroelectron.com. https://mikroelectron.com/Product/SparkFun-Heart-Rate-Monitor-AD8232/

[51] (S/f-b). Geekymedics.com. https://geekymedics.com/understanding-an-ecg/

[52] Wikipedia contributors. . Heart. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Heartoldid=1233530196

[53] Myocardial infarction. (s/f). Ecgpedia.org. https://en.ecgpedia.org/wiki/Myocardial_Infarction

[54] Sinus arrhythmia. (s/f). Cleveland Clinic. https://my.clevelandclinic.org/health/diseases/21666-sinus-arrhythmia

[55] Tinkercad - From mind to design in minutes. (s/f). Tinkercad. https://www.tinkercad.com/

Additional Material

Arduino code

```

void setup () {
// initialize the serial communication:
Serial .begin(9600);
pinMode(2, INPUT); // Setup for leads off detection LO +
pinMode(3, INPUT); // Setup for leads off detection LO −

}

void loop () {

if (( digitalRead (2) == 1) ||( digitalRead (3) == 1)){
Serial . println ('!');
}
else{
// send the value of analog input 0:
Serial . println (analogRead(A0));
}
//Wait for a bit to keep serial data from saturating
delay (1);
}
```

Python code

Application Code

```

import tkinter as tk
from GUI_ECG import ECGApp
from Serial_Manager import SerialManager

#This function creates the window for the GUI
def main():
    # Initialize the main window
    main_window = tk.Tk()

    app = ECGApp(main_window)

    # Start the Tkinter main loop, waiting for events
    main_window.mainloop()


if __name__ == "__main__":
    main()
```

Serial Manager Code

```

import tkinter as tk
from tkinter import messagebox
import serial
import threading
import logging
import numpy as np

class SerialManager:
    # Class for managing serial communication and data processing

    def __init__(self, filter_size=5):
        # Initialize the SerialManager object with default values

        # Serial communication variables
        self.ser = None                      # Serial object
        self.ser_open_port = False           # Indicates if serial port is open
        self.sel_port = tk.StringVar()       # Selected serial port
        self.filter_size = filter_size       # Size of the moving average filter
        self.bpm = 0                         # Heart rate (beats per minute)

        # Data variables
        self.ser_data_ecg = [0] * 150        # Buffer for ECG data
        self.ser_data_ecg_recorded = []      # Recorded ECG data
        self.rec = False                     # Flag for recording mode

        # GUI elements for connection and recording status
        self.con_text = None                 # Connection status text
        self.con_label = None                # Connection status label
        self.rec_text = None                 # Recording status text
        self.rec_label = None                # Recording status label

    def read_from_port(self):
        # Function to read data from the serial port

        try:
            while self.ser_open_port:
                reading = self.ser.readline().strip()  # Read line from serial
port
                try:
                    reading_value = float(reading)  # Convert reading to float
                    # Apply moving average filter to the data
                    filtered_value = self.moving_average(reading_value)
                    self.ser_data_ecg.append(int(filtered_value))  # Append
filtered data to buffer
                    if self.rec:
                        self.ser_data_ecg_recorded.append(filtered_value)  #
Append data if recording
```

```
                    except ValueError:
                        logging.warning("Invalid value received: %s", reading)  # Log warning for invalid data

            except serial.SerialException as e:
                logging.exception("SerialException occurred: %s", e)  # Log exception for serial errors
                messagebox.showinfo("Error", "The port is not correct")  # Show error message
                self.con_text.set("Error: check if the port is correct")  # Update connection status text
                self.con_label.config(fg="red")  # Set connection status label color to red

    def moving_average(self, new_value):
        # Function to compute moving average of ECG data

        self.ser_data_ecg.pop(0)                 # Remove oldest value from buffer
        self.ser_data_ecg.append(new_value)      # Add new value to buffer
        # Calculate and return the moving average
        return sum(self.ser_data_ecg) / len(self.ser_data_ecg)

    def start_serial(self):
        # Function to start serial communication

        try:
            s = self.sel_port.get()  # Get selected serial port from GUI
            if self.ser and self.ser.is_open:
                self.ser.close()  # Close serial port if already open
            self.ser = serial.Serial(s, 9600, timeout=20)  # Open serial port
            self.ser_open_port = True  # Set flag to indicate port is open
            thread = threading.Thread(target=self.read_from_port)  # Start thread to read from port
            thread.start()
            self.con_text.set("Connected to " + s)  # Update connection status text
            self.con_label.config(fg="green")  # Set connection status label
```

```
color to green

        except serial.SerialException as e:
            logging.exception("SerialException occurred: %s", e)  # Exception for
serial errors
            messagebox.showinfo("Error", "Wrong port?")  # Show error message
            self.con_text.set("Error: wrong port?")  # Update connection status
text in the GUI
            self.con_label.config(fg="red")  # Set connection status label color
to red

    def close_serial(self):
        # Function to close the serial communication

        self.ser_open_port = False  # Set flag to indicate port is closed
        if self.ser and self.ser.is_open:
            self.ser.close()  # Close serial port if open
        self.con_text.set("Not connected")  # Update connection status text in
the GUI
        self.con_label.config(fg="red")  # Set connection status label color to
red in the GUI

    def start_recording(self):
        # Function to start recording ECG data

        self.rec = True  # Set recording flag to True
        self.ser_data_ecg_recorded.clear()  # Clear recorded data list

    def stop_recording(self):
        # Function to stop recording ECG data

        self.rec = False  # Set recording flag to False

    def is_recording(self):
        # Function to check if recording is active

        return self.rec  # Return recording flag status
```

ECG GUI Code

```

import tkinter as tk
from tkinter import ttk, messagebox, Toplevel, Label, Button, filedialog,
 StringVar, PhotoImage, SUNKEN
from tkinter.constants import *
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import peakutils
import threading
import time
import logging
import csv
from scipy.signal import butter, filtfilt, iirnotch
from Serial_Manager import SerialManager # Assuming Serial_Manager is a custom
module
import os
import scipy

# Define the main application class
class ECGApp:
    serial_manager: SerialManager # Type hint for better IDE support

    # Initialize the class and declare variables
    def __init__(self, GUI):
        self.GUI = GUI
        self.GUI.title("ECG Application") # Set window title
        self.logo = PhotoImage(file='logo.png') # Load application logo
        self.GUI.iconphoto(False, self.logo) # Set window icon
        self.serial_manager = SerialManager() # Initialize SerialManager instance

        self.create_widgets() # Create GUI widgets (labels, buttons, etc.)
        self.setup_animation() # Set up real-time plot animation
        self.port_combobox = None # Initialize COM port selection combobox

        # Start the periodic BPM update
        self.update_bpm()

    # Function to create GUI widgets
    def create_widgets(self):
        # Initialize and set default values for labels displaying connection status, recording status, and BPM
        self.serial_manager.con_text = tk.StringVar()
        self.serial_manager.con_text.set("Status: Not connected")

        self.serial_manager.rec_text = tk.StringVar()
        self.serial_manager.rec_text.set("Not Recording")

        self.serial_manager.bpm_text = tk.StringVar()
        self.serial_manager.bpm_text.set("BPM: --")

        # Create labels to display connection status, recording status, and BPM
        self.serial_manager.con_label = tk.Label(self.GUI, textvariable=self.serial_manager.con_text, fg="red", bd=1,
                                                 relief=SUNKEN, anchor=W)
        self.serial_manager.con_label.pack(side=BOTTOM, fill=X)

        self.serial_manager.rec_label = tk.Label(self.GUI, textvariable=self.serial_manager.rec_text, fg="black", bd=1,
                                                 relief=SUNKEN, anchor=W)
        self.serial_manager.rec_label.pack(side=BOTTOM, fill=X)

        self.serial_manager.bpm_label = tk.Label(self.GUI, textvariable=self.serial_manager.bpm_text, fg="blue", bd=1,
                                                 relief=SUNKEN, anchor=W)
        self.serial_manager.bpm_label.pack(side=BOTTOM, fill=X)

        # Menu bar configuration
        menu = tk.Menu(self.GUI)
        self.GUI.config(menu=menu)

        # Serial menu dropdown
        SerialMenu = tk.Menu(menu)
        menu.add_cascade(label='About Serial', menu=SerialMenu)
        SerialMenu.add_command(label='Start Serial ...', command=self.serial_manager.start_serial)
        SerialMenu.add_command(label='Stop Serial ...', command=self.serial_manager.close_serial)

        # Help menu dropdown
        instructions_menu = tk.Menu(menu)
        menu.add_cascade(label='Help', menu=instructions_menu)
        instructions_menu.add_command(label='How to put the electrodes', command=self.electrodes)
        instructions_menu.add_command(label='How to start', command=self.order)

        # COM port menu dropdown
        com_menu = tk.Menu(menu)
        menu.add_cascade(label='About COM port', menu=com_menu)
        com_menu.add_command(label='How to know my COM', command=self.check_COM)
        com_menu.add_command(label='Change my COM', command=self.combobox_window)

        # Toolbar for recording-related buttons
        toolbar = tk.Frame(self.GUI)

        # Load icons for record and stop buttons
        rec_path = 'circulo.png'
        self.icon_rec = PhotoImage(file=rec_path)

        stop_rec_path = 'cuadrado.png'
        self.icon_rec_stop = PhotoImage(file=stop_rec_path)

        # Create record and stop buttons and pack them in the toolbar
        self.record_button = tk.Button(toolbar, image=self.icon_rec, command=self.start_recording)
        self.record_button.pack(side=tk.LEFT, padx=2, pady=2)

        self.stop_record_button = tk.Button(toolbar, image=self.icon_rec_stop,
    command=self.stop_recording)
        self.stop_record_button.pack(side=tk.LEFT, padx=2, pady=3)

        toolbar.pack(side=tk.TOP, fill=tk.X)

    # Function to handle COM port selection window
    def combobox_window(self):
        com_win = Toplevel(self.GUI)
        com_win.title('Select Your COM port')
        com_win.geometry('180x180')
        com_logo = PhotoImage(file='com_icon.gif')
        com_win.iconphoto(False, com_logo)

        # Label and combobox for COM port selection
        port_label = tk.Label(com_win, text="Select or change your COM Port:")
        port_label.pack()

        self.port_combobox = ttk.Combobox(com_win,
    textvariable=self.serial_manager.sel_port,  # COM Selection
                                          values=[f"COM{i}" for i in range(19)],
     state='readonly')
        self.port_combobox.pack()

        selected_port = self.port_combobox.get()
        combo_label = Label(com_win, text=f"You selected: {selected_port}")
        combo_label.pack()

        close_button = Button(com_win, text="Close", command=com_win.destroy)
        close_button.pack()

    # Function to set up Matplotlib plot for real-time animation
    def setup_animation(self):
        fig, ax = plt.subplots()
        self.ax = ax
        canvas = FigureCanvasTkAgg(fig, master=self.GUI)
        self.canvas = canvas  # Store canvas reference
        canvas_widget = canvas.get_tk_widget()
        canvas_widget.pack(fill="both", expand=True)

        # Start a separate thread to update the plot periodically
        self.update_thread = threading.Thread(target=self.update_plot_thread)
        self.update_thread.daemon = True  # Set as daemon so it terminates when
    the main thread terminates
        self.update_thread.start()

    # Function to create a Butterworth low-pass filter
    def butter_lowpass(self, cutoff, fs, order=6):
        nyquist = 0.5 * fs
        normal_cutoff = cutoff / nyquist
        b, a = butter(order, normal_cutoff, btype='low', analog=False)
        return b, a

    # Function to create a Butterworth high-pass filter
    def butter_highpass(self, cutoff, fs, order=6):
        nyquist = 0.5 * fs
        normal_cutoff = cutoff / nyquist
        b, a = butter(order, normal_cutoff, btype='high', analog=False)
        return b, a

    # Apply the defined filters (low-pass, high-pass, notch)
    def apply_filters(self, data, fs):
        # Low-pass filter
        low_cutoff = 40.0  # Hz
        b, a = self.butter_lowpass(low_cutoff, fs)
        data = filtfilt(b, a, data)

        # High-pass filter
        high_cutoff = 0.5  # Hz
        b, a = self.butter_highpass(high_cutoff, fs)
        data = filtfilt(b, a, data)

        # Notch filter (for 50 Hz power line interference)
        notch_freq = 50.0  # Hz
        quality_factor = 30.0
        b, a = iirnotch(notch_freq, quality_factor, fs)
        data = filtfilt(b, a, data)

        return data

    # Function to calculate BPM (Beats Per Minute)
    def calculate_bpm(self, data, fs):
        # Apply filters
        filtered_data = self.apply_filters(data, fs)

        # Detect R-peaks in the ECG signal
        indexes = peakutils.indexes(filtered_data, thres=0.5, min_dist=int(0.6 *
fs))

        # Calculate BPM
        rr_intervals = np.diff(indexes) / fs  # in seconds
        bpm = 60.0 / np.mean(rr_intervals)

        return bpm

    # Function to update the plot in a separate thread
    def update_plot_thread(self):
        fs = 500  # Sample rate (Hz)
        display_seconds = 3  # Number of seconds to display
        display_samples = display_seconds * fs  # Number of samples to display

        while True:
            if self.serial_manager.ser_open_port:
                if len(self.serial_manager.ser_data_ecg) > display_samples:
                    self.serial_manager.ser_data_ecg =
self.serial_manager.ser_data_ecg[-display_samples:]

                data = self.serial_manager.ser_data_ecg.copy()

                # Apply filters
                filtered_data = self.apply_filters(data, fs)

                try:
                    # Baseline noise reduction
                    baseline = peakutils.baseline(filtered_data, 2)
                    processed_data = filtered_data - baseline
                except ZeroDivisionError:
                    logging.warning("ZeroDivisionError: baseline calculation encountered zero norm.")
                    processed_data = filtered_data  # Fall back to using filtered data without baseline correction

                self.ax.clear()
                self.ax.plot(np.linspace(0, display_seconds, num=len(data)), processed_data)

                # Set y-axis limits based on processed_data characteristics
                y_min = np.min(processed_data) - 10
                y_max = np.max(processed_data) + 10
                self.ax.set_ylim(y_min, y_max)

                self.canvas.draw()

            time.sleep(0.2)

    # Function to update BPM label periodically
    def update_bpm(self):
        fs = 500
        if self.serial_manager.is_recording:
            data = self.serial_manager.ser_data_ecg

            # Calculate BPM
            bpm = self.calculate_bpm(data, fs)

            # Update the BPM label
            self.serial_manager.bpm_text.set(f"BPM: {bpm:.2f}")

            # Program the next BPM update
            self.GUI.after(1000, self.update_bpm)

    # Function for cleaning the graph when reaching 150 values in the list
    def animate(self, i: int) -> None:
        if len(self.serial_manager.serial_data) > 150:
            self.serial_manager.serial_data =
self.serial_manager.serial_data[-150:]
        data = self.serial_manager.serial_data.copy()
        data = data[-150:]
        self.ax.clear()
        self.ax.plot(np.linspace(0, len(data)-1, dtype='int', num=len(data)),
data)

    # Function to hide the COM port selection combobox
    def hide_combobox(self):
        self.port_combobox.pack_forget()

    # Function to show the COM port selection combobox
    def show_combobox(self):
        self.port_combobox.pack()

    # Function to show electrode placing instructions
    def electrodes(self):
        window_elec = tk.Toplevel()
        window_elec.title("Electrode Placing")

        # Load and display image of electrodes
        elec_path = "electrodes.gif"
        elec_img = Image.open(elec_path)
        elec_img = ImageTk.PhotoImage(elec_img)

        image_label = tk.Label(window_elec, image=elec_img)
        image_label.image = elec_img  # To prevent image from being garbage
collected
        image_label.pack()

        # Display text instructions
        text = "Red: RA (Right Arm) \n Yellow: LA (Left Arm) \n Green: RL (Right Leg)"
        text_label = tk.Label(window_elec, text=text)
        text_label.pack()

        # Close button
        close_button = tk.Button(window_elec, text="Close", command=window_elec.destroy)
        close_button.pack()

    # Function to show 'How to start' instructions
    def order(self):
        messagebox.showinfo('How to turn on the ECG',
                            '1. Use the dropdown menu to select the COM port in which the device is connected (If you do not how to do this check it) \n 2. Start the Serial Communication \n 3. Start recording if required \n 4. Stop recording')

    # Function to show instructions on how to check COM port
    def check_COM(self):
        window_com = tk.Toplevel()
        window_com.title("Check your COM port")

        # Load and display image of COM port
        com_path = "com_port.gif"
        com_img = Image.open(com_path)
        com_img = ImageTk.PhotoImage(com_img)

        im_label = tk.Label(window_com, image=com_img)
        im_label.image = com_img # To prevent image from being garbage collected
        im_label.pack()

        # Display text instructions
        text_com = "1. Open the Device Manager \n 2. Open 'Ports (COM & LPT)' \n 3. Check where is the ARDUINO MEGA2560 connected, that is your port"
        text_label = tk.Label(window_com, text=text_com)
        text_label.pack()

    # Function to start recording
    def start_recording(self):
        # Start recording
        self.serial_manager.start_recording()
        self.record_button.config(state=tk.DISABLED)
        self.stop_record_button.config(state=tk.NORMAL)

    # Function to stop recording
    def stop_recording(self):
        # Stop recording
        self.serial_manager.stop_recording()
        self.record_button.config(state=tk.NORMAL)
        self.stop_record_button.config(state=tk.DISABLED)
        self.save_recorded_data()

    # Function to save recorded data to a CSV file
    def save_recorded_data(self):
        if self.serial_manager.ser_data_ecg_recorded:
            filename = filedialog.asksaveasfilename(defaultextension=".csv",
                                                    filetypes=(("CSV files",
"*.csv"), ("All files", "*.*")))
            if filename:
                with open(filename, "w", newline="") as csvfile:
                    csvwriter = csv.writer(csvfile)
                    csvwriter.writerow(["Index", "Value"])
                    for i, value in
enumerate(self.serial_manager.ser_data_ecg_recorded):
                        csvwriter.writerow([i, value])
                messagebox.showinfo("Info", "Data saved successfully.")
        else:
            messagebox.showwarning("Warning", "No recorded data to save.")

    # Function for signal processing and saving cleaned data
    def process_recording(self, data):
        z = scipy.signal.savgol_filter(data, 11, 3)  # Savitzky-Golay filter
(provisional)
```

```
        data_smoothed = np.asarray(z, dtype=np.float32)
        a = len(data)
        baseline = peakutils.baseline(data_smoothed, 2)  # Baseline noise reduction (Mandatory)

        y = data_smoothed - baseline

        # Ask user for save directory
        directory = filedialog.asksaveasfilename(
            defaultextension=".csv",
            filetypes=(("CSV File", "*.csv"), ("All Files", "*.*")),
        )

        # If the user cancels the dialog, return
        if directory is None:
            return

        # Create a file path for the saved file
        file_path = os.path.join(directory, 'output.csv')

        # Save the cleaned signal to a CSV file
        with open(file_path, 'w', newline='') as csvfile:
            csvwriter = csv.writer(csvfile)
            csvwriter.writerow(['Index', 'Value'])
            for i in range(a):
                csvwriter.writerow([i, y[i]])
```