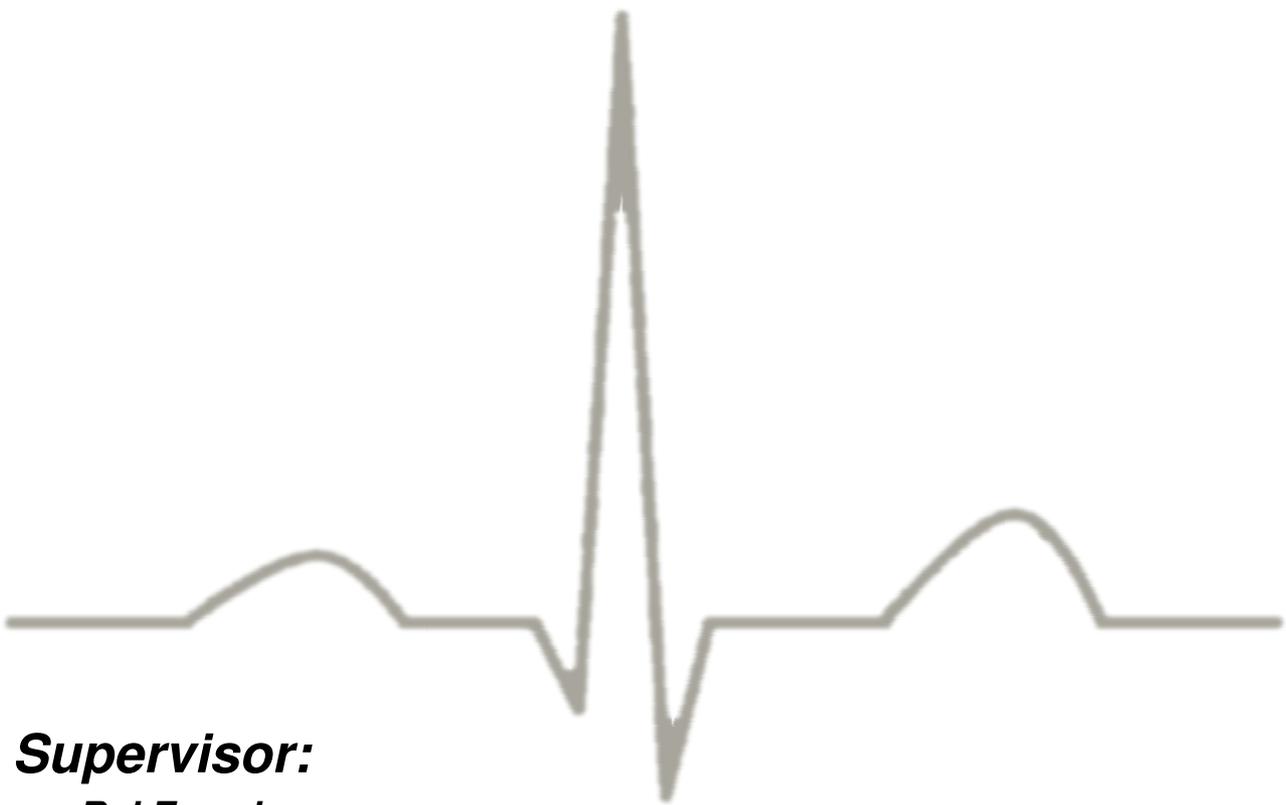




Bruno Miguel
Ferreira dos
Santos

BIOS: BIOLOGICAL INPUT-OUTPUT SYSTEMS



Supervisor:

Rui Escadas

***External
teacher:***

Farid Melgani



Bruno Miguel
Ferreira dos
Santos

BIOS: BIOLOGICAL INPUT-OUTPUT SYSTEMS

Thesis submitted to the University of Aveiro for compliance with requirements to obtain the degree of Master in Electronics and Telecommunications Engineering, held under the scientific guidance of Dr. Farid Melgani, Assistant Professor of Telecommunications, Department of Information Engineering and Computer Science, University of Trento, and Dr. Ernesto Ventura Fernando Martins, Assistant Professor, Department of Electronics, Telecommunications and Informatics, University of Aveiro.

Jury

President

Prof. Dr. António Luís Jesus Teixeira
Professor Associado do Departamento Electrónica,
Telecomunicações e Informática da Universidade de
Aveiro.

Prof. Dr. José Miguel Costa Dias Pereira
Professor Coordenador do Departamento de
Sistemas e Informática da Escola Superior de
Tecnologia de Setúbal.

Prof. Dr. Rui Manuel Escadas Ramos Martins
Professor Auxiliar do Departamento Electrónica,
Telecomunicações e Informática da Universidade de
Aveiro.

Acknowledgments

I would like to give special thanks to Professor Farid Melgani, who helped me throughout this year. Professor Melgani helped me with the project and also with my integration in the university and in Italy. Without him, my stay in Trento along with this project would not have been possible. I would also like to thank Professor Fernando Pianegiani who worked with me during the whole project, and also facilitated my work at the University of Trento.

I would also like to thank Professor Rui Escadas, who was my link to the University of Aveiro. He made the connection process as smooth as possible and also provided me with help during the making of this project.

This wouldn't be complete without recognizing the support given by my parents and my sister, not only in this project but also during the last five years, providing me with all the help I needed in order to be successful in this course.

Thanks to all my friends and also to the new ones I met in Italy, for all their help in this project. They helped me when I needed and I want to thank them especially for all the moral support.

Again, I would like to recognize the enormous amount of work my sister had reviewing the grammar in this paper.

Palavras-chave

Arritmia, atraso, detector, diferenciador, electrocardiograma, fibrilação auricular, filtro, integrador, janela média móvel, limite de decisão, ritmo cardíaco, sensibilidade, sinal ECG, preditivo positivo.

resumo

Este projecto visa a produção de um aparelho portátil para monitorização/análise da actividade cardíaca.

Recebendo como entrada o sinal cardíaco (ECG - electrocardiograma), o dispositivo detecta a característica do sinal e, com base nessa característica, averigua se ocorreu alguma anomalia, produzindo um sinal de alarme se tal acontecer.

O aparelho em causa tem que ser acessível, ter um baixo consumo energético, e a detecção de anomalias tem de ser efectuada em tempo real. Para que esta ultima condição se verificasse, o software implementado requer o mínimo de computação possível, de forma a efectuar todas as operações de detecção e análise num prazo limite de tempo, mantendo-se um detector fiável.

O detector foi optimizado para uma frequência de entrada de 200Hz, e implementado em computador pessoal e num DSP (Digital Signal Processor – processador de sinal digital) desenvolvido pela Microchip®. Ambas as implementações foram testadas recorrendo a sinais cardíacos fornecidos pela MIT-BIH Arrhythmia Database. Os testes desenvolvidos têm o propósito de testar a “qualidade” do detector e possíveis limitações do microprocessador usado, particularmente, no que diz respeito à capacidade de executar a detecção de anomalias cardíacas em tempo real.

Keywords

Arrhythmia, atrial fibrillation, delay, detector, differentiator, ECG signal, Electrocardiogram, filter, heart-rate, integrator, moving-window, positive predictive, sensitivity, threshold.

Abstract

This project intends to produce a portable device for monitoring/analysis of cardiac activity.

Receiving the cardiac signal (ECG – electrocardiogram) as input, the device detects the signal's characteristics and, based upon those characteristics, checks whether some anomaly has occurred, producing an alarm signal in such case.

The aforementioned device has to be affordable, low-power consumption, and the detection of anomalies has to be done in real-time. For this last condition to be verified, the implemented software requires as little computation as possible, so as to carry out all of the detection and analysis operations in a specified amount of time, while maintaining the characteristics of a reliable detector.

The detector has been optimized for an input sampling frequency of 200Hz, and it has been implemented in a personal computer and in a DSP (Digital Signal Processor), developed by Microchip®. Both implementations have been tested resorting to cardiac signals supplied by the MIT-BIH Arrhythmia Database. The tests that have been developed are designed to verify the quality of the detector, as well as possible limitations of the microprocessor used, particularly, when it comes to the ability to carry out the detection of cardiac anomalies in real-time.

Table of Contents

Chapter 1 – INTRODUCTION..... 23

 1.1 - Heart Monitoring23

 1.2 - ECG Signal23

 1.3 - Remote Monitoring.....24

 1.4 - Objective25

 1.5 – Document Organization25

Chapter 2 – ECG 27

 2.1 Heart Activity27

 2.1.1 The Heart in the Circulatory System27

 2.1.1.1 – Systemic Circulation28

 2.1.1.2 – Pulmonary Circulation28

 2.1.2 – Rate28

 2.1.3 – Inside the Heart.....29

 2.2- Heart diseases30

 2.3 – ECG Signal Characteristics.....31

 2.4 ECG measurement34

 2.4.1 Electrodes34

 2.4.2 The 12-lead ECG System36

 2.4.2.1 Limb leads (bipolar)37

 2.4.2.2 Augmented Limb Leads
(Unipolar)38

 2.4.2.3 The precordial/chest leads.....40

 2.5 ECG Device: Holter Monitor41

Chapter 3 - ECG SIGNAL PROCESSING AND ANALYSIS..... 43

3.1 QRS detection algorithm	43
3.1.1 Band-pass filter	44
3.1.2 Differentiator	46
3.1.3 Absolute value	47
3.1.4 Moving-window integrator	47
3.1.5 Detection rules: Adaptive threshold and decision	47
3.2 Signal analysis	50
3.2.1 Heart-rate	50
3.2.2 Atrial fibrillation detection	50
Chapter 4 - ONBOARD IMPLEMENTATION.....	51
4.1 FLEX Full Base Board.....	51
4.1.1 Daughter boards	52
4.1.2 Microchip dsPIC® DSC microcontroller dsPIC33FJ256MC710	53
4.2 QRS detection implementation.....	54
4.2.1 Detection delay	56
4.2.1.1 Band-pass filter delay.....	57
4.2.1.2 Differentiator delay	58
4.3 Atrial fibrillation detection implementation.....	59
Chapter 5 - EXPERIMENTAL RESULTS	61
5.1 MIT-BIH Arrhythmia Database	61
5.2 Test QRS Detection Algorithm	62
5.2.1 Sensitivity and Positive Predictive.....	62

5.2.1.1 Sensitivity and Positive Predictive Test	63
5.2.2 Performance, Delays and Errors.....	66
5.2 Test Atrial Fibrillation Algorithm.....	74
6 Chapter 6 – CONCLUSIONS.....	79
6.1 Result Interpretation.....	79
6.2 Future Developments.....	82
7 Appendices	83
Appendix A – qrs.c.....	83
Appendix B – alarmDet.c	93
Appendix C – How to Run the Program on the PC	97
Appendix D – How to Run the Program on the board.....	101
Appendix E – CD.....	105
8 Bibliography	107

List of Figures

Figure 2.1 - Closed Circulatory System. [2]27

Figure 2.2 - Inside Heart Chambers. [6]30

Figure 2.3 - Prevalence of Atrial Fibrillation by Age and Sex. [8]31

Figure 2.4 - Representation of Five Seconds of an ECG Trace. [10]32

Figure 2.5 - Beat Detail With Emphasis on PQRST Waves.....33

Figure 2.6 – Different Possible Shapes of the QRS Complex. [12]34

Figure 2.7 - Wave Generation When the Positive Electrode is in the Middle of the Cell.....35

Figure 2.8 - Electrodes Placement.....37

Figure 2.9 - Representation of Leads I, II and III in an Axial Reference System. [12].....38

Figure 2.10 - Bipolar Limb Leads and Augmented Limb Leads Represented in the Axial Reference System. [12].....40

Figure 2.12 - Six Precordial Leads.....41

Figure 3.1 - QRS Detection Algorithm, Modified Pan-Tompkins Algorithm.44

Figure 3.2 - Frequency Response of the Low-Pass Filter in Magnitude and Unwrapped Phase.....45

Figure 3.3 - Frequency Response of the High-Pass Filter in Magnitude and Unwrapped Phase.....46

Figure 3.4 - Output of Each Stage of the Original Pan-Tompkins Algorithm.....48

Figure 3.5 - Typical Waveform of Time-Averaged Signal during the QRS Complex.....49

Figure 4.1 - FLEX Full Base Board. [25]51

Figure 4.2 - MPLAB ICD 252

Figure 4.3 - FLEX Multibus Base Daughter Board. [25].....53

Figure 4.4 - FLEX Multibus RS232 Module. [25]53

Figure 4.5 - Beat Detection Algorithm Sequence.....54

Figure 4.6- Low-Pass Filter Impulse Response.....57

Figure 4.7- High-Pass Filter Impulse Response.....58

Figure 4.8 - Differentiator Impulse Response.....59

Figure 5.1 - Output of Each Stage of the Implemented Algorithm.67

Figure 5.2 - Detail Analysis of the Record 103, Sampled at 200 Hz.68

Figure 5.3 - Detail Analysis of the Record 103, Sampled at 360 Hz.69

Figure 5.4 – Detail of ECG with Variations on Peak Height.....71

Figure 5.5 – Detail of ECG with Large P Waves.....72

Figure 5.6 - Detail of ECG with Irregular beats.73

Figure 5.7 – 30 seconds of Record 100, sampled at 200Hz.75

Figure 5.8 - 30 seconds of Record 232, sampled at 200Hz.76

Figure 5.9 - 30 seconds of Record 233, sampled at 200Hz.77

List of Tables

Table 2.1 – Tanaka, Monahan and Seals, Formula for Maximum Heart-Rate.	29
Table 5.1 – Sensitivity and Positive Predictive Test for 'square' and 'absolute', sampled at 200 Hz.	63
Table 5.2 – Sensitivity and Positive Predictive Test for 'median_mean', sampled at 200Hz.	64
Table 5.3 – Conclusions based on the previous tests run for the 4 different types of implementations tested, on the 18 records.	65
Table 5.4 - Sensitivity and Positive Predictive Test for 'absolute', sampled at 360 Hz.	65
Table 5.5 - Comparison Between the use of 200Hz and 360Hz for Sample Frequency, based on Sensitivity and Positive Predictive results.	65
Table 5.6 – Analysis of delays represented in Figure 5.2.	69
Table 5.7 – Analysis of delays represented in Figure 5.3.	70
Table 5.8 – Average Delays of Records 104, 108 and 232.	73
Table 5.9 - False QRS Peaks Detected.	73
Table 5.10 – Relative Error of R-R Intervals, for 30 seconds of each Record.	74

Chapter 1 – INTRODUCTION

1.1 - Heart Monitoring

The monitoring and study of vital signals has always been important, whether as a means to avoid future problems or to solve current ones.

The heart is linked to the entire body and it is responsible for pumping blood to all of its parts. Obviously, if the heart isn't working, nothing else is working either, many of the problems that happen in the body can have their genesis in this organ. Therefore, the study of the heart signal is of the highest importance.

In order to monitor the heart, scientists can make use of electrocardiographic studies, echocardiographic studies or x-rays to get a picture. However, if they really want to explore the heart, they have to perform surgery, which is very risky because the heart's pumping action is critical for the patient's survival. If the heart stops pumping, the body cannot survive.

This project will explore electrocardiographic studies as a means to monitor the heart activity.

1.2 - ECG Signal

The heart has its own electrical activity, which can be measured by an array of electrodes placed on the body surface. The electrocardiogram (ECG or EKG) is the recorded tracing of this measurement, thus it can be concluded that the ECG is nothing more than a graphical picture of the heart's electrical activity.

The ECG helps determine the direction of the electrical waves inside the heart as well as any evidence of heart chamber enlargement or whether the muscular walls of the right or left ventricles are thickened (right or left ventricular hypertrophy).

Before being analyzed by a doctor, the ECG signal is subject to a processing phase. The basic task of this phase is to detect each heartbeat. But, getting to this final goal requires particular stages in the processing pipeline to solve some difficulties that the ECG process encounters, such as:

- Irregularities on the patient's heartbeat, leading to a signal with irregular distance between beats or irregular beat forms;
- Possible patient's movements.
- The patient's breathing, that introduces a low-frequency component in the ECG image;

- Electronics noise of the sensor and the device used to produce the ECG image.

Each one of the particular stages that form the processing part will aim to reduce the influence of these factors, so that the image can be analyzed by the doctor (these stages are explained on Chapter 3 – ECG Signal Processing and Analysis).

Heart problems change the electrical signature of the heart and usually, just by analyzing the picture of the heart's electrical activity (ECG), the Doctor is able to decide on the cause of the problem and, in some cases, solve it. Problems like heart attacks, irregularities in the patient's heartbeat, as well as birth defects of the heart, among others, can be determined just by analysing the ECG.

Since it helps detecting a large range of problems, and being a test that causes no pain or side effects, the ECG test is really common and required in many cases, like when diagnosing pain, when the heartbeats produce strange noises, when a patient faces breathing difficulties and also in other cases like in routine exams, because the comparison of ECGs taken at different times helps the diagnosis, and finally, because the ECGs can monitor how the patient is responding to a specific treatment or surgical intervention.

What can be concluded is that the ECG test is of great importance and when taken often by a patient, increases the probability of detecting diseases at an early stage, thus allowing the patient to be treated immediately.

1.3 - Remote Monitoring

It's unbearable for a patient to go so many times to the hospital to take the ECG test, and it would also be impossible for the hospital to make the ECG test available for all patients every day. Even more impossible would be to have a machine like the ones in hospitals in every patient's house. That's why portable machines that detect and store the ECG trace were created.

The existence of these devices has direct consequences in many people's lives, and it also allows a better management of the hospital resources.

The consequences include a reduction in the number of home care providers, which leads to a higher availability of staff in the hospital and also makes the patient feel more autonomous, not having the necessity of having other people around all the time, which is a factor of great importance, especially for elderly people.

Elderly people, an increasing sector of population, sometimes in a handicapped situation, or people under observation after surgery, and people who are required to take the ECG test regularly as a measure of control can also benefit from the use of these devices. With this technology they don't need to go to the hospital each and every time to take the exam, also reducing their spending with in-hospital assistance.

Furthermore there are some diseases like atrial fibrillation which require immediate medical attention, and that can lead to life-threatening strokes if left untreated. So, the sooner the diagnosis is done, the sooner the treatment can begin.

1.4 - Objective

This work is the beginning of a project with the final goal of producing an affordable low-power consumption device for house monitoring, that receives the heart signal as input - ECG -, detects each heartbeat and its components (detailed on Chapter 2 – ECG), applies some specifications previously determined by a doctor in order to check for diseases (main focus in atrial fibrillation), and finally produces an alarm signal in case it finds signs of possible diseases. This device should work in real-time; therefore its software is implemented with as little computation as possible.

In this first step the main goal is to test the efficiency of the QRS detector and the arrhythmia algorithm on a personal computer and after on a DSP to infer about the possible constraints the microprocessor could bring to the work, in terms of real-time detection.

The joined knowledge of doctors and engineers allows the creation of these devices. As said before, the physician has the necessary expertise to diagnose and cure pathologies through the analysis of specific measurements, in this case, through the analysis of the ECG trace and from other measurements that arise from it. What they don't have is the necessary knowledge to create the machine itself. Such knowledge, like electronics, software development and digital signal processing, among others, is required to create these devices. The engineers' mission is to contribute with this kind of knowledge, and since they have no knowledge about the medical part, the doctors are responsible for bringing their knowledge to the project.

1.5 – Document Organization

Chapter 2 introduces some important terminology about the heart – like ECG, electrocardiogram – describing it and giving a general idea about its function and working theory, as well as possible problems it could face.

Chapter 3 gives a first approach to the problem of analyzing the ECG signal. How to detect each beat occurrence and the following analysis to check for arrhythmias.

Chapter 4 explains how to implement the solution described in Chapter 3 for signal analysis, with emphasis on the limitations of the implementation to real-time, such as delays that each step of the implementation brings. In this Chapter an introduction to the device chosen to process the ECG analysis is also made.

Chapter 5 provides the results of the tests made to explore the quality of the solution implemented, such as Sensitivity and Positive Predictive tests, along with the explanation of what each test mean. A description of the dataset used to test the program is made on the beginning of this Chapter.

Chapter 6 concludes about the results obtained on Chapter 5 and the future developments that can be done to improve or to reuse this project.

Chapter 2 – ECG

2.1 Heart Activity

2.1.1 The Heart in the Circulatory System

Average adults have about 5 liters of blood running in their vessels. Its function is to transport oxygen as well as nourishment (from digestion) and hormones (from glands) along the body, disease fighting substances to the tissue and waste to the kidneys.

The absence of blood would cause the body to stop working.

The main function of the heart is to manage the circulatory system, also called the circle of blood.[1]

As the common ground between two different blood circulations, Systemic Circulation and Pulmonary Circulation, the heart is able to exchange blood between both, closing the circle, as shown on Figure 2.1. These two different circulations link the heart to the entire body and to the lungs, respectively.

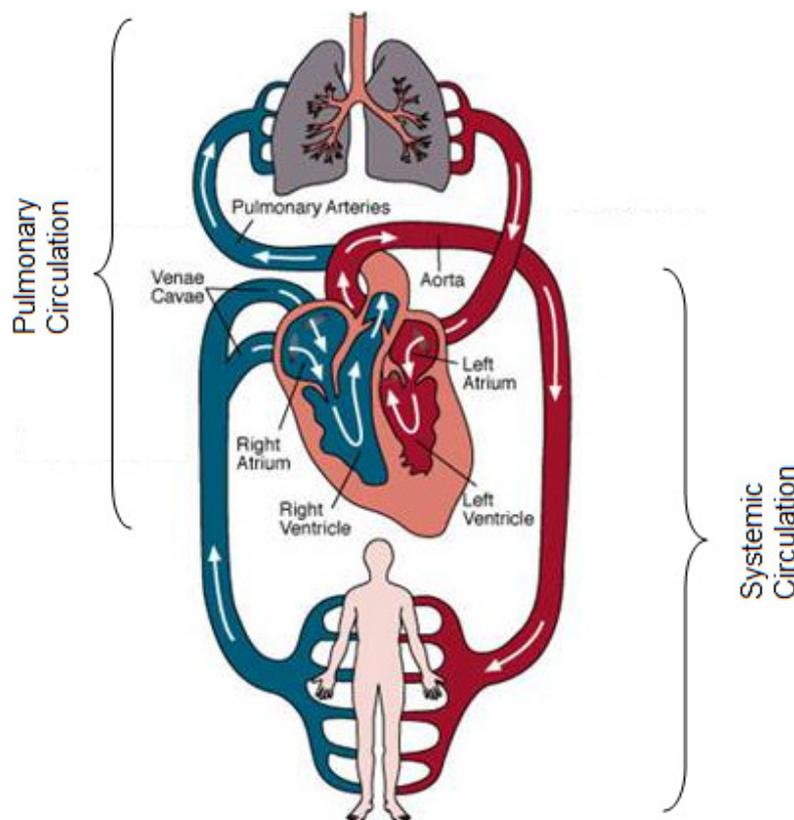


Figure 2.1 - Closed Circulatory System. [2]

The heart also needs nourishment, which occurs through the circulation of blood along the heart tissue, enabled by the presence of capillaries on the heart. This is called the Coronary Circulation.

2.1.1.1 – Systemic Circulation

The goal is to supply nourishment (oxygen and nutrients) to the entire body tissue, with the exception of the heart and lungs, which already have their own systems.

The blood, rich in oxygen, enters the heart through the blood vessels where it is pumped through the heart's main artery, aorta, which branches into other smaller vessels that run throughout the entire body, making the blood arise to the capillaries where oxygen and nutrients are released and the waste collected. Then the blood, rich in waste, flows into the veins that will take it back to the heart, after passing the kidneys, where much of the waste is filtered. [1]

After this process, the heart is full of waste-rich blood, and the pulmonary circulation follows.

2.1.1.2 – Pulmonary Circulation

This phase includes heart and lungs. The purpose of the lungs is to get enough oxygen into the body and to get rid of the waste product of the systemic circulation - carbon dioxide.

The blood rich in carbon dioxide is pumped from the heart to the lungs throughout the arteries, until it reaches the lungs capillaries where the exchange of carbon dioxide for oxygen occurs. Then, the blood with fresh oxygen and free of waste enters the pulmonary veins returning to the heart. After this a systemic circulation will occur. [1]

This will be repeated once in each heartbeat.

2.1.2 – Rate

The heart is the most important muscle in our organism. It is about the same size as a person's fist, and with the body growth, the heart also increases in size.

Like the rest of the body parts, the heart also needs oxygen to grow properly. During childhood there is a greater need for oxygen, causing the heart rate (number of beats per minute) to be faster during this period in life - about 120 beats per minute (bpm). When a child reaches the age of seven, normally the heart rate is of about 90 bpm and by the age of 18 the rate tends to stabilize at 70 bpm. [3]

Therefore, until the age of 18 years old - growing period -, the heart rate is suffering a deceleration. This can also be proved by Tanaka, Monahan and Seals (2001) [4] proposal for calculation of the maximum heart rate (which is adopted nowadays by many institutions), that predicts the formulas present in Table 2.1, that show that with an increase of age the maximum heart rate decreases.

Group	Formula
all patients	$208 - (0.7 * (\text{age in years}))$
sedentary	$212 - (0.7 * (\text{age in years}))$
endurance trained	$205 - (0.6 * (\text{age in years}))$

Table 2.1 – Tanaka, Monahan and Seals, Formula for Maximum Heart-Rate.

However these results are not linear, the heart rate can vary due to many factors. It can be accelerated, for example, in response to:

- Exercise;
- Loud noises;
- Sexual arousal;
- Drugs;
- Mental effort.

In the same way, it is likely to suffer deceleration, for example, during:

- A relaxation period;
- Therapy;
- Medication.

2.1.3 – Inside the Heart

The heart consists of 4 chambers separated by cardiac muscle. The left and right atria are located in the two upper chambers, and the left and right ventricles in the two lower chambers. There are two valves located between the atria (plural of atrium) and the ventricles with the purpose of preventing blood from flowing into the atria when the ventricles contract.

The valve located between the right atrium and the right ventricle is called tricuspid and, as the name implies, it consists of three flexible cusps, which are used to seal the heart valves when closed. On the other side, between the left atrium and the left ventricle, the valve has only two flexible cusps, therefore it is named bicuspid, but it can also be called mitral.

There is also a tissue wall separating the left and right sides, it is called the septum.

The heart has its own electrical activity due to the Sinus or Sinoatrial Node (SA) located on the Right Atrium (Figure 2.2). This node consists of electrical cells sending electrical signals that will cause the blood to be pumped.

Therefore, the SA can be called a “natural pacemaker.”

Each heartbeat is set in motion by an electrical signal sent by the SA, thus controlling the Heart Rate, which becomes the number of electrical signals produced in one minute by the sinoatrial node.

The SA node sends electrical impulses to the atria chambers and the atrioventricular (AV) node, which will cause the four chambers to contract and become empty. The atria are the first to be stimulated followed by the ventricles, which are stimulated by the AV node. [5]

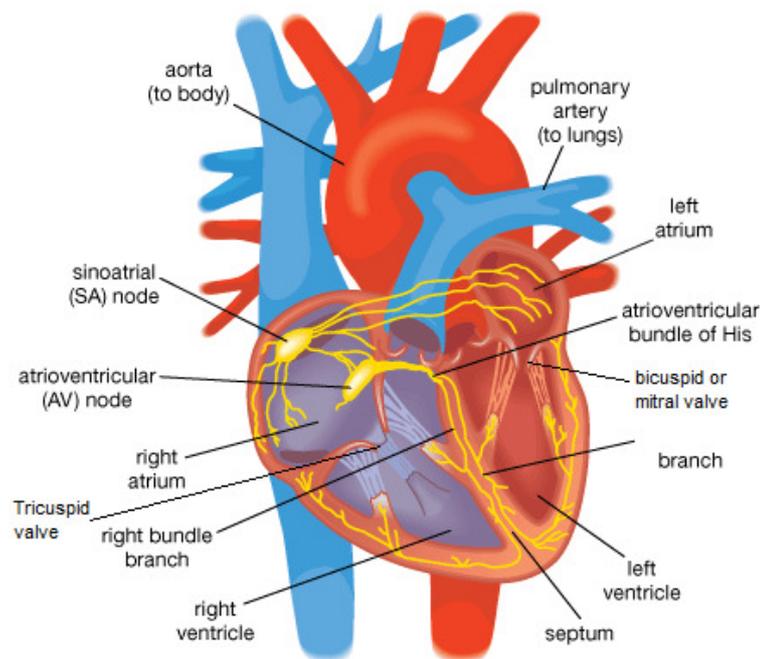


Figure 2.2 - Inside Heart Chambers. [6]

2.2- Heart diseases

By now it can be concluded that the heart is the organ responsible for the vital action of pumping blood to all the tissues of the body. The problem appears when this pumping action is disrupted. These disruptions occur due to diseases affecting the heart. If the heart is unable to pump blood to the body, it may cause damage to some organs.

One common problem is the heart’s irregular beat. The sinoatrial node controls the heartbeat, and if by any reason a disorder affects the electrical signal an irregular beat will occur, those disorders are called Arrhythmias.

Arrhythmias cause the heart to beat too quickly or too slowly or with an irregular pattern. If an acceleration of the heartbeat occurs it is called tachycardia; on the other hand, if a deceleration occurs it is called bradycardia.

Imbalances of blood chemistry, abnormal hormone levels or the use of drugs can be the cause of arrhythmias. [6-7]

The most common type of arrhythmias is Atrial Fibrillation (AF). Fibrillation refers to the rapid and unsynchronized contraction of muscle fibers.

AF prevalence studies (total number of individuals affected by the disease) reveal an increase with age (Figure 2.3). One risk of this disease is that in some cases it is revealed as being asymptomatic, only discovered after an ECG analysis is done.

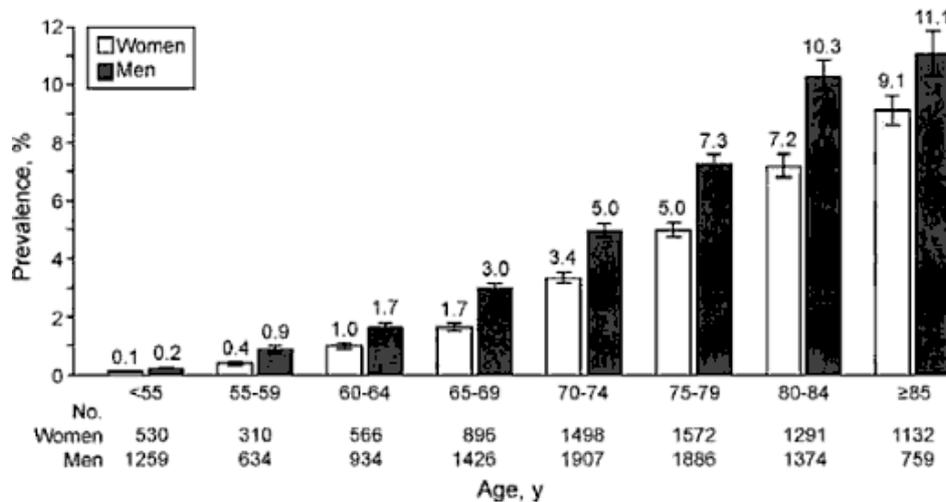


Figure 2.3 - Prevalence of Atrial Fibrillation by Age and Sex. [8]

Like other types of arrhythmias, it is caused by a disorder in the heart’s electrical system. As mentioned above, irregular electrical signals can cause the atria to contract irregularly and faster than they are supposed to (therefore, leading to atrial fibrillation). As a result, the blood isn’t completely pumped to the ventricles; and, as a consequence, the atria and the ventricles will stop working together.

Atrial Fibrillation can result in other diseases either cardiovascular or non cardiovascular. As it results in an increase of the ventricular rate, the irregular rhythm may cause inefficient short cardiac cycles, which can lead to heart attacks or heart failure; loss of physiological control of the heart rate can also occur, lowering the exercise capacity because tachycardia will occur early during exercise; if silent brain infarcts occur after AF, then they can lead, for example, to dementia. [8-9]

2.3 – ECG Signal Characteristics

Each cycle of the cardiac cycle described above has 3 main actions:

- Atrial Systole – the atria contract and pump blood into the ventricles;
- Ventricular Systole – the ventricles contract and pump blood out of the heart;

- Diastole – phase in which the atria and the ventricles relax and begin to fill with blood, it happens after each systole.

The waves and lines represented in a normal ECG (Figure 2.4) are the representation of the electrical events produced by these three main actions. In the following picture is possible to see the existence of a typical pattern for each beat (Figure 2.5).

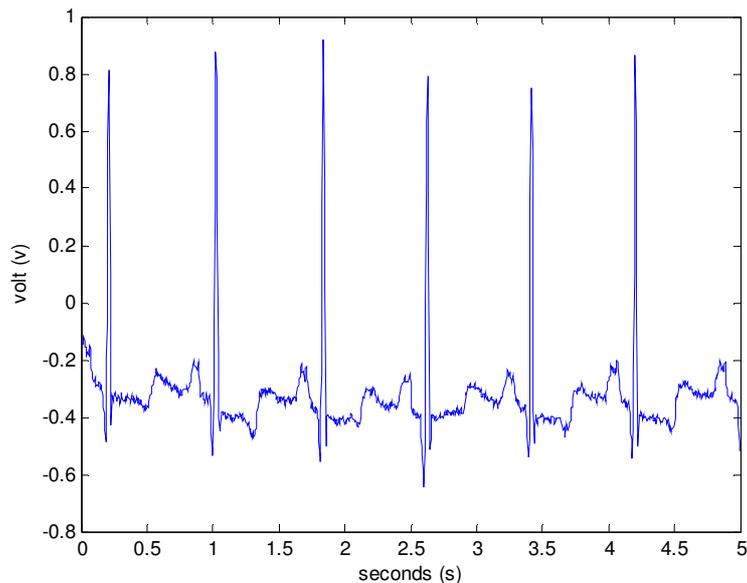


Figure 2.4 - Representation of Five Seconds of an ECG Trace. [10]

- P wave: The P wave represents the atrial depolarization (atrial contraction).

When the right atrium fills with blood, a wave of depolarization begins to spread across the cells of the heart (right and left atria) causing the atria to contract. Because the SA node is located in the right atrium, its depolarization will happen first than in the left atrium, therefore the first part of the P wave, predominantly, represents the right atrium depolarization, and the second part the left atrium depolarization. The atrial contraction will cause the valves between the atria and the ventricles to open, leading the blood to flow from the atria to the ventricles.

- QRS interval: Represents the ventricular depolarization (ventricular contraction).

When the signal is released it moves to the bundle of His, where it is divided in left and right bundle branches which run through the septum (Q wave). This bundle branches terminate in countless Purkinje fibers connected to the cells in the ventricular walls. These fibers are responsible for delivering the electrical current to the ventricles, causing the ventricles to contract. This will

not happen at the same time for both, the left ventricle contracts an instant before the right one does. Left ventricle contraction pushes blood through the aortic valve to the rest of the body (R wave), on the other hand right ventricle contraction pushes blood through the pulmonary valve to the lungs (S wave). Because the left ventricle muscle mass is about three times bigger than the one on the right, the R wave is normally higher than the S wave.

- T wave: Corresponds to the ventricular repolarization.

After depolarization, the ventricles need to restore the electronegativity of their interior so that they can be stimulated again.

Atrial repolarization happens as well, but during the QRS interval, hidden by the much more prominent QRS complex. [11-13]

Figure 2.5 shows a beat detail of the real signal (represented in Figure 2.4) and an approximation to help understand the wave format.

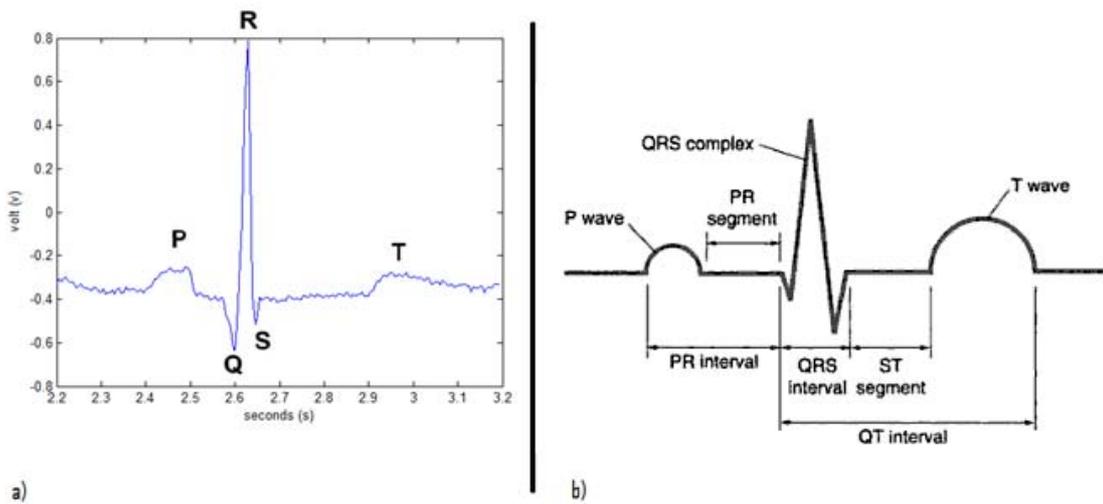


Figure 2.5 - Beat Detail With Emphasis on PQRST Waves.
a) Real Signal b) Approximation. [13]

- PR interval: time from the start of atrial depolarization to the start of ventricular depolarization.
- PR segment: time between the end of atrial depolarization and the start of ventricular depolarization. It corresponds to electrical silence where the Electrical signal arrives to the AV node being then slowed to allow ventricles to fill with blood.

- ST segment: time between the end of ventricular depolarization and the start of ventricular repolarization.
- QT interval: time from the start of ventricular depolarization to the end of ventricular repolarization.
- QRS interval: time of ventricular depolarization.

From the previous it can be concluded that there is a complex (QRS) which corresponds to the ventricular depolarization and that by measuring the time interval between QRS complexes it is possible to find the ventricular rate.

The QRS complex is a very important part of the ECG trace, as the study of its width, amplitude and morphology can help diagnose heart diseases. It does not present the same shape each time; furthermore it may not even present Q, R or S waves.

The variation can be related to the recording electrodes, which can be combined in different ways to present different images of the heart's electrical activity (approached later on this chapter); or it can be related to the abnormal conduction of electrical impulses within the ventricles.

Regardless the waves present and their shape it can always be referred to as QRS complex; however it can also be named according to the wave/waves present (Figure 2.6).

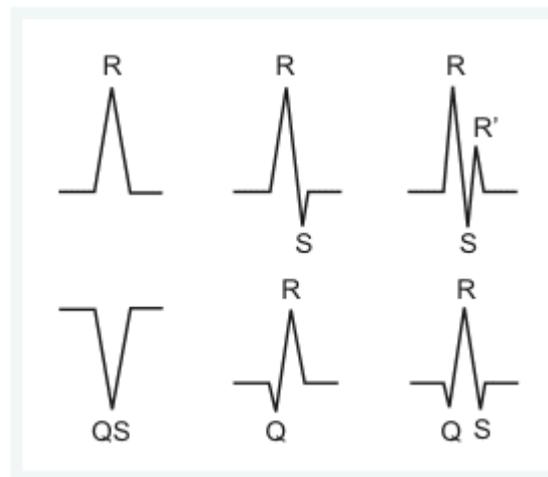


Figure 2.6 – Different Possible Shapes of the QRS Complex. [12]

2.4 ECG measurement

2.4.1 Electrodes

As seen before, the ECG is the measurement of the electrical current generated by the heart during depolarization and repolarization, this measurement is commonly made by an array of electrodes placed on the body.

The electrical image recorded by an electrode varies with the patch's position. A wave recorded by a positive electrode placed on the right arm will be different from one recorded on the left arm or leg. The rules are simple:

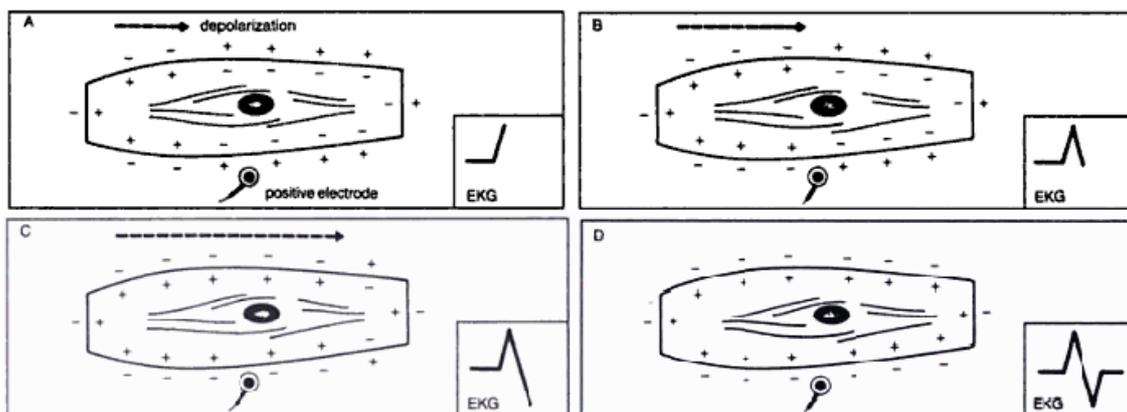
- If depolarization moves towards the positive electrode, it causes a positive deflection on ECG;
- If depolarization moves away from the positive electrode a negative deflection appears on the ECG image.

The opposite can be said about repolarization:

- A wave of repolarization moving towards the positive electrode causes a positive deflection on ECG;
- A wave of repolarization moving away from the positive electrode causes a negative deflection to appear on the ECG image.

And the last rule:

- A wave of depolarization or repolarization traveling perpendicularly to an electrode axis results in a biphasic deflection of equal positive and negative voltages (Figure 2.7). [12-13]



**Figure 2.7 - Wave Generation When the Positive Electrode is in the Middle of the Cell.
A) Wave Moves Towards the Electrode; B) Wavefront Reaches the Electrode; C) Wave Moves Away from Electrode; D) EKG Return to Baseline, hence the Entire Muscle is Depolarized. [13]**

2.4.2 The 12-lead ECG System

Since the heart is 3-dimensional its electrical activity must be considered 3-dimensionally as well. As a result, a pair of electrodes is not enough to study the heart's behavior with regards to its electrical activity.

The 12-lead system is the standard used today to get a 3-dimensional image of the heart's electrical activity, viewing the heart in two planes, frontal and horizontal, with each lead giving details of a certain region of the heart, the combination of all leads allows all sorts of information to be extracted from the ECG, for instance, where pathologic processes are occurring. Normally, when an ECG is recorded, all leads are recorded simultaneously, which gives the name to this system: 12-lead ECG.

In this system 10 electrodes are placed in standard positions of the body and connected to a device that measures potential differences between selected electrodes to produce the characteristic ECG tracings.

Electrodes are usually embedded in the middle of very sticky circles of thick tape-like material, in order to be placed in the human body (Figure 2.8), on the following positions: [13-15]

- Right arm (RA) and left arm (LA), avoiding bony prominences;
- Left leg (LL) and sometimes right leg (RL) used as grounds to minimize the interference, also avoiding bony prominences;
- V1: right 4th intercostal space;
- V2: left 4th intercostal space;
- V3: halfway between V2 and V4;
- V4: left 5th intercostal space, mid-clavicular line;
- V5: horizontal to V4, anterior axillary line;
- V6: horizontal to V5, mid-axillary line.

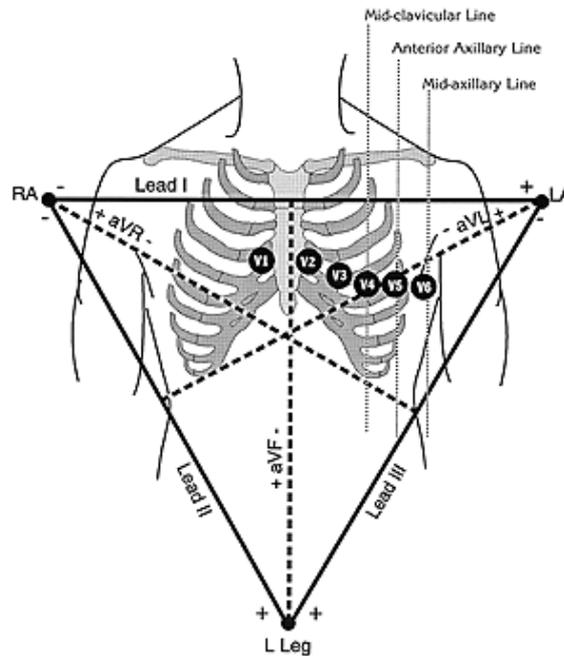


Figure 2.8 - Electrodes Placement.

Some of the ECG leads (signals transmitted and received between two electrodes) are bipolar leads (limb leads) that utilize only a single positive and a single negative electrode to measure electrical potentials between them. Others are unipolar leads (augmented leads and chest leads) that have a single positive and make use of combinations of other electrodes to serve as a composite negative electrode.

2.4.2.1 Limb leads (bipolar)

With the bipolar system, one limb is connected to the positive terminal of the recording galvanometer and another limb to its negative terminal. Three limbs (right arm-RA, left arm-LA and left leg/foot-LL) are used. As said before, an electrode on the right leg is used as a reference electrode for recording purposes. It doesn't matter if the limb leads are attached to the wrists/ankles or if they are attached to the shoulder/upper thigh, because the limb behaves as a long conductor wire.

We have the following bipolar leads:

- Lead I: RA (-) to LA (+), measures the potential difference between the two arms
- Lead II: RA (-) to LL (+)
- Lead III: LA (-) to LL (+) [16]

By the rules mentioned before it can be said that the maximal positive ECG deflection occurs in lead I when a depolarization wave runs parallel to the axis between right and left arms, a wave of repolarization also causes a positive deflection, and if the wave of depolarization moves away from the left arm (positive) the deflection is negative. The same rules can also be applicable to Leads II and III.

It is shown in Figure 2.8 that the three bipolar limb leads roughly form an equilateral triangle (with the heart at the center) which is called Einthoven's triangle.

“The triangle lies in the frontal plane of the body. The electromotive force of the heart is mathematically equivalent to a single dipole, which lies at the center of the triangle. The apices of the triangle are equidistant from the dipole.

The following Einthoven postulates are basic assumptions of the equilateral triangle hypothesis:

The dipole-the heart is a single dipole at any given moment of the cardiac cycle representing the electromotive force generated by the heart.

The electrical center-the electrical center of the heart is at the center of the dipole which coincides with the center of the triangle.

The conducting medium-from the electrical standpoint the triangle may be conceived as a homogeneous plane of conducting materials-bones, muscles, etc.”[17]

The leads can be moved, to be represented over the heart as shown in Figure 2.9, as long as the result lead is parallel and of the same polarity. [15]

The positive electrode for lead I is said to be at zero degrees relative to the heart (along the horizontal axis).

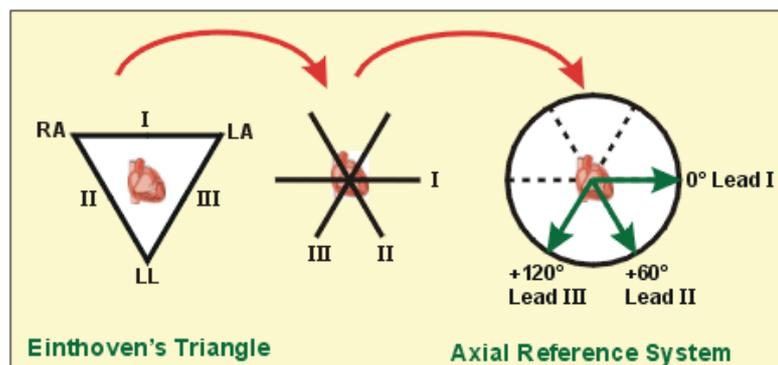


Figure 2.9 - Representation of Leads I, II and III in an Axial Reference System. [12]

2.4.2.2 Augmented Limb Leads (Unipolar)

In addition to the three bipolar limb leads described above, there are three augmented unipolar limb leads, in the frontal plane.

The same electrodes used in bipolar limb leads are now used to form the augmented lead, the ECG machine does the rearranging of the electrodes, to switch between augmented limb leads and bipolar limb leads.

They are called unipolar leads because they use only one true pole referenced against a combination of other limb electrodes. The limb connections are modified, to "augment" the size of the deflections obtained from leads L, R and F.

- aVR (Augmented Voltage from heart to Right arm) positive electrode located on right arm:

RA (+) to [LA & LL] (-);

- aVL (Augmented Voltage from heart to Left arm) positive electrode located on left arm:

LA (+) to [RA & LL] (-);

- aVF (Augmented Voltage from heart to left Foot) positive electrode located on left led:

LL (+) to [RA & LA] (-).

This three augmented leads can also be represented in the axial reference system. Relative to the lead axis I (0°) the aVL lead is at -30° , aVR at -120° and aVF is at $+90^\circ$ as depicted in Figure 2.10.

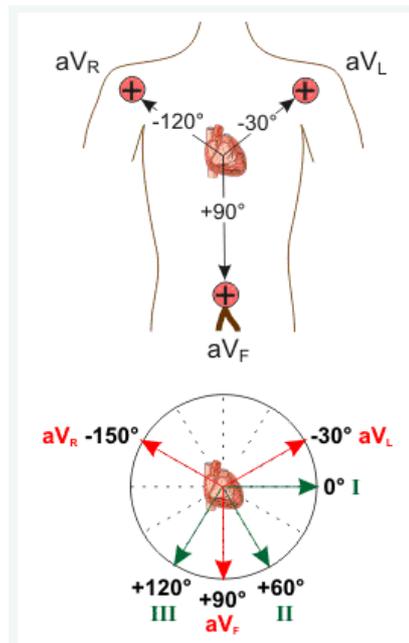


Figure 2.10 - Bipolar Limb Leads and Augmented Limb Leads Represented in the Axial Reference System. [12]

2.4.2.3 The precordial/chest leads

As said before, the view of the electrical activity of the heart from the frontal plane is not enough to monitor the heart; a view from the horizontal plane is also needed to form the 3-dimensional analysis.

In order to do that, there are the 6 precordial, unipolar chest leads, mentioned above (V1 to V6), positioned on the surface of the chest over different regions of the heart (Figure 2.11). Due to their close proximity to the heart, they do not require augmentation.

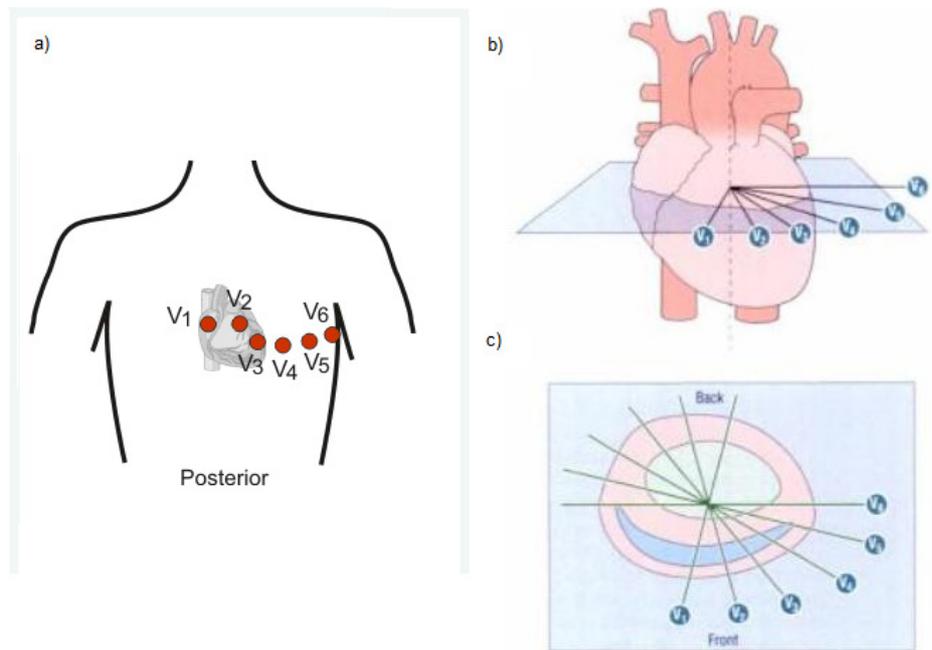


Figure 2.11 - Six Precordial Leads.
a) Positioning on the Body; [12] b) Frontal Vision of the Precordial Leads on Axial System; c) Top Vision of the Precordial Leads on Axial System. [15]

2.5 ECG Device: Holter Monitor

A Holter monitor is a battery-powered device that continuously records the electrical impulses of the heart during normal activity, through a series of electrodes attached to the chest.

The monitoring is made for at least 24 hours, which is very useful since it allows the doctor to know exactly what the patient was doing when he or she felt a specific symptom, thus helping to deduce the nature of the heart problem. [18]

Chapter 3 - ECG SIGNAL PROCESSING AND ANALYSIS

3.1 QRS detection algorithm

The objective of this work is to provide a real time detection of the heartbeats, so that they can be used for the purpose of atrial fibrillation analysis. This real-time detection method should be simple to perform and yield a fast, practically sample-to-sample, decision result.

Yet, the QRS detection is not an easy process. Apart from the variability of shapes that the QRS complex can present (Chapter 2), there are also noise sources hindering the signal processing: muscle noise; 50 Hz power-line interference, due to differences in the electrode impedances and to stray currents passing through the patient and the cables; baseline wander, that occurs, for example, due to respiration; and T waves interference, similar in frequency to the QRS complex.

In order to lower the effect of these noise sources, digital filters are applied, improving the SNR (signal-to-noise ratio).

Once the filtering is done, the process can focus on the characteristic rapid conduction and depolarization of the ventricles that cause the QRS complex to have, normally, the largest slope in the ECG signal, which corresponds to the biggest rate of voltage change (Figure 2.5). Thus, the next step should be derivation, since it is associated with the change rate, it will produce an output with QRS enhancement.

However, the standalone analysis based on the R-slope is not enough to produce reliable QRS detection. Other processing tools based on signal parameters, like the amplitude and width of the QRS complex should be taken into account, as abnormal QRS, mentioned above, may occur. For example, a QRS complex with large amplitude and long duration will hold relatively low R-wave slopes, therefore possible to be missed by the derivative analysis.

In 1985, Pan and Tompkins [19] proposed a real time algorithm based on the processing techniques mentioned above. This technique served as basis for this work; it is divided into three major processing steps, Figure 3.1: [19-21]

- Linear process: digital filtering, including band-pass filter, differentiator and moving window integrator;
- Nonlinear transformation: absolute value of the signal's amplitude;
- Decision rules: adaptive threshold and T-wave discriminator.

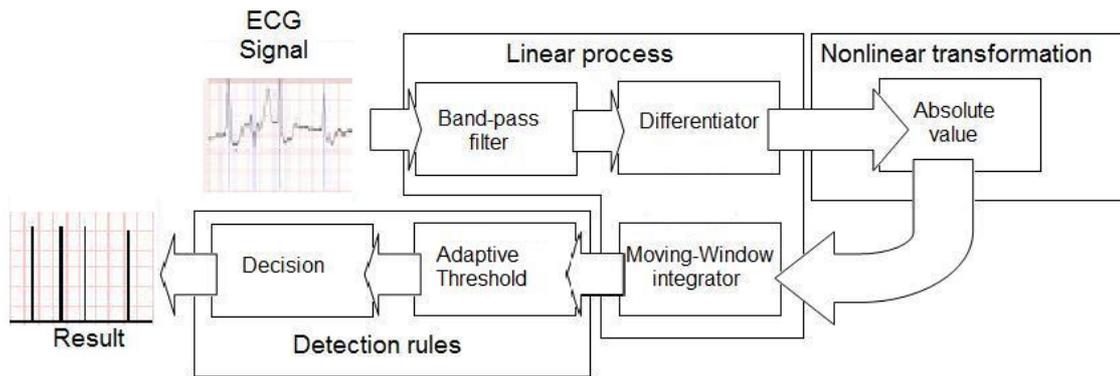


Figure 3.1 - QRS Detection Algorithm, Modified Pan-Tompkins Algorithm.

3.1.1 Band-pass filter

The setting of a band-pass filter, in order to attenuate the noise, is based on the frequency of the ECG QRS complex wave from 5 to 15 Hz, which is roughly the bandwidth that maximizes the QRS energy. [22]

As this project intends to be run in real-time, filters are designed using integer coefficients only, allowing an easier and faster filtering process, as only integer arithmetic operations are needed. But, it presents a consequence in the bandwidth frequency filtered, because integer coefficients will present a limited flexibility to the band-pass design.

According to what was said, it's not possible to directly design filters to the mentioned frequency with integer coefficients, therefore the band-pass is cascaded into low-pass and high-pass filters to achieve cut-off frequencies close to 5 and 15 Hz.

The filters used on this project arise from the ones suggested by Pan-Tompkins on [19]; however, they are not exactly the same. The change is made concerning results provided in [23].

The transfer function of the low-pass filter

$$H(z) = \frac{1}{32} \cdot \frac{(1 - z^{-5})^2}{(1 - z^{-1})^2} \quad (3.1)$$

results in the difference equation,

$$y(nT) = 2y \cdot (nT - T) - y(nT - 2T) + \frac{1}{32} \cdot [x(nT) - 2x(nT - 5T) + x(nT - 10T)] \quad (3.2)$$

where T is the sample period and n an arbitrary integer. For a sample rate of 200 Hz, the filter presents a low cut-off frequency of 13 Hz.

The frequency response is depicted in Figure 3.2.

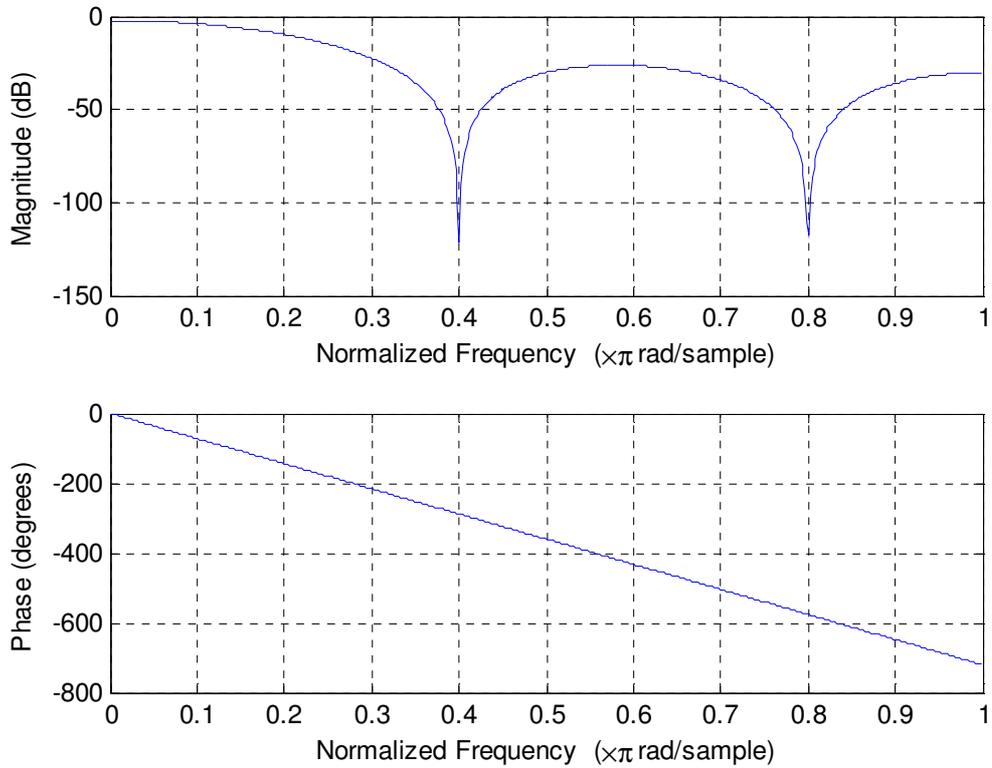


Figure 3.2 - Frequency Response of the Low-Pass Filter in Magnitude and Unwrapped Phase.

The high-pass filter presents a particularity, since it is implemented combining an all-pass filter with a low-pass filter. The low-pass filter component presents the transfer function

$$H_{lp}(z) = \frac{1 - z^{-32}}{1 - z^{-1}} \quad (3.3)$$

relating the output to the input,

$$y(nT) = y(nT - T) + x(nT) - x(nT - 32t) \quad (3.4)$$

The high-pass filter is the all-pass filter minus the low pass filter

$$H_{hp}(z) = z^{-16} - \frac{1}{32} \cdot H_{lp}(z) \quad (3.5)$$

with the difference equation,

$$p(nT) = x(nT - 16T) - \frac{1}{32} \cdot [y(nT - T) + x(n) - x(nT - 32T)] \quad (3.6)$$

The frequency response of equation (3.5) is shown in Figure 3.3. This high-pass filter has a cut-off frequency of 5 Hz.

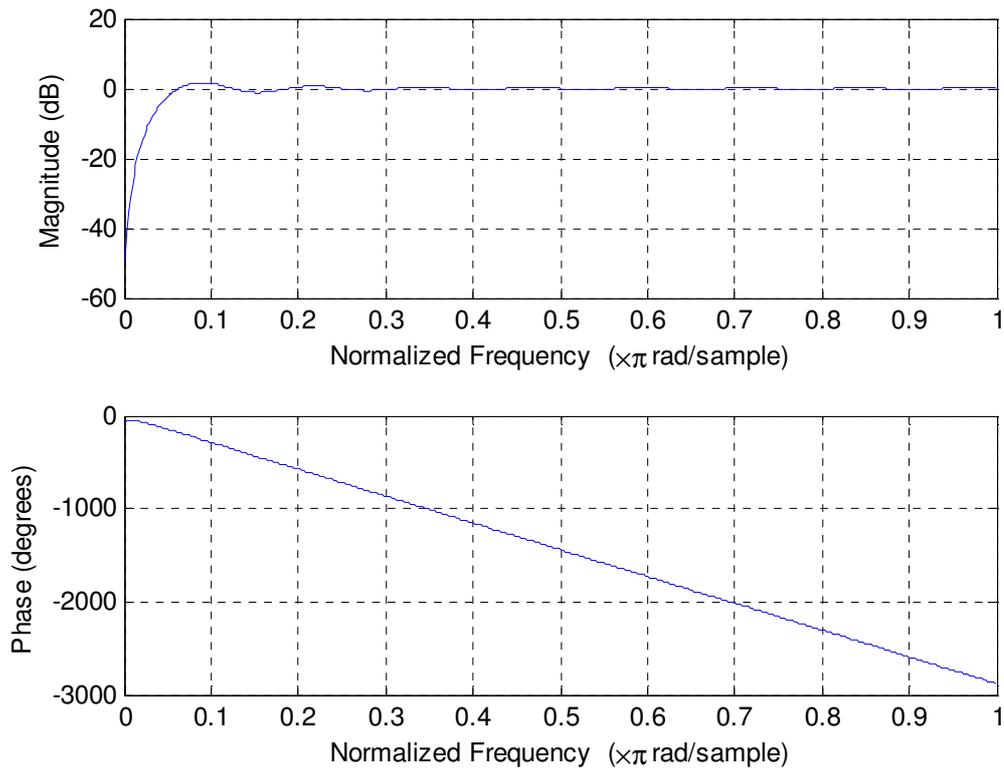


Figure 3.3 - Frequency Response of the High-Pass Filter in Magnitude and Unwrapped Phase.

The band-pass filter will be comprehended between 5 and 13 Hz - values which are relatively close to the design's purpose.

3.1.2 Differentiator

After filtering the noise, the signal can finally be differentiated to provide the QRS complex slope information. The derivative used is the one described in the Pan-Tompkins algorithm.

$$y(nT) = \frac{1}{8} \cdot (2x(nT) + x(nT - T) - x(nT - 3T) - 2x(nT - 4T)) \quad (3.7)$$

With this step, low frequency components of P and T waves are suppressed and the gain increases with the frequency, emphasizing the QRS complex because of its high slopes.

3.1.3 Absolute value

The Pan-Tompkins algorithm suggests squaring the filtered signal; however some experiences on [23] concluded that, with the squaring operation, the QRS detector becomes a little bit gain sensitive. Therefore, the absolute value is used instead, to make all data points positive with less gain sensitivity.

3.1.4 Moving-window integrator

The use of a moving-window integrator aims to solve the problem of only having information about the R-slope. It allows the extraction of the waveform feature information of the QRS complex, width and amplitude. Also, in the ECG there is broadband noise, which means noise overlapping the signal; the use of traditional linear filters will remove some of the ECG frequency components, thereby distorting the ECG. Averaging the signal is the solution, as it improves the SNR (signal-to-noise ratio).

If the window chosen is too large, it's easy to conclude that the T wave can be merged with QRS waves, producing an error. If too small, some QRS can produce several peaks in the integration waveform, which can also result in an error.

Hence, the choice of the width of the moving-window is a very important step on the detector algorithm.

Once the width is chosen, the output of the integrator can be calculated following,

$$y(nT) = \frac{1}{N} \cdot [x(nT - (N - 1)T) + x(nT - (N - 2)T) + \dots + x(nT)] \quad (3.8)$$

where N is the number of samples, which are chosen taken into account the width of the QRS complex and the sample frequency. In the original Pan-Tompkins algorithm the width is 150 ms, in order to allow wide QRS complexes. Since then, studies show that a narrow window produces better results [24]. In this project, the width of the window used is 80 ms, at a frequency of 200 Hz, 16 samples are averaged ($N = 16$).

3.1.5 Detection rules: Adaptive threshold and decision

It was already discussed that different patients have different ECG signals, and multiple signal morphologies can also occur for the same patient. The main goal is the production of an algorithm that can be adapted to the widest possible range of different ECGs, with quality results.

Bearing this in mind, Pan-Tompkins built an algorithm that promotes the use of an adaptive threshold. The algorithm adapts to changes in the ECG signal by computing estimates of noise and signal peak heights. [19-20]

Even with the changes promoted in this project's algorithm (for example, absolute value instead of squaring phase) the produced graphical result should be similar to the one presented in Figure 3.4 (note the increase in the time delay, with each new step; this issue is addressed in Chapter 4 - ONBOARD IMPLEMENTATION).

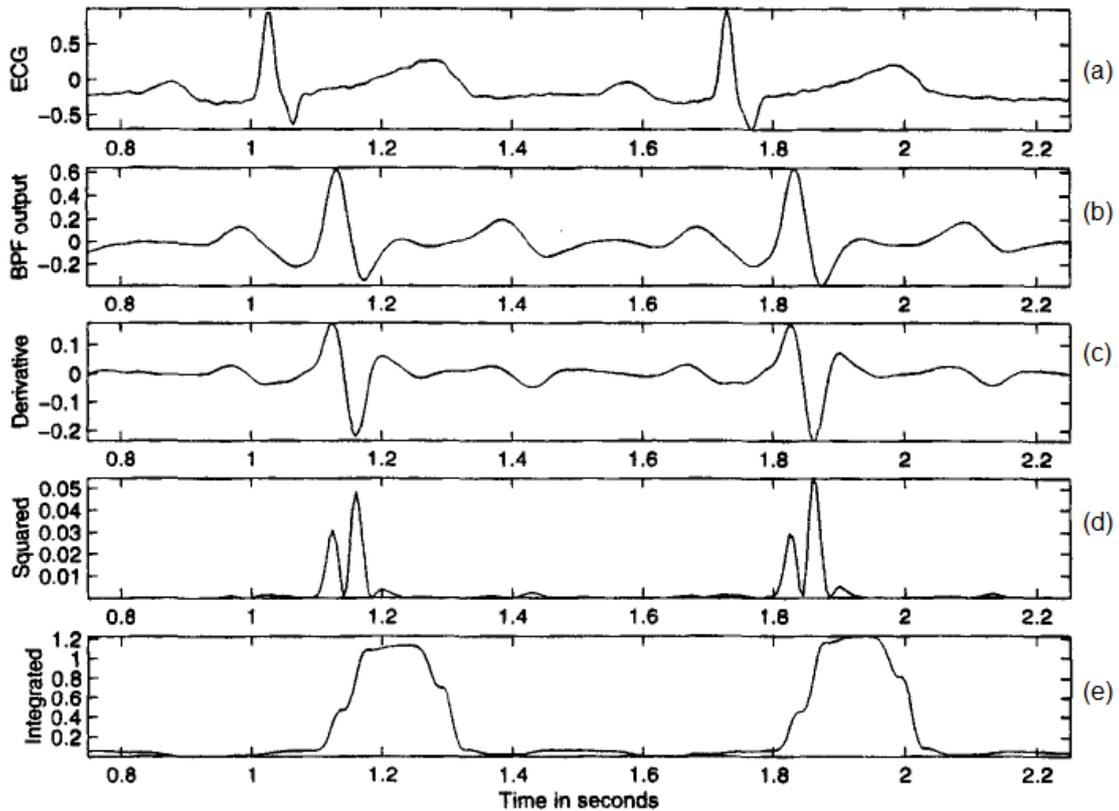


Figure 3.4 - Output of Each Stage of the Original Pan-Tompkins Algorithm.
 (a) Original Signal; (b) Output of Band-Pass (High-Pass and Low-Pass) Filter; (c) Output of Derivation; (d) Output of Squaring; (e) Output of Integration. [20]

After applying the moving-integrator, the produced output will be something similar to what is shown in Figure 3.4 (e). Figure 3.5 represents, in detail, one possible lump of the integration output.

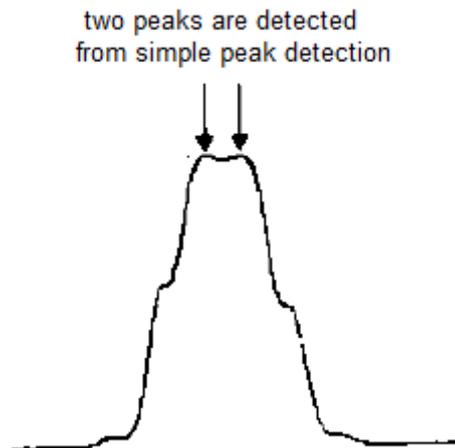


Figure 3.5 - Typical Waveform of Time-Averaged Signal during the QRS Complex.

As it can be seen, more than one peak can be detected. It is the algorithm's task to detect which one is correct, by following a set of rules. [19, 23]

- **Rule 1:** QRS complex can't occur in a period of time shorter than 200 ms, because of the refractory period, hence, the detection algorithm should ignore all peaks that precede or follow larger peaks by less than 200 ms;
- **Rule 2:** A peak is considered whenever the final output changes direction within a specified interval. The raw signal should present the positive and negative slopes, if not, the peak represents a baseline shift;
- **Rule 3:** If the peak is larger than the detection threshold it is considered as QRS complex, otherwise it is considered noise;
- **Rule 4:** If no QRS is detected within 1.5 R-R interval, it means that there was a peak narrower than the detection threshold, but larger than half the detection threshold, following the preceding detection by at least 360 ms, that peak should be found and considered as QRS complex.

After detecting a peak (two first detection rules followed), it is necessary to classify it as QRS complex or noise. In order to decide this, a threshold is used (see the two final detection rules).

As said before, in this algorithm, the threshold adapts to changes in the ECG signal. The method is this: every time a peak is classified as a QRS complex it is added to a buffer containing the eight most recent QRS peaks. In the same way, a peak that is not classified as a QRS peak (considered noise) is added to a buffer containing the eight most recent noise peaks. The detection threshold makes use of the average of the noise and QRS peaks that the buffers contain in each moment.

$$Detection_{threshold} = Npeak_{average} + TH \cdot (QRSpeak_{average} - Npeak_{average}) \quad (3.9)$$

where $Npeak_{average}$ is the average of the eight last peaks considered as noise, $QRSpeak_{average}$ is the average of the eight last peaks considered as QRS peaks, TH is the threshold coefficient, with the value 0.375 (determined experimentally). [23]

The output of this stage corresponds to the final output of the algorithm, a pulse every time a QRS complex is detected (result obtained in the diagram shown in Figure 3.1).

3.2 Signal analysis

After the signal is processed, and the QRS complex is estimated, the algorithm uses a set of specifications for signal monitoring.

In this step the project had the help of Doctor Del Greco (Santa Chiara Hospital, Trento - Italy), who suggested a possible analysis, based on the measurement of R-R intervals (heart rate periods), in order to detect anomalies in the heart rhythm.

3.2.1 Heart-rate

R-R intervals are averaged on 30 second-interval non-overlapping windows. The algorithm registers a possible heart-rate deficiency in this 30 second-window, whenever the result of $60/average$ is < 40 or > 100 bpm (beats per minute). Which relates to and confirms what has been said on Chapter 2 – ECG: the average heart rhythm on an adult is 70 bpm (> 40 and < 100).

3.2.2 Atrial fibrillation detection

Variations between R-R intervals are also calculated on 30 second-non-overlapping windows. Following Doctor Dell Greco indications, the algorithm detects a possible atrial fibrillation during these 30 seconds, if during this period more than 4 variations higher than 20 % are detected.

Chapter 4 - ONBOARD IMPLEMENTATION

The algorithm, described on chapter 3, is implemented in C (based on studies of EP limited [23]) to be run in real-time on a dsPIC® DSC microcontroller.

There is little doubt concerning the capability of the dsPIC to run the algorithm in real-time, nevertheless integer arithmetic operations are used; filter, differentiator and integrator are implemented only resorting to the use of add and shift operations, avoiding the use of multiplications of any precision. This results in lighter processing, allowing future researchers to use the algorithm on more limited microprocessors, as well as to add new detection features or analysis apart from atrial fibrillation, keeping the program quality to be run in real-time.

4.1 FLEX Full Base Board

This project intends to use an external gateway to process the signal and to detect events. In order to do that, the FLEX Full Base Board (Figure 4.1) – integrating a standard Microchip dsPIC ® DSC microcontroller – is used for the program implementation.

FLEX is an embedded board designed with the goal of exploiting all the potential of dsPIC ® DSC microcontrollers. It provides a set of 2.54 mm connectors to export all the microcontrollers' connections, so that a user can easily connect to its desired ports, when building a specific application.

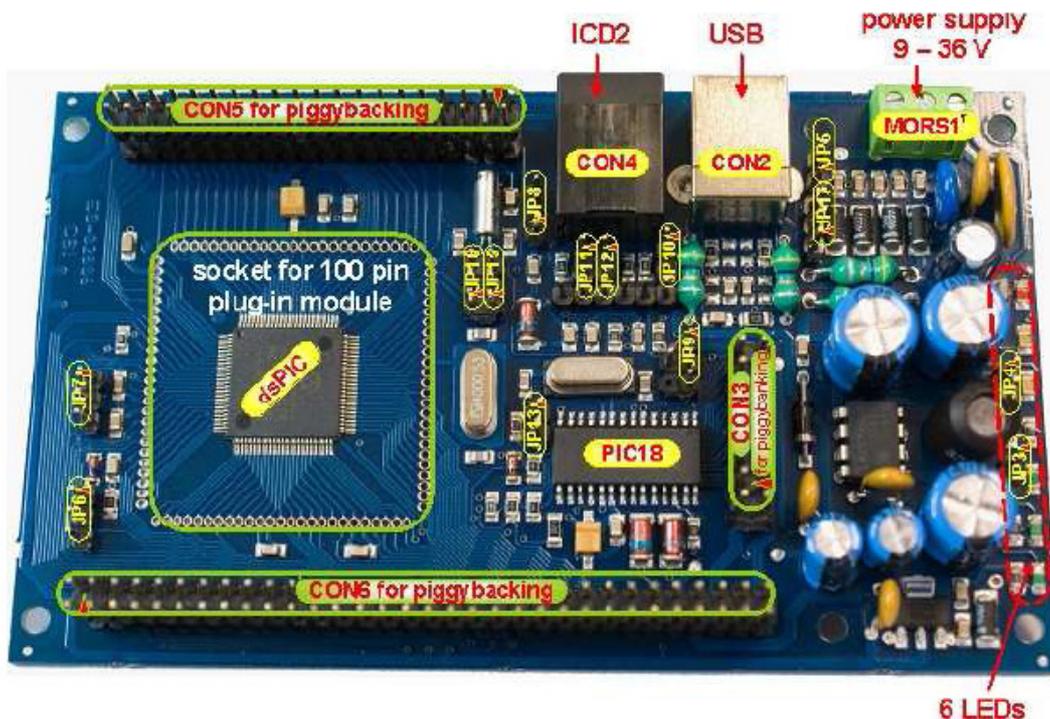


Figure 4.1 - FLEX Full Base Board. [25]

This board integrates an extra-robust power-supply circuit; allowing a wide range of power suppliers – accepts voltages between 9 and 36 Volts.

Although it includes a USB port for data transfer or to be used as programming interface, the PIC18® PIC18F2550 – microcontroller for integrated programming – hadn't been programmed yet so as to make use of these USB functionalities when the experimental part of the project took place.

The socket for the 100-pin plug-in module is made available to mount a microcontroller – through plug-in modules (PIMs) made available from the microchip. In this project this is not relevant as the microcontroller soldered on the surface of the board is sufficient to reach the project's goal.

The board also supplies a set of LEDs, part monitor the function of the board and the others can be controlled to be used in the program, for example when a specified event is detected.

An ICD-2 in circuit program connector provided by the board is used to program the microcontroller.

CON 3, 5 and 6 are the connectors for Daughter boards piggybacking. [25]

The programmer selected to connect the PC and the ICD-2 in circuit program connector is the MPLAB ICD2.



Figure 4.2 - MPLAB ICD 2

4.1.1 Daughter boards

Daughter boards are boards with specific features that can be added on top of FLEX base boards (by piggybacking). For example, this project makes use of the FLEX Multibus Base Daughter Board (Figure 4.3), used to simplify the work with the communication peripherals integrated on the microcontroller, together with the FLEX Multibus RS232 Module mounted on the UART2 slot, to work with the serial input/output module available on the microcontroller used.

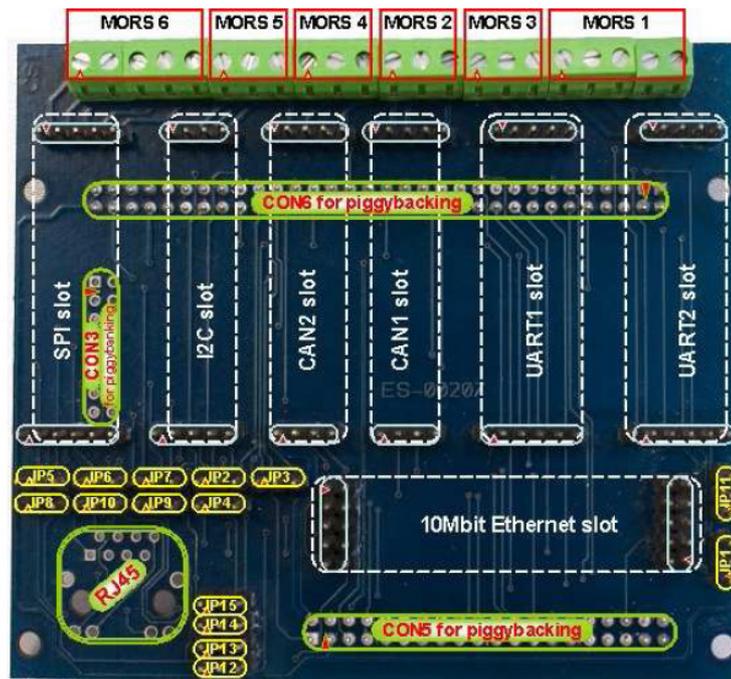


Figure 4.3 - FLEX Multibus Base Daughter Board. [25]

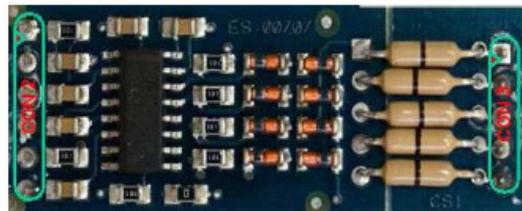


Figure 4.4 - FLEX Multibus RS232 Module. [25]

4.1.2 Microchip dsPIC® DSC microcontroller dsPIC33FJ256MC710

When a program needs to perform a large number of mathematic operations on a large set of data quickly, the use of a Digital Signal Processor (DSP) represents a good option. This microprocessor is optimized for fast operational needs of a digital signal processing work.

The microcontroller chosen employs a powerful 16-bit architecture, integrating the control features of a Microcontroller Unit (MCU) with the computational capabilities of a DSP. As a result, one obtains an ideal device for applications that require high speed, repetitive computation, as well as control.

The dsPIC33FJ256MC710 has a program flash memory with 256 Kbytes and 30 Kbytes of RAM for data storage. The use of CMOS flash technology brings two advantages to the use of this microcontroller: low-power consumption and high-speed of the flash technology.

The microcontroller has 2 ADC with 32 channels and 2 UART, one of these serial input/output modules is used on the program to export results, and ADC can be useful in future developments. [26]

One of the big advantages of using the FLEX boards is the set of predefined libraries and complex applications based on the FLEX base board and on Daughter boards; leaving only to the developer the task of implementing the program logic.

As the FLEX base board already integrates this microprocessor and since its use brings no limitations to the implementation and running of the algorithm, this microcontroller can be used.

4.2 QRS detection implementation

The implementation of the QRS detection for a frequency of 200 Hz is in

Appendix A – qrs.c.

A simplified version of the detector can be seen in Figure 4.5.

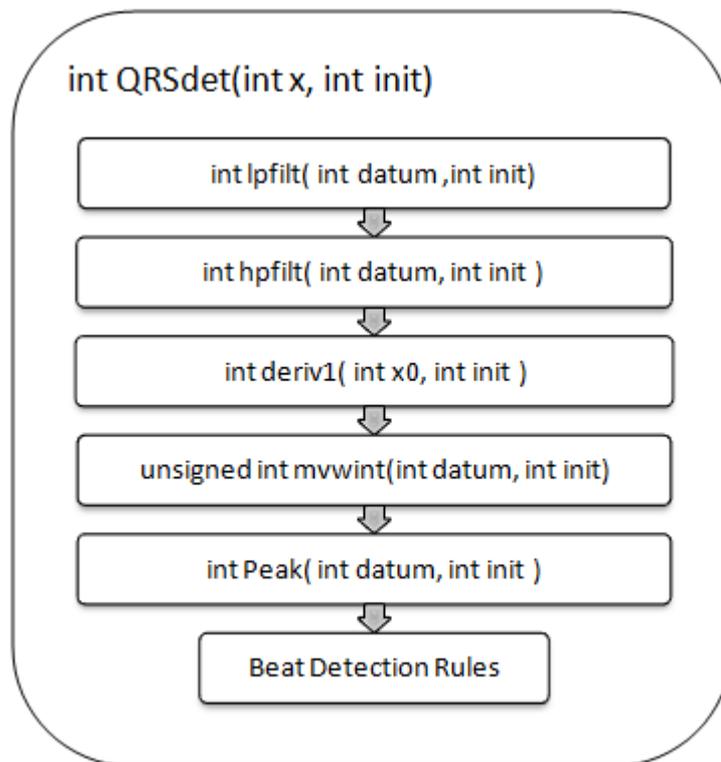


Figure 4.5 - Beat Detection Algorithm Sequence.

Before inputting the signal all filters are initialized, passing a non-zero value to the variable 'init', on QRSdet(), which is the same as saying that the filters are reset, so as

to ensure that the values of previous ECG signal analysis will not influence the new one. On initialization the value passed to “x” is irrelevant.

After initialization, the first sample of the ECG trace can be passed, for evaluation, to QRSdet “x”. Attention should be paid to the variable “init”, that has to be zero until a new analysis is desired, or else it will restart the filters.

On **QRSdet()** the following sequence takes place:

- Function **lpfilt()** is called, passing in “datum” the sample present in “x”. **lpfilt()** implements the digital filter represented by equation (3.2), that shows that the previous ten input arguments, and the last two outputs, must be stored to estimate the difference. Returns the filtered sample to QRSdet();
- The output of **lpfilt()** is the input (“datum”) of **hpfilt()**. This function implements the high-pass filter represented by equation (3.6). The last two filtered samples are also used, but in this case a larger buffer (for input values) than in **lpfilt()** will be needed as the last thirty-two inputs must be present to produce the result of the difference equation. Returns the signal filtered by the band-pass;
- Function **deriv1()** receives this filtered signal in “x0” and implements the derivative approximations according to the equation (3.7), using the previous four inputs. QRSdet() receives the sample derivative;
- The algorithm calculates the absolute value of the sample. If the sample is negative it becomes positive, if it is already positive nothing is done;
- **Mwint** implements the moving window integrator represented by equation (3.8). The averaging of the signal over the last sixteen samples takes place. The new sample present in “datum” is added, while the older one (that occurred sixteen samples, i.e. 80 ms, ago) is subtracted;
- The sample can now be passed to **Peak()**. This function returns the peak height as soon as 95 ms have passed after the peak height is detected or when the signal returns to half its peak height;
- The peak found can be noise peak or a QRS peak, as seen before. It’s up to the **Beat detection rules**, to determine whether this peak is noise or a QRS. In this step the rules present in “3.1.5 Detection rules: Adaptive threshold and decision” are tested.

There are two possibilities, **Peak()** returns a peak height, which means that a peak was detected, or it returns zero, meaning no peak is ready to be tested.

If no peak is detected two things can happen: there's already one peak waiting for the 200 ms to pass (rule 1), or no peak was found yet. In case the first one happens, the algorithm will check if the 200 ms have passed and if they have, the peak is passed for evaluation.

If the Peak() returns a peak height, also the two cases are possible: in case no peak was already found, the output of Peak() is the new peak; in case there's another peak waiting for the 200 ms to pass, check to see if the new one is higher, if it is, restart counting the 200 ms.

Once a peak is passed for evaluation, the algorithm will decide whether it is peak or noise, with the help of the detection threshold, if the peak is below the threshold it is considered noise, if not it is considered as a QRS peak.

The threshold is not static, it adapts to the signal, using the average of the 8 last QRS peaks and 8 last noise peaks, as shown by equation (3.9). These averages are estimated on UpdateQ() and UpdateN(), for QRS and noise, respectively. Each function receives the respective peaks, and updates the *detection_threshold*, with respect to its new average.

Since detection threshold depends on previous peaks, in the beginning the program can't possibly deduce which ones are QRS and which ones are noise peaks. To solve this problem, in the beginning the algorithm detects the maximum peaks in eight consecutive 1-second intervals, and considers them the first eight QRS peaks; the first eight noise peaks are considered zero. The detection threshold can then be estimated.

If a peak is below the detection threshold, apart from updating the noise peaks array (function UpdateN();), a condition will check rule 4; that is to say that if more than 360 ms have passed since the last QRS peak was found, this noise peak is stored as a "search back peak". If this peak is at least half of the detection threshold, it is added to the QRS buffer (function UpdateQ();) and the detection threshold is updated;

- The algorithm returns the peak value if it is considered QRS, if not it returns zero.

4.2.1 Detection delay

The filters described in "3.1.1 Band-pass filter" as well as the moving-window integrator are different from the ones described in the Pan-Tompkins algorithm, whose implementation results in the graphs shown in Figure 3.4. Therefore the detection delay (period of time that the algorithm takes to detect the QRS complex) will also be different.

$$Det_{delay} = bpfilter_{delay} + dif_{delay} + mw_{delay} + 200\ ms \quad (4.1)$$

Where Det_{delay} is the total detection delay; $bp\ filter_{delay}$ the time delay that the band-pass filter adds; dif_{delay} delay added by the differentiator; mw_{delay} the delay associated to the moving-window integration width and; 200 ms the delay required by rule 1.

If the peak is found using rule 4 (search back condition) the delay will be this delay plus half the average of the R-R interval.

The Delay added by the moving-window integrator is know, since it is the width of the window considered, 80 ms or 16 samples, for a frequency of 200 Hz. The ones that have to be estimated are the delays induced by the band-pass filter and differentiator.

4.2.1.1 Band-pass filter delay

The filters implemented are FIR filters (Finite Impulse Response), showing a symmetrical impulse response depicted in Figure 4.6 for the low-pass filter and Figure 4.7 for the high-pass filter, for a sampling frequency of 200 Hz.

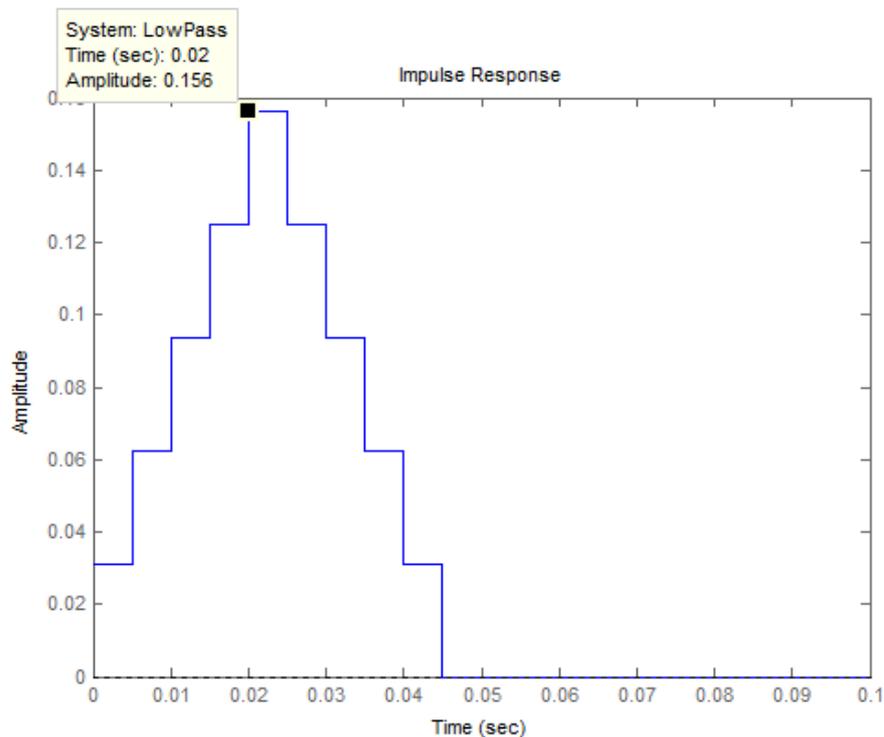


Figure 4.6- Low-Pass Filter Impulse Response.

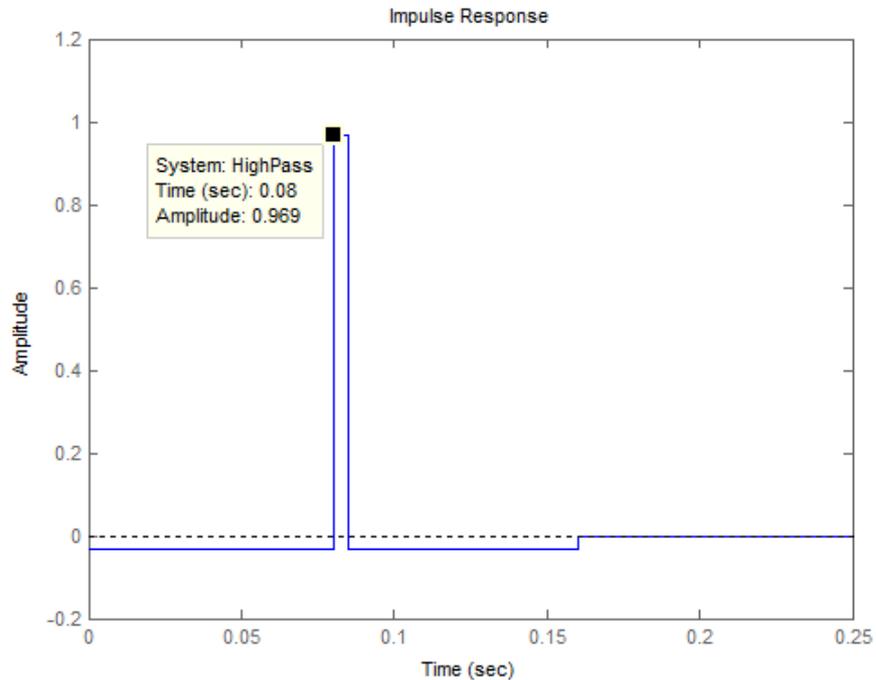


Figure 4.7- High-Pass Filter Impulse Response.

This symmetrical characteristic of the implemented FIR filters (its coefficients are symmetrical around the center coefficient, i.e. the first coefficient is the same as the last; the second is the same as the next-to-last, etc.) shows that they are linear-phase filters. This can be seen in Figure 3.2 and Figure 3.3, for low and high-pass filters, respectively, the phase response of the filter is a linear (straight-line) function of frequency. This implies that the filters do not cause “phase distortion” or “delay distortion”. All the samples suffer the same delay.

The delay of these filters is the center coefficient, or point of symmetry. On the low-pass filter the delay is 20 ms, or 4 samples, with a sample frequency of 200 Hz; the high-pass has a delay of 80 ms, or 16 samples, with the same 200 Hz of sample frequency.

4.2.1.2 Differentiator delay

Although the differentiator presents an inverse symmetry, Figure 4.8, the same analysis performed for the filters can be done to estimate the delay introduced by the differentiator.

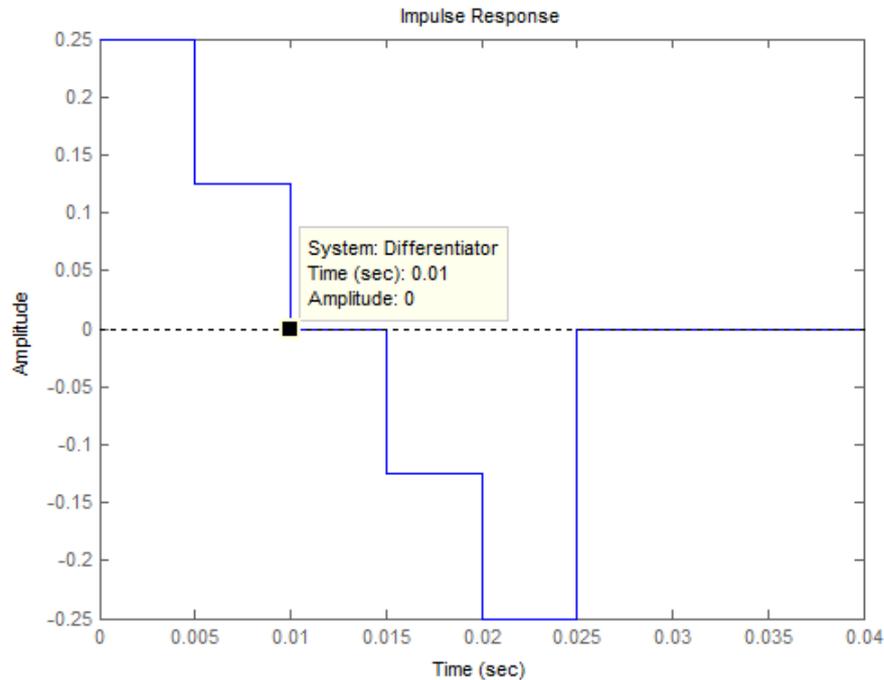


Figure 4.8 - Differentiator Impulse Response.

The differentiator causes a delay of 0.01 ms, or 2 samples, considering a sample frequency of 200 Hz.

4.3 Atrial fibrillation detection implementation

The algorithm implementation for a frequency of 200 Hz is in Appendix B – alarmDet.c.

Atrial fibrillation detection is estimated by the function:

- unsigned int alarmDet(unsigned int y);

After QRSdet(), follows an algorithm to calculate R-R intervals – time between consecutive R-waves – to be input to alarmDet(). If no peak is detected (QRSdet() returns zero) zero is passed to alarmDet() on “y”; if a peak is found (QRSdet() returns a value different than zero) the value passed to “y” is the number of samples that have passed between this peak and the one that occurred before. Concerning the first peak, there’s no previous peak, so zero is passed to “y”.

At the top of alarmDet() a condition will test to see if a R-R interval (in number of samples) is received, if so it is stored in a buffer, containing consecutive R-R intervals (inside the same 30-second window). Another condition follows, that checks to see if the 30 seconds (window of analysis) have passed (3.2 Signal analysis). When this is

confirmed, the algorithm is ready to test for the presence of anomalies in the last 30 seconds.

The search for anomalies on the last 30 seconds is performed in the following manner:

- if 60 divided by the average of the R-R intervals – stored in the vector during the last 30 seconds – is smaller than 40 or bigger than 100 (3.2.1 Heart-rate), the function returns the value '1';
- the vector is organized in an increasing order, so as to reduce computational time; if the difference between the biggest and the smallest R-R interval is smaller than 20 %, the difference between the other R-R intervals will also be smaller than 20%, thus the other intervals no longer need to be compared; if it's higher than 20 %, the algorithm will compare the lower values with the highest R-R interval. If less than 5 variations higher than 20 % are found, the algorithm compares the higher values with the lowest R-R interval.

Whenever 5 variations higher than 20% are found, no more comparisons are made. This way, the algorithm only compares all the elements in the vector if less than 5 variations are present, saving in computational time.

The function returns the value '2' when more than 4 variations higher than 20% are found (3.2.2 Atrial fibrillation detection);

- if the previous two cases happen, alarmDet() returns the value '3';
- value '10' is returned if no anomalies were found.

Chapter 5 - EXPERIMENTAL RESULTS

5.1 MIT-BIH Arrhythmia Database

The MIT-BIH Arrhythmia Database is a set of standard test ECG recordings for evaluation of arrhythmia detectors. These records are the ones used as input for the algorithm mentioned above, in order to test its performance, and can be downloaded from [27].

Between 1975 and 1979, Beth Israel Hospital Arrhythmia Laboratory, performed 4000 24-hour Holter recordings (in two leads) in a mixed population of inpatients (about 60%) and outpatients (about 40 %). The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts taken from the 4000 24-hour recordings, divided in two main groups:

- The first group of records has the objective of testing the algorithm for different types of waveforms that an arrhythmia detector can encounter during clinical routine. In order to do this, 23 half-hour excerpts (from record 100 to 124, except records 110 and 120) were randomly chosen from the set of 4000. The segments chosen to this group would only be excluded if the ECG signal, from both channels, didn't have enough quality to be analyzed by human experts.
- The second group corresponds to the remaining 25 records (from record 200 to 234, with some records missing). They arise from the same set, but instead of being randomly chosen they are selected to present significant difficulties to the arrhythmia detector. Records including less common but clinically important arrhythmias, variation of QRS morphology, or signal quality implying difficult analysis.

The subjects were 25 men aged 32 to 89 years old and 22 women aged 23 to 89 years old. (Records 201 and 202 came from the same male subject.)

The database provided two channel ambulatory ECG recordings – in channel one modified limb lead II (MLII); and in channel two, modified lead V1 (occasionally V2 or V5, and in one record V4), – however the algorithm used only works with one of this channels. It is always considered the first, since normal QRS complexes are usually prominent in MLII; and normal beats are nearly biphasic, causing the second lead – which is almost orthogonal to the mean cardiac electrical axis – to present a signal that can be approximately isoelectric.

As said before, this is a set of ECG standard signals to allow the developer to test his algorithm efficiency, comparing its results with the ones present on previous signal analysis annotations, made available on the MIT-BIH database. These annotations were made by a cardiologist who classified each record, labeling every beat, adding rhythm labels (for example Normal sinus rhythm, Atrial fibrillation...), signal quality labels (extreme noise, missed beats...) and comments. [27]

The analog ECG signals are converted into digital (ADC – analog-to-digital converter), at 360 samples per second (on each channel) with 11-bit resolution over a range of 5 mV. These digital values correspond to the input of the algorithm; being the previous ADC unipolar, sample values range from 0 to 2047 ($2^{11}-1$), corresponding 1024 to zero volts.

The algorithm, after receiving the previous values, puts them on a different range, setting the baseline to zero and the resolution 0.005 mV (200 units/mV).

5.2 Test QRS Detection Algorithm

5.2.1 Sensitivity and Positive Predictive

The analysis of these two statistical measures has the objective of measuring the performance of classification tests; therefore they are of the utmost importance to test the quality of the implemented solution.

Sensitivity is a measure of the portion of actual positives correctly identified by the algorithm; the relation between the number of true positives considered by the tested algorithm and the sum of this number with the ones that the algorithm considers false but turn to be true – number of false negatives.

$$\text{Sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \quad (5.1)$$

In the sense of this project, sensitivity reflects the relation between the number of correctly identified QRS complexes by the algorithm and the number of QRS complexes that should have been identified, i.e. the number of QRS complexes considered in the annotations provided by the MIT-BIH database.

But this measure alone doesn't provide all of the information about the behavior of the algorithm, probability of wrong positive detections should also be considered. Positive Predictive relates the number of correct true positives the algorithm finds with the total number of positives it considers.

$$\text{Positive Predictive} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}} \quad (5.2)$$

Positive predictive can be explained for this project as the relation between the number of correctly identified QRS complexes by the algorithm and the total number of QRS complexes it considers.

Positive predictive studies the effect of false QRS detections, while sensitivity gives an idea about the probability the algorithm has of missing a QRS. [27]

5.2.1.1 Sensitivity and Positive Predictive Test

To evaluate the behavior of the detector in terms of sensitivity and positive predictive the instructions described in Appendix C – How to Run the Program on the PC, are followed.

Table 5.1 and Table 5.2 show the results of the tests executed for each record.

RECORD	'square'		'Absolute'	
	QRS sensitivity (%)	QRS positive predictive (%)	QRS sensitivity (%)	QRS positive predictive (%)
100	99.95 (1901/1902)	100.00 (1901/1901)	99.95 (1901/1902)	100.00 (1901/1901)
101	100.00 (1523/1523)	99.87 (1523/1525)	100.00 (1523/1523)	99.87 (1523/1525)
102	99.95 (1820/1821)	100.00 (1820/1820)	99.95 (1820/1821)	100.00 (1820/1820)
103	99.88 (1727/1729)	100.00 (1727/1727)	99.94 (1728/1729)	100.00 (1728/1728)
104	95.91 (1781/1857)	99.94 (1781/1782)	99.57 (1849/1857)	99.89 (1849/1851)
105	99.77 (2150/2155)	97.42 (2150/2207)	99.95 (2154/2155)	97.86 (2154/2201)
106	99.65 (1690/1696)	100.00 (1690/1690)	99.88 (1694/1696)	100.00 (1694/1694)
107	99.89 (1782/1784)	100.00 (1782/1782)	99.89 (1782/1784)	100.00 (1782/1782)
108	97.57 (1444/1480)	98.43 (1444/1467)	99.59 (1474/1480)	95.40 (1474/1545)
109	99.86 (2096/2099)	100.00 (2096/2096)	99.95 (2098/2099)	100.00 (2098/2098)
115	99.94 (1636/1637)	100.00 (1636/1636)	99.94 (1636/1637)	100.00 (1636/1636)
117	99.92 (1283/1284)	100.00 (1283/1283)	99.92 (1283/1284)	100.00 (1283/1283)
200	99.86 (2165/2168)	99.95 (2165/2166)	99.82 (2164/2168)	99.95 (2164/2165)
202	99.41 (1860/1871)	100.00 (1860/1860)	99.73 (1866/1871)	100.00 (1866/1866)
220	99.94 (1693/1694)	100.00 (1693/1693)	99.94 (1693/1694)	100.00 (1693/1693)
221	99.70 (2014/2020)	100.00 (2014/2014)	99.95 (2019/2020)	100.00 (2019/2019)
232	99.93 (1484/1485)	100.00 (1484/1484)	100.00 (1485/1485)	99.87 (1485/1487)
233	99.88 (2558/2561)	100.00 (2558/2558)	99.92 (2559/2561)	100.00 (2559/2559)
Total	99.51 (32607/32766)	99.74 (32607/32691)	99.88 (32728/32766)	99.62 (32728/32853)

Table 5.1 – Sensitivity and Positive Predictive Test for 'square' and 'absolute', sampled at 200 Hz.

RECORD	Median		mean	
	QRS sensitivity (%)	QRS positive predictive (%)	QRS sensitivity (%)	QRS positive predictive (%)
100	99.95 (1901/1902)	100.00 (1901/1901)	99.95 (1901/1902)	100.00 (1901/1901)
101	100.00 (1523/1523)	99.87 (1523/1525)	100.00 (1523/1523)	99.93 (1523/1524)
102	99.95 (1820/1821)	100.00 (1820/1820)	99.95 (1820/1821)	100.00 (1820/1820)
103	99.94 (1728/1729)	100.00 (1728/1728)	99.94 (1728/1729)	100.00 (1728/1728)
104	99.95 (1856/1857)	99.89 (1856/1858)	99.89 (1855/1857)	99.84 (1855/1858)
105	99.91 (2153/2155)	98.09 (2153/2195)	99.86 (2152/2155)	97.86 (2152/2199)
106	99.88 (1694/1696)	100.00 (1694/1694)	99.94 (1695/1696)	99.94 (1695/1696)
107	99.94 (1783/1784)	100.00 (1783/1783)	100.00 (1784/1784)	100.00 (1784/1784)
108	99.66 (1475/1480)	95.41 (1475/1546)	99.53 (1473/1480)	96.53 (1473/1526)
109	99.95 (2098/2099)	100.00 (2098/2098)	99.95 (2098/2099)	100.00 (2098/2098)
115	99.94 (1636/1637)	100.00 (1636/1636)	99.94 (1636/1637)	100.00 (1636/1636)
117	99.92 (1283/1284)	100.00 (1283/1283)	99.92 (1283/1284)	100.00 (1283/1283)
200	99.82 (2164/2168)	99.95 (2164/2165)	99.86 (2165/2168)	99.91 (2165/2167)
202	99.73 (1866/1871)	100.00 (1866/1866)	99.73 (1866/1871)	100.00 (1866/1866)
220	99.94 (1693/1694)	100.00 (1693/1693)	99.94 (1693/1694)	100.00 (1693/1693)
221	99.80 (2016/2020)	100.00 (2016/2016)	99.80 (2016/2020)	100.00 (2016/2016)
232	100.00 (1485/1485)	99.87 (1485/1487)	100.00 (1485/1485)	99.73 (1485/1489)
233	99.92 (2559/2561)	100.00 (2559/2559)	99.92 (2559/2561)	100.00 (2559/2559)
Total	99.90 (32733/32766)	99.63 (32733/32853)	99.90 (32732/32766)	99.66 (32732/32843)

Table 5.2 – Sensitivity and Positive Predictive Test for 'median_mean', sampled at 200Hz.

	'square'	'Absolute'	median	mean
True positive	32607	32728	32733	32732
False positive	84	125	120	111
False negative	159	38	33	34

Table 5.3 – Conclusions based on the previous tests run for the 4 different types of implementations tested, on the 18 records.

		'absolute'			
RECORD	QRS sensitivity (%)	QRS positive predictive (%)	RECORD	QRS sensitivity (%)	QRS positive predictive (%)
100	99.95 (1901/1902)	100.00 (1901/1901)	109	99.95 (2098/2099)	99.95 (2098/2099)
101	100.00 (1523/1523)	99.87 (1523/1525)	115	99.94 (1636/1637)	100.00 (1636/1636)
102	99.89 (1819/1821)	99.95 (1819/1820)	117	99.92 (1283/1284)	100.00 (1283/1283)
103	99.94 (1728/1729)	100.00 (1727/1727)	200	99.91 (2166/2168)	99.45 (2166/2178)
104	99.95 (1856/1857)	99.94 (1856/1876)	202	99.68 (1865/1871)	100.00 (1865/1865)
105	99.86 (2152/2155)	98.35 (2152/2188)	220	99.94 (1693/1694)	100.00 (1693/1693)
106	99.29 (1684/1696)	100.00 (1684/1684)	221	98.02 (1980/2020)	100.00 (1980/1980)
107	100.00 (1784/1784)	100.00 (1784/1784)	232	100.00 (1485/1485)	99.80 (1485/1488)
108	99.26 (1469/1480)	95.33 (1469/1541)	233	99.77 (2555/2561)	100.00 (2555/2555)
Total QRS sensitivity (%)		Total QRS positive predictive (%)			
99.73 (32677/32766)		99.62 (32677/32803)			

Table 5.4 - Sensitivity and Positive Predictive Test for 'absolute', sampled at 360 Hz.

	'Absolute'	
	Sampled at 200Hz	Sampled at 360Hz
True positive	32728	32677
False positive	125	126
False negative	38	89

Table 5.5 - Comparison Between the use of 200Hz and 360Hz for Sample Frequency, based on Sensitivity and Positive Predictive results.

5.2.2 Performance, Delays and Errors

In the next steps, in order to simplify the reading of the inputs, the program developed in the 'ReadFromMITDB' folder reads the signal, with the help of the WFDB libraries, and stores it in .txt files, with a sampling frequency of 200 Hz (and in another test 360 Hz). By doing this the following programs will not need to incorporate the MIT-BIH libraries, very useful, for example, when implementing the program on the board.

The inputs in .txt files are stored in './reorg2/dTxt/X.txt'; with 'X' as the number of the record.

The folders present in 'QRSpc.rar' are '200.rar' and '360.rar' which contain the following files – with different input frequencies, 200 Hz and 360 Hz, respectively:

- 'ReadFromMITDB': whenever a different record should be tested, it has to be added in 'inputs.h', the program present in this folder will generate the .txt file corresponding to the new sample;
- 'reorg2': after adding a new record, it should be pasted inside the folder './reorg2/dTxt'. The program in reorg2 is useful to help with Matlab® analysis;
- 'picLP', 'picHP', 'picDV', 'picMW', 'picPK', 'picrealPK', 'picRR', 'AtrialF': each folder implements the same function but prints different pieces of information. 'picLP' prints, for each record, the signal after the low-pass filter; 'picHP' after high-pass filter; 'picDV' after derivation and after taking the absolute value; 'picMW' after the moving-window; 'picPK' after the peak is detected; 'picrealPK' after peak is considered true or false; 'picRR' prints the R-R intervals for each record; and 'AtrialF' from 30 to 30 seconds it will say if a arrhythmia was found for the previous 30 seconds. Except for 'picRR', the others show the effect that each function produces.

To run one of these programs, steps 4-6 referred in Appendix C – How to Run the Program on the PC, should be taken.

These folders are meant to be run on the computer. But, the goal of this project is to make the detector run on a different terminal – FLEX base boards. In order to compare the possible limitations of the board implementation, the previous programs are implemented in a similar way on the board. "QRSboard.rar" contains "QRS200.rar" and "QRS360.rar", which are the files that implement the program. The instructions to execute them are detailed on

Appendix D – How to Run the Program on the board.

The software necessary to run programs on the board is made available on the cd provided with this paper, on a file called 'software.rar':

- **RT-Druid**. A development environment for ERIKA Enterprise. Based on Eclipse; which gives full support for **microchip C30 compiler**, which is a full-

featured C compiler. RT-Druid configures the applications needed and is responsible for the compilation;

- **MPLAB IDE v8.00.** After compilation, a file called pic30.cof is created, this file is imported by MPLAB IDE, which is responsible for debugging, so that the program can be written into the dsPIC EPROM flash.
- **Terminal v1.9b** or **Putty.** A simple terminal to read the results that the RS232 is sending to the computer.

The results obtained on both cases (on the computer and on the FLEX board) are depicted in the following images.

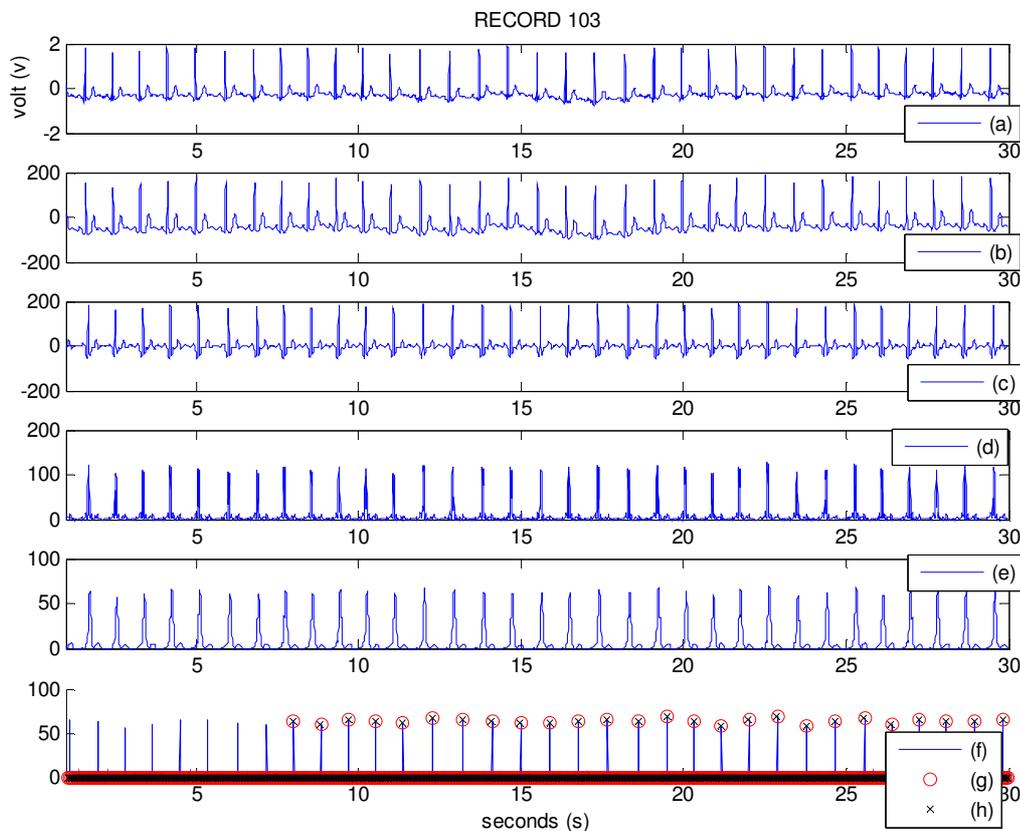


Figure 5.1 - Output of Each Stage of the Implemented Algorithm.
(a) 30 seconds of RECORD 103, 200Hz; **(b)** Output of Low-Pass Filter; **(c)** Output of High-Pass Filter; **(d)** Output of Derivation + Absolute value; **(e)** Output of Integration; **(f)** Output of Peak detection; **(g)** Peaks Considered True QRS Peaks by the Computer; **(h)** Peaks Considered True QRS Peaks by the dsPIC.

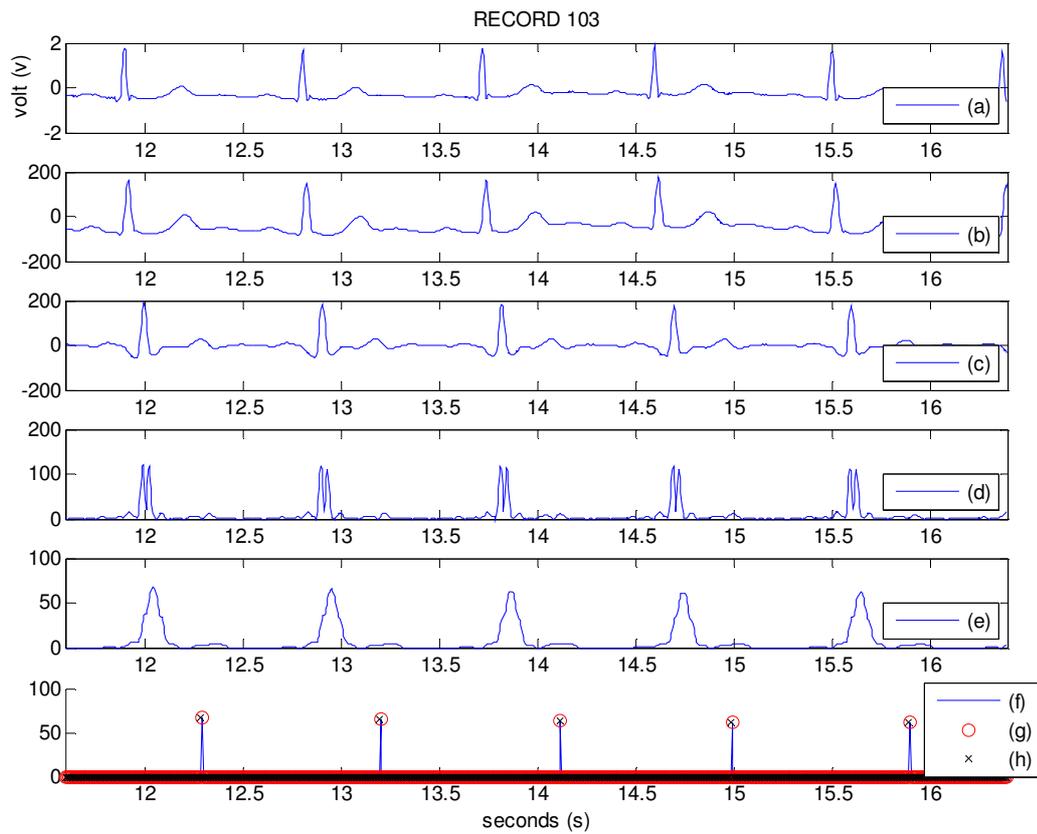


Figure 5.2 - Detail Analysis of the Record 103, Sampled at 200 Hz.
(a)Original signal; (b) Ouput of Low-Pass Filter; (c) Output of High-Pass Filter; (d) Output of Derivation + Absolute value; (e) Output of Integration; (f) Output of Peak detection; (g) Peaks Considered True QRS Peaks by the Computer; (h) Peaks Considered True QRS Peaks by the dsPIC.

From the analysis of Figure 5.2, it is possible to make an estimate of the delay that each function presents. On Table 5.6 the delays are considered regarding previous ones (i.e. Integration delay is the delay on the processing caused exclusively by Integration).

	1st Peak (s)	2nd Peak (s)	3rd Peak (s)	4th Peak (s)	5th Peak (s)
Original Peak	11.90	12.81	13.72	14.60	15.50
Low-Pass Delay	0.02	0.01	0.01	0.02	0.02
High-Pass Delay	0.08	0.08	0.09	0.08	0.07
Derivation Delay	0.01	0.01	0.01	0.01	0.02
Integration Delay	0.08	0.08	0.08	0.08	0.08
Peak Delay	0.20	0.21	0.20	0.19	0.20
Final Peak	12.29	13.20	14.11	14.98	15.89
Total Delay	0.39	0.39	0.39	0.38	0.39

Table 5.6 – Analysis of delays represented in Figure 5.2.

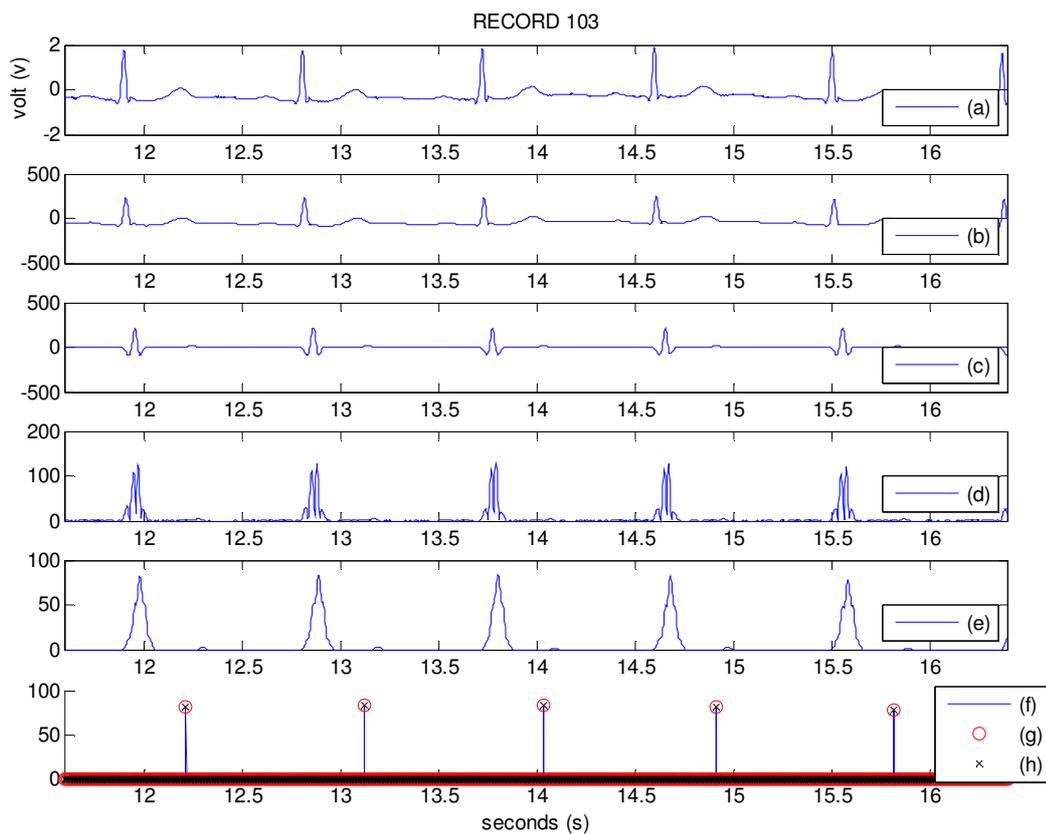


Figure 5.3 - Detail Analysis of the Record 103, Sampled at 360 Hz.
 (a)Original signal; (b) Output of Low-Pass Filter; (c) Output of High-Pass Filter; (d) Output of Derivation + Absolute value; (e) Output of Integration; (f) Output of Peak detection; (g) Peaks Considered True QRS Peaks by the Computer; (h) Peaks Considered True QRS Peaks by the dsPIC.

	1 st Peak (s)	2 nd Peak (s)	3 rd Peak (s)	4 th Peak (s)	5 th Peak (s)
Original Peak	11.90	12.81	13.72	14.60	15.50
Low-Pass Delay	0.01	0.01	0.01	0.01	0.01
High-Pass Delay	0.04	0.04	0.04	0.04	0.04
Derivation Delay	0.01	0.01	0.01	0.01	0.01
Integration Delay	0.05	0.05	0.05	0.05	0.05
Peak Delay	0.20	0.20	0.20	0.20	0.20
Final Peak	12.21	13.12	14.03	14.91	15.81
Total Delay	0.31	0.31	0.31	0.31	0.31

Table 5.7 – Analysis of delays represented in Figure 5.3

Record 103 is relatively simple to analyze due to its regular behavior, but there are some ECGs that present irregular characteristics, like the following.

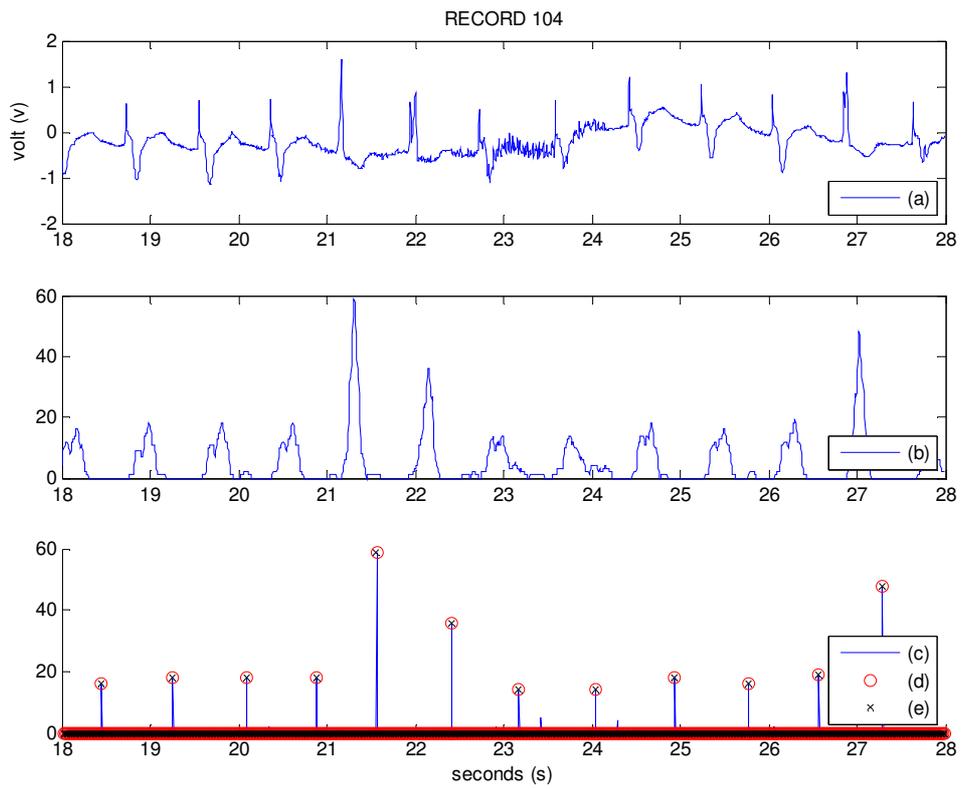


Figure 5.4 – Detail of ECG with Variations on Peak Height.
(a) Record 104 sampled at 200Hz; (b) Output of Integration; (c) Output of Peak detection; (d) Peaks Considered True QRS Peaks by the Computer; (e) Peaks Considered True QRS Peaks by the dsPIC.

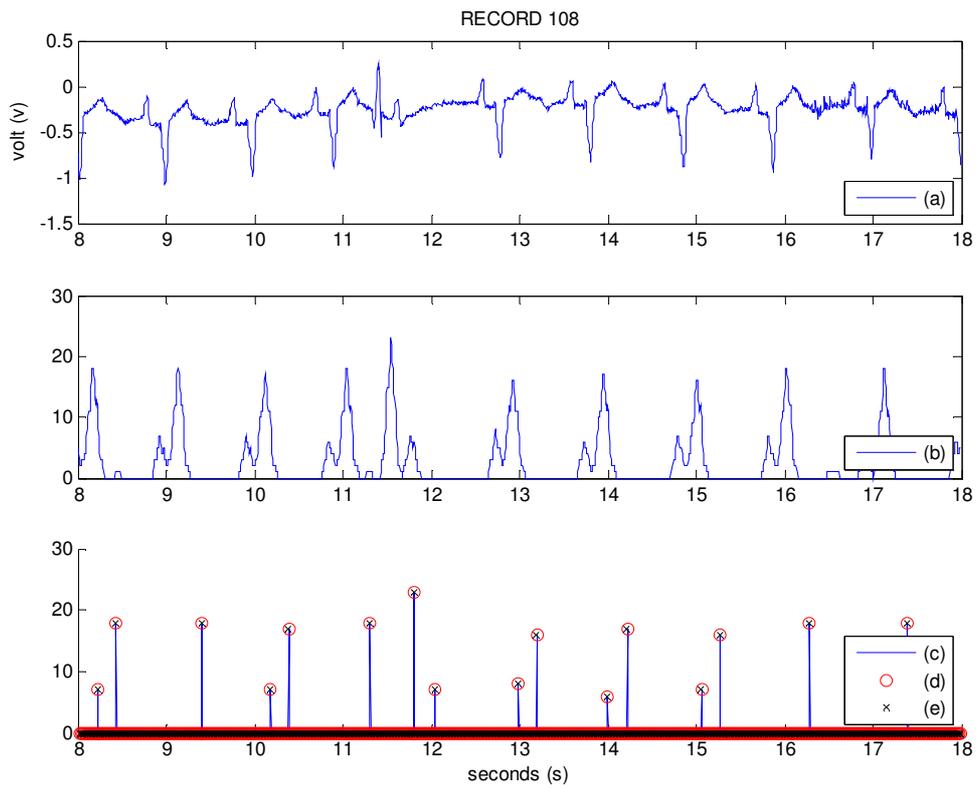


Figure 5.5 – Detail of ECG with Large P Waves.
(a) Record 108 sampled at 200Hz; (b) Output of Integration; (c) Output of Peak detection; (d) Peaks Considered True QRS Peaks by the Computer; (e) Peaks Considered True QRS Peaks by the dsPIC.

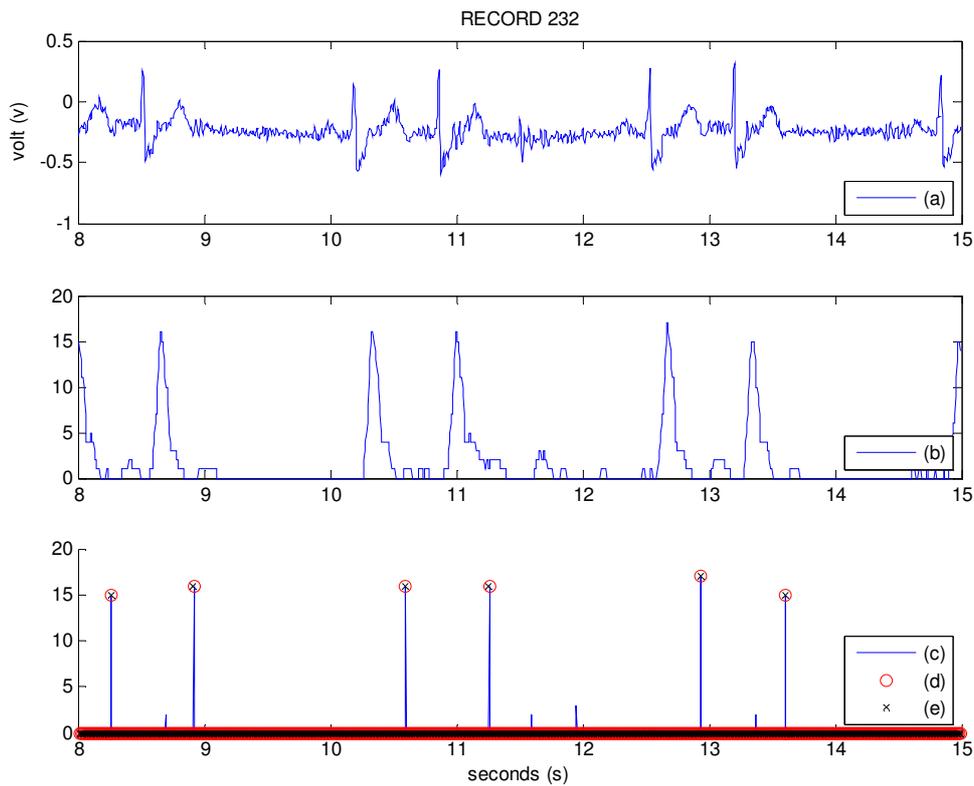


Figure 5.6 - Detail of ECG with Irregular beats.
(a)Record 232 sampled at 200Hz; (b) Output of Integration; (c) Output of Peak detection; (d) Peaks Considered True QRS Peaks by the Computer; (e) Peaks Considered True QRS Peaks by the dsPIC.

The next tables show how the algorithm behaves in terms of delay for these 3 cases.

	Average Delay (ms)
Record 104	450
Record 108	410
Record 232	400

Table 5.8 – Average Delays of Records 104, 108 and 232.

In order to understand how the P waves are considered QRS peaks on Figure 5.5 it is useful to know their location.

In Reality these peaks are...	P wave	T wave	P wave	P wave	P wave
Time (s)	10.18	12.04	12.98	14.00	15.06
Real QRS peak occurs in...(ms)	+ 210	-240	+210	+220	+210

Table 5.9 - False QRS Peaks Detected.

Executing the program 'picRR' for all the records for the frequencies of 200Hz and 360Hz, and consulting the results of R-R intervals on the physionet web page [10], it is possible to estimate the relative error.

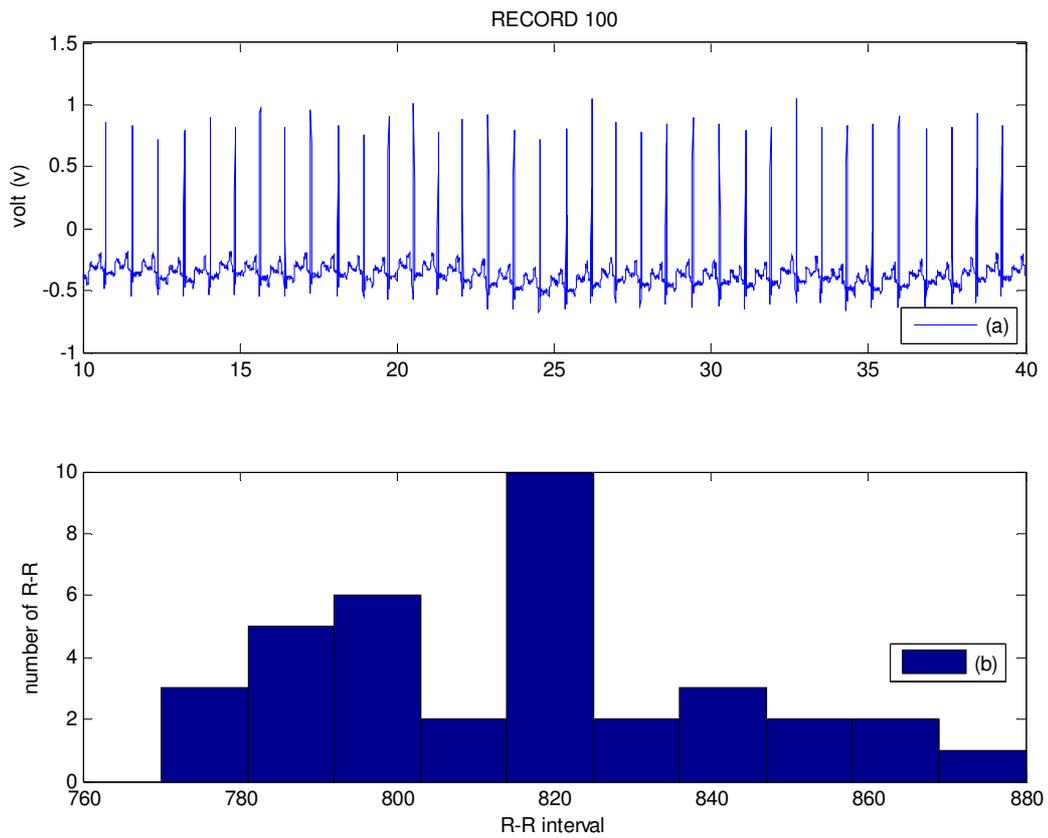
$$Relative\ Error = \frac{|real_{R-R} - experimental_{R-R}|}{real_{R-R}} \cdot 100\ (%) \quad (5.3)$$

Records	Relative Error (%)		Records	Relative Error (%)	
	200 Hz	360Hz		200Hz	360Hz
100	0.29	0.31	109	0.62	0.66
101	0.27	0.14	115	0.23	0.20
102	2.43	0.61	117	0.75	0.68
103	0.27	0.14	200	2.40	2.66
104	9.62	1.07	202	0.41	0.22
105	0.30	0.53	220	0.27	0.17
106	0.29	0.16	221	0.47	1.79
107	0.43	0.18	232	0.28	0.22
108	39.11	48.05	233	1.57	10.89

Table 5.10 – Relative Error of R-R Intervals, for 30 seconds of each Record.

5.2 Test Atrial Fibrillation Algorithm

The Atrial Fibrillation algorithm consists only on the analysis of the results. Takes the values of the R-R intervals and concludes about their meaning. Next it is showed some results.



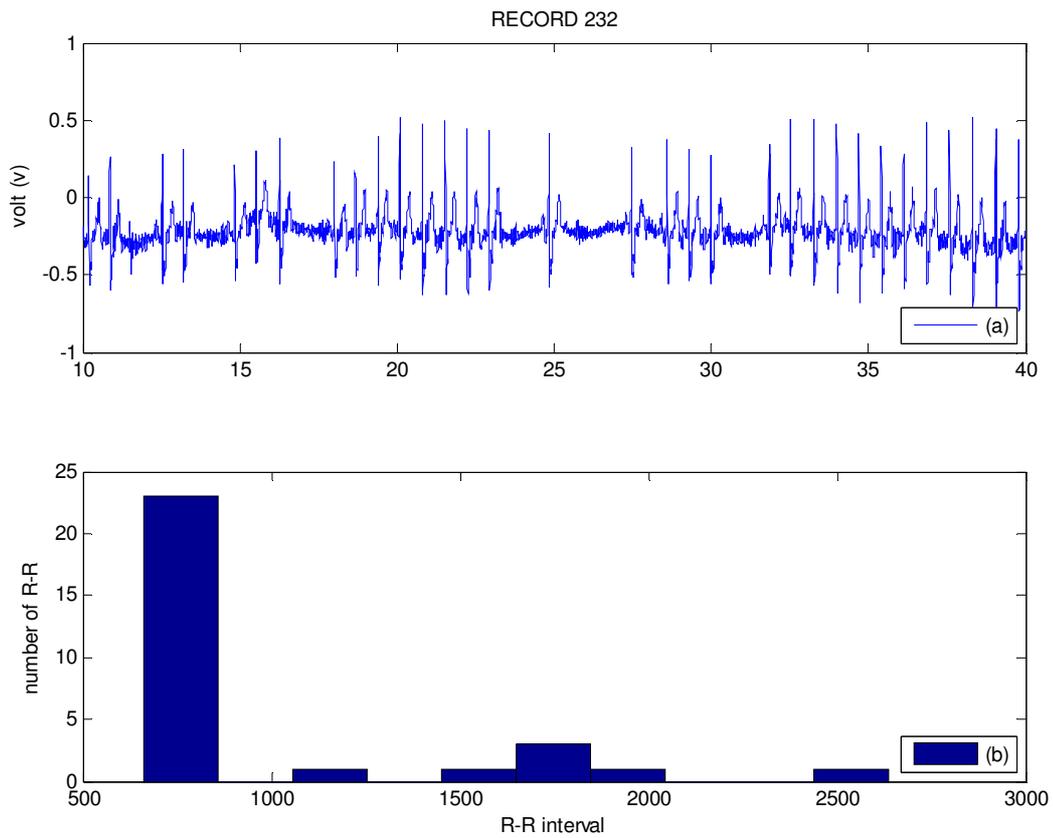
**Figure 5.7 – 30 seconds of Record 100, sampled at 200Hz.
 (a) original signal; (b) Representation of the dispersion of the R-R intervals, in the 30 seconds.**

For this interval of time the answer of the algorithm is '10', which represents no alarm is sent. From the last figure it's possible to conclude that the maximum variation is smaller than 20 %,

$$variation = \frac{880 - 780}{880} \cdot 100 \Leftrightarrow variation = 11.36\% \quad (5.4)$$

Averaging the R-R intervals, the result obtained is 816 ms. Following the algorithm,

$$40 < \frac{60}{average} < 100 \Rightarrow 40 < \frac{60}{0.816} < 100 \Rightarrow 40 < 73.5 < 100 \text{ (bpm)} \quad (5.5)$$



**Figure 5.8 - 30 seconds of Record 232, sampled at 200Hz.
 (a) original signal; (b) Representation of the dispersion of the R-R intervals, in the 30 seconds.**

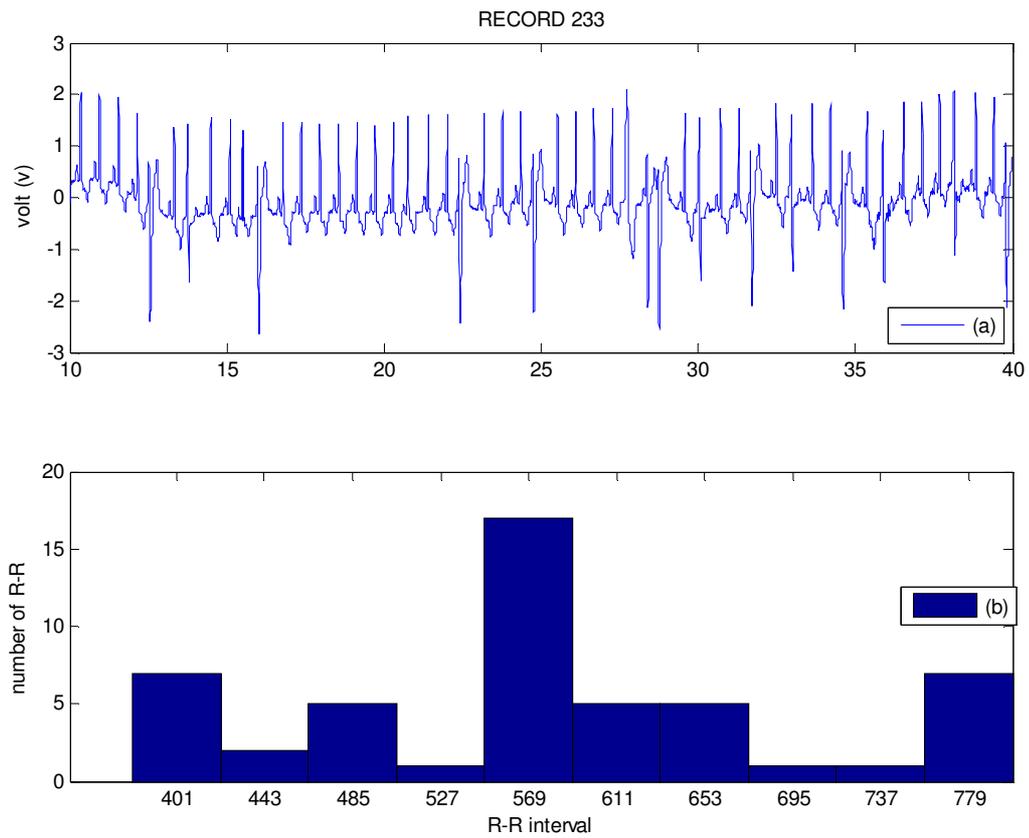
For this case the algorithm sends the alarm '2' (more than 4 variations higher than 20% are found).

$$variation = \frac{2500 - 1000}{2500} \cdot 100 \Leftrightarrow variation = 60\% \quad (5.6)$$

As for the Average, its result is 962 ms, which leads to

$$40 < \frac{60}{0.962} < 100 \Rightarrow 40 < 62.4 < 100 \quad (\text{bpm}) \quad (5.7)$$

Another example where the more than 4 variation are found', is record 233. But in this case the alarm sent is '3', because also the heart rate is considered dangerous.



**Figure 5.9 - 30 seconds of Record 233, sampled at 200Hz.
 (a) original signal; (b) Representation of the dispersion of the R-R intervals, in the 30 seconds.**

Using the same expression for variation,

$$variation = \frac{779 - 401}{779} \cdot 100 \Leftrightarrow variation = 48.5\% \quad (5.8)$$

And the Average of the R-R intervals is 581 ms, which leads to

$$40 < \frac{60}{0.581} < 100 \Rightarrow \text{False condition. } \frac{60}{0.581} = 103.4 (> 100) \text{ (bpm)} \quad (5.9)$$

Chapter 6 – CONCLUSIONS

6.1 Result Interpretation

Some heart signals are relatively easy to analyze, because they present the shape expected for a normal ECG signal, i.e. a “normal” QRS complex, approached on 2.3 – ECG Signal Characteristics. When different characteristics of QRS complexes occur (also visible on the aforementioned chapter), the analysis becomes harder.

However, by the study of sensitivity and positive predictive on signals with different characteristics (5.2.1.1 Sensitivity and Positive Predictive Test), it can be concluded that regardless of the chosen implementation, it adapts to multiple waveforms; being the lowest sensitivity obtained in record 104, with a value of 95.91% (corresponding to 76 false negatives in 1857 detected peaks) using the algorithm that implements the “square” of the signal after the derivation step (as suggested on the Pan-Tompkins implementation) and the lowest positive predictive value 95.40% (corresponding to 71 false positives in 1545 detected peaks) on record 108, using the “absolute” value after the derivation method.

The four QRS detection algorithms tested produce very good results in detecting QRS complexes for the 18 records tested; choosing between them is a matter of detail, since they all produce similar results. Yet a choice has to be done.

Choosing between the “absolute” method and the “square” method can be justified by the sensitivity study. On the 18 records, the “square” method presents a high number of False negatives when compared with the ‘Absolute’ method (159 False negatives against the 38 found with the ‘Absolute’); in this way, many QRS complexes would be considered false, which damages the quality of the analysis. On the other hand, the ‘square’ method has a lower number of false positives for the records tested (84 against 125 from the ‘Absolute’), but the difference is not as significant as the difference in terms of False negatives (Table 5.3).

Between median and mean methods, the decision was to estimate the threshold with the mean value, since the difference shown on Table 5.3 is not significant; the choice was made bearing in mind that irregular beats can occur. Median is more affected with irregularities. Considering, for example, that there is a beat much lower than the others, the median value will be more affected than the mean. A good example that justifies this choice is depicted in Figure 5.4.

In conclusion, the decision was to implement the ‘Absolute’ method, estimating the threshold with the mean of QRS and Noise peaks.

Figure 5.1 shows the analysis of 30 seconds of the ECG signal of Record 103, as it can be seen in Figure 5.1 (a), this ECG signal presents “normal” QRS complexes, which simplifies the analysis. On the first seconds the program does not consider them as real peaks; this has to do with the adaptive threshold; to compute the equation (3.9) and have a threshold value, the eight last values of QRS peaks must be known. The first eight

peaks are considered QRS peaks for threshold creation, but are not considered real peaks for analysis purposes.

Concerning board implementation, the results produced were exactly the same as the ones obtained with the computer. This demonstrates the high capacity of the dsPIC to execute all the processing in a very short period of time.

On Figure 5.2 and Figure 5.3 show the similarity of the algorithm implemented to the original Pan-Tompkins algorithm depicted in Figure 3.4.

Table 5.6 and Table 5.7 verify what was said about the time the solution takes to detect Peaks (4.2.1 Detection delay) – Detection Delay.

Using an input frequency of 200Hz the detection delay is roughly 390 ms, corresponding to the minimum detection delay of the algorithm. While recurring to an input frequency of 360Hz results in a lower detection delay for the same signal, around 310 ms.

Since the device used implies no limitations to using a frequency of 360Hz, based on the delay value, the input frequency should be 360Hz. But, accordingly to Table 5.5, this frequency carries a higher number of false negatives (89 false negatives against the 38 using 200Hz for sampling). Therefore it is a tradeoff between missed QRS complexes and detection delay.

The main problem of the delay is the 200 ms period that follows a QRS peak detection to ensure that no larger peaks exist in this period. One possible solution is to ignore the 200 ms that follow a peak, but this can be dangerous in irregular beats. On Figure 5.5 it's possible to see the effect of larger P waves, resulting on False QRS peaks. Ignoring the 200 ms that follow the peak would result in the detection of only these large P waves.

Record 108 is a particular case where P waves are considered QRS peaks (and also a T wave). Observing Table 5.9 it's possible to see that the detection occurs in the border, the near real Peaks occur roughly at a distance of 210 ms. If it was 200 ms, these P waves would be ignored.

Results on the Table 5.8 show that the delay is not 390 ms (for 200Hz) in all cases, this is justified by rule 4 described in 3.1.5 Detection rules: Adaptive threshold and decision; according to this rule the maximum delay will be the 390 ms plus the average of the R-R interval. This can be a problem, that can be solved by removing the condition that follows rule 4, but it can mean the loss of real peaks. In this project the rule was followed at the cost of a longer delay.

In conclusion, the program has a delay that can go from 390 ms to 390 ms plus half the average of the R-R interval (for 200Hz), or 80 ms less for 360 Hz.

The most important measurement of the processing part of the algorithm is the R-R interval, since it is the measure used to detect events of alarm. These intervals depends on the correct detection of QRS complexes, therefore, if the detection is deficient, it will cause errors on the R-R interval estimation, causing the algorithm of detecting completely useless events.

On Table 5.10 it can be seen that in general the algorithm for detection peaks is working really well, with the difference from the real R-R interval being less than 1% on most cases. The differences are due to small oscillations on the delay – as it can be seen for the 4th peak on Table 5.6. On the others, the errors are related to significant wrong detections on the 30 seconds of representation. The most obvious is Record 108 – with values of 39.11% for 200Hz and 48.05% for 360 Hz. The false peaks, caused by P and T waves, completely change the R-R interval values (Figure 5.5).

Concerning the Atrial Fibrillation algorithm, as said, it depends only on the R-R intervals. On the experiences made on the QRS detector, it was proved that it is reliable, which is the same as saying that the QRS peaks found can be trusted to calculate the R-R intervals, used to detect situations of alarm.

As said in 2.1.2 – Rate, a normal heart rate is about 70 bpm. On Figure 5.7 and Figure 5.8, it's represented two ECG signals with normal beats, being 73.5 bpm for Record100 and 62.4 for Record 232. But, on the 30 seconds showed of the Record 232 an arrhythmia is present. By equation (5.6) it can be seen a variation of 60% between the highest and the lowest R-R interval, observing the histogram on Figure 5.8 (b), it's possible to see that the majority of the values of the R-R intervals will be around the minimum, so a variation around 60% will be verified for all this values when compared with the highest value.

The same occurs for the 30 seconds observed of the Record 233 on Figure 5.9, but in this case with a lower variation – 48.5% – between the extreme values of R-R. Again, since many of this R-R are located around the minimum (and also around the maximum) more than 4 variations bigger than 20% will be verified. In this 30 seconds analyzed the heart rate is 103.4 which passes the higher limit, meaning a pace too fast(Tachycardia).

For these 3 cases the alarms generated are correct, which attests the correct implementation of this mathematical analysis.

It can be said that the algorithm of alarm is just mathematical analysis, thereby its errors on the results depend only on the errors of the QRS detector. Thus, the quality of the final work depends only on the quality of the QRS detector.

There was an intention of testing the final module with a real signal through a sensor including electrodes – accordingly to the theory described in 2.4.1 Electrodes – developed specially for this project, fulfilling the requirements of low power consumption and low price. Unfortunately this device break before tests could be done. Nevertheless the program with the ADC configured correctly for an input voltage of 0 to 3 Volts, is made available on the CD – by the name of ADCinput.rar – for future developers.

Although it wasn't a goal of this project, this work is ready to be used for analysis of previous ECG (instead of use it only for real-time application). For example, the data

generated by the Holter device (2.5 ECG Device: Holter Monitor), can be processed with the help of this work, in order to lighten up the Doctors work.

6.2 Future Developments

In order to consider the extra noise of signals coming from real ECG electrodes (and acquisition electronics) and lost of resolution due to lower resolution of the dsPIC (its ADC is only 10 bits) the tests can be done adding 2 bits of digital noise to the database signals.

Although the results obtained for the detector are very good, it has to be taken into account that they were obtained on 18 records, therefore extending these tests to more records and real signals, is recommended.

As seen, the dsPIC added no processing power limitations to the project and since the code was already thought to be as fast and simple as possible, could be interesting to port the work from the dsPIC to an ultra-low power microcontroller.

It would also be a good improvement to put the device to work with a battery, to increase its portability.

The most important development that can be done for this project is to connect it to a sensor (as mentioned above), in order to give atrial fibrillation alarms in real-time.

In a further development the platform can include a wireless module – GSM, for example – to send the alarms automatically to a clinical center in case of urgency.

Appendices

Appendix A – qrs.c

Source code of the QRS detector used.

```

// Time interval constants.
#define MS80      16
#define MS95      19
#define MS150     30
#define MS200     40
#define MS360     72
#define MS450     90
#define MS1000    200
#define MS1500    300
#define WINDOW_WIDTH  MS80
#define FILTER_DELAY  21 + MS200

// Global values for QRS detector.
unsigned int Q0 = 0, Q1 = 0, Q2 = 0, Q3 = 0, Q4 = 0, Q5 = 0, Q6 = 0, Q7 = 0 ;
unsigned int N0 = 0, N1 = 0, N2 = 0, N3 = 0, N4 = 0, N5 = 0, N6 = 0, N7 = 0 ;
unsigned int RR0=0, RR1=0, RR2=0, RR3=0, RR4=0, RR5=0, RR6=0, RR7=0 ;
unsigned int QSum = 0, NSum = 0, RRSum = 0 ;
unsigned int det_thresh, sbcount ;
unsigned int tempQSum, tempNSum, tempRRSum ;
unsigned int QN0=0, QN1=0 ;
int Reg0=0 ;

/*****
* QRSdet takes 16-bit ECG samples (5 uV/LSB) as input and returns the
* detection delay when a QRS is detected.  Passing a nonzero value for init
* resets the QRS detector.
*****/
int QRSdet(int x, int init)
{
    static unsigned int tempPeak, initMax ;
    static unsigned char preBlankCnt=0, qpkcnt=0, initBlank=0 ;
    static unsigned int count, sbpeak, sbloc ;
    unsigned int QrsDelay = 0 ;
    unsigned int temp0, temp1 ;
    int outVal;

    if(init)
    {
        hpfilt(0,1) ;
        lpfilt(0,1) ;
        deriv1(0,1) ;
        mwwint(0,1) ;
        Peak(0,1) ;
        qpkcnt = count = sbpeak = 0 ;
        QSum = NSum = 0 ;
    }

```

```

RRSum = MS1000<<3 ;
RR0=RR1=RR2=RR3=RR4=RR5=RR6=RR7=MS1000 ;

Q0=Q1=Q2=Q3=Q4=Q5=Q6=Q7=0 ;
N0=N1=N2=N3=N4=N5=N6=N7=0 ;
NSum = 0 ;

return(0) ;
}

x = lpfilt(x,0) ;
x = hpfilt(x,0) ;
x = deriv1(x,0) ;
if(x < 0) x = -x ;
x = mwint(x,0) ;
x = Peak(x,0) ;

// Hold any peak that is detected for 200 ms
// in case a bigger one comes along. There
// can only be one QRS complex in any 200 ms window.
if(!x && !preBlankCnt)
    x = 0 ;
else if(!x && preBlankCnt) // If we have held onto a peak for
    { // 200 ms pass it on for evaluation.
        if(--preBlankCnt == 0)
            x = tempPeak ;
        else x = 0 ;
    }
else if(x && !preBlankCnt) // If there has been no peak for 200 ms
    { // save this one and start counting.
        tempPeak = x ;
        preBlankCnt = MS200 ;
        x = 0 ;
    }
else if(x) // If we were holding a peak, but
    { // this ones bigger, save it and
        if(x > tempPeak) // start counting to 200 ms again.
            {
                tempPeak = x ;
                preBlankCnt = MS200 ;
                x = 0 ;
            }
        else if(--preBlankCnt == 0)
            x = tempPeak ;
        else x = 0 ;
    }
}

```

```

// Initialize the qrs peak buffer with the first eight
// local maximum peaks detected.
if( qpkcnt < 8 )
    {
    ++count ;
    if(x > 0) count = WINDOW_WIDTH ;
    if(++initBlank == MS1000)
        {
        initBlank = 0 ;
        UpdateQ(initMax) ;
        initMax = 0 ;
        ++qpkcnt ;
        if(qpkcnt == 8)
            {
            RRSum = MS1000<<3 ;
            RR0=RR1=RR2=RR3=RR4=RR5=RR6=RR7=MS1000 ;
            sbcount = MS1500+MS150 ;
            }
        }
    if( x > initMax )
        initMax = x ;
    }
else
    {
    ++count ;

    // Check if peak is above detection threshold.
    if(x > det_thresh)
        {
        UpdateQ(x) ;
        // Update RR Interval estimate and search back limit
        UpdateRR(count-WINDOW_WIDTH) ;
        count=WINDOW_WIDTH ;
        sbpeak = 0 ;
        QrsDelay = WINDOW_WIDTH+FILTER_DELAY ;
        }

    // If a peak is below the detection threshold.
    else if(x != 0)
        {
        UpdateN(x) ;
        QN1=QN0 ;
        QN0=count ;
        if((x > sbpeak) && ((count-WINDOW_WIDTH) >= MS360))
            {

```

```

        sbpeak = x ;
        sbloc = count-WINDOW_WIDTH ;
    }
}

// Test for search back condition. If a QRS is found in
// search back update the QRS buffer and det_thresh.
if((count > sbcount) && (sbpeak > (det_thresh >> 1)))
{
    UpdateQ(sbpeak) ;

    // Update RR Interval estimate and search back limit
    UpdateRR(sbloc) ;
    QrsDelay = count - count - sbloc;
    QrsDelay += FILTER_DELAY ;
    sbpeak = 0 ;
}
}

outVal = 0;
if (QrsDelay != 0)
    outVal = x;

return(outVal) ;
}
/*****
* UpdateQ takes a new QRS peak value and updates the QRS mean estimate
* and detection threshold.
*****/
void UpdateQ(unsigned int newQ)
{
    QSum -= Q7 ;
    Q7=Q6; Q6=Q5; Q5=Q4; Q4=Q3; Q3=Q2; Q2=Q1; Q1=Q0;
    Q0=newQ ;
    QSum += Q0 ;
    det_thresh = QSum-NSum ;
    det_thresh = NSum + (det_thresh>>1) - (det_thresh>>3) ;
    det_thresh >>= 3 ;
}
/*****
* UpdateN takes a new noise peak value and updates the noise mean estimate
* and detection threshold.
*****/
void UpdateN(unsigned int newN)
{
    NSum -= N7 ;
    N7=N6; N6=N5; N5=N4; N4=N3; N3=N2; N2=N1; N1=N0; N0=newN ;

```

```

NSum += N0 ;

det_thresh = QSum-NSum ;
det_thresh = NSum + (det_thresh>>1) - (det_thresh>>3) ;

det_thresh >>= 3 ;
}
/*****
* UpdateRR takes a new RR value and updates the RR mean estimate
*****/
void UpdateRR(unsigned int newRR)
{
    RRSum -= RR7 ;
    RR7=RR6; RR6=RR5; RR5=RR4; RR4=RR3; RR3=RR2; RR2=RR1; RR1=RR0 ;
    RR0=newRR ;
    RRSum += RR0 ;

    sbcount=RRSum+(RRSum>>1) ;
    sbcount >>= 3 ;
    sbcount += WINDOW_WIDTH ;
}
/*****
* lpfilt() implements the digital filter represented by the difference
* equation:
*  $y[n] = 2*y[n-1] - y[n-2] + x[n] - 2*x[n-5] + x[n-10]$ 
* Note that the filter delay is five samples.
*****/
int lpfilt( int datum ,int init)
{
    static int y1 = 0, y2 = 0 ;
    static int d0,d1,d2,d3,d4,d5,d6,d7,d8,d9 ;
    int y0 ;
    int output ;

    if(init)
    {
        d0=d1=d2=d3=d4=d5=d6=d7=d8=d9=0 ;
        y1 = y2 = 0 ;
    }

    y0 = (y1 << 1) - y2 + datum - (d4<<1) + d9 ;
    y2 = y1 ;
    y1 = y0 ;
    if(y0 >= 0) output = y0 >> 5 ;
    else output = (y0 >> 5) | 0xF800 ;
}

```

```

    d9=d8 ;
    d8=d7 ;
    d7=d6 ;
    d6=d5 ;
    d5=d4 ;
    d4=d3 ;
    d3=d2 ;
    d2=d1 ;
    d1=d0 ;
    d0=datum ;

    return(output) ;
}

/*****
* hpfilt() implements the high pass filter represented by the following
* difference equation:
*    $y[n] = y[n-1] + x[n] - x[n-32]$ 
*    $z[n] = x[n-16] - y[n]$  ;
* Note that the filter delay is 15.5 samples
*****/

#define HPBUFFER_LGTH 32
int hpfilt( int datum, int init )
{
    static int y=0 ;
    static int data[HPBUFFER_LGTH] ;
    static int ptr = 0 ;
    int z ;
    int halfPtr ;
    if(init)
    {
        for(ptr = 0; ptr < HPBUFFER_LGTH; ++ptr)
            data[ptr] = 0 ;

        ptr = 0 ;
        y = 0 ;
        return(0) ;
    }
    y += datum - data[ptr];
    halfPtr = ptr-(HPBUFFER_LGTH/2) ;
    halfPtr &= 0x1F ;
    z = data[halfPtr] ;           // Compensate for CCS shift bug.
    if(y >= 0) z -= (y>>5) ;
    else z -= (y>>5)|0xF800 ;
    data[ptr] = datum ;
    ptr = (ptr+1) & 0x1F ;
    return( z ) ;
}

```

```

    }
/*****
* deriv1 and deriv2 implement derivative approximations represented by
* the difference equation:
*    $y[n] = 2*x[n] + x[n-1] - x[n-3] - 2*x[n-4]$ 
* The filter has a delay of 2.
*****/
int deriv1( int x0, int init )
{
    static int x1, x2, x3, x4 ;
    int output;
    if(init)
        x1 = x2 = x3 = x4 = 0 ;
    output = x1-x3 ;
    if(output < 0) output = (output>>1) | 0x8000 ;    // Compensate for shift bug.
    else output >>= 1 ;
    output += (x0-x4) ;
    if(output < 0) output = (output>>1) | 0x8000 ;
    else output >>= 1
    x4 = x3 ;
    x3 = x2 ;
    x2 = x1 ;
    x1 = x0 ;
    return(output);
}
/*****
* mvwint() implements a moving window integrator, averaging
* the signal values over the last 16
*****/
unsigned int mvwint(int datum, int init)
{
    static unsigned int sum = 0 ;
    static unsigned int d0,d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15 ;

    if(init)
        {
            d0=d1=d2=d3=d4=d5=d6=d7=d8=d9=d10=d11=d12=d13=d14=d15=0 ;
            sum = 0 ;
        }
    sum -= d15 ;

    d15=d14 ;
    d14=d13 ;
    d13=d12 ;
    d12=d11 ;
    d11=d10 ;

```

```

    d10=d9 ;
    d9=d8 ;
    d8=d7 ;
    d7=d6 ;
    d6=d5 ;
    d5=d4 ;
    d4=d3 ;
    d3=d2 ;
    d2=d1 ;
    d1=d0 ;
    if(datum >= 0x0400) d0 = 0x03ff ;
    else d0 = (datum>>2) ;
    sum += d0 ;
    return(sum>>2) ;
}
/*****
* peak() takes a datum as input and returns a peak height
* when the signal returns to half its peak height, or it has been
* 95 ms since the peak height was detected.
*****/
int Peak( int datum, int init )
{
    static int max = 0, lastDatum ;
    static int timeSinceMax = 0 ;
    int pk = 0 ;
    if(init)
    {
        max = 0 ;
        timeSinceMax = 0 ;
        return(0) ;
    }
    if(timeSinceMax > 0)
        ++timeSinceMax ;
    if((datum > lastDatum) && (datum > max))
    {
        max = datum ;
        if(max > 2)
            timeSinceMax = 1 ;
    }
    else if(datum < (max >> 1))
    {
        pk = max ;
        max = 0 ;
        timeSinceMax = 0 ;
    }
    else if(timeSinceMax > MS95)

```

```
    {  
      pk = max ;  
      max = 0 ;  
      timeSinceMax = 0 ;  
    }  
lastDatum = datum ;  
return(pk) ;  
}
```

Appendix B – alarmDet.c

Source code of the algorithm used to detect anomalies on the heart beating.

```

// Time interval constants
#define MAXBEATS          110          // 110 max number of beat in 30 seconds
#define OBSPERIOD        30          //obs period = number of seconds it will take until calculate heartrate and variation,
                                     //maximum 30 sec, non overlapping.

//global variables
unsigned int nRR = 0;    // number of RR in the interval
unsigned int avarage = 0;
unsigned int rvec[MAXBEATS];

unsigned int alarmDet(unsigned int y)
{
    float aux, variat;
    unsigned int i, detection, j, nVar, period, HeartRate;
    detection = 0;          // no ALARM is detected
    if (y != 0)    //RR interval is received
    {
        rvec[nRR] = y;
        avarage += y;
        nRR ++;
    }
    period = OBSPERIOD * 200;    //number of samples in obsperiod. with 200 Hz
    nSamples ++;
    if (nSamples == period)
    {
        // Heart rate!
        avarage = avarage / nRR;
        //Avarage in samples in y. transforms in sec
        HeartRate = (60*200)/ avarage;

        /* VARIATION Check between RR intervals (in 30 seconds) */
        //puting array in crescent order, to analyse variation
        for (i = 0; i < nRR; i++)
        {
            for (j=i; j<= nRR; j++)
            {
                if (rvec[i] > rvec[j])
                {
                    aux = rvec[j];
                    rvec[j] = rvec[i];
                    rvec[i] = aux;
                }
            }
        }
        //comparing intervals
        nVar = 0;          // number of variations
        if(rvec[0] == 0) rvec[0]= rvec[1];
    }
}

```

```

variat = diff(rrvec[nRR], rrvec[0]); // if the difference between the
// biggest and the smallest value is
// not >20%, the other intervals are also not
if (variat > 20) //difference bigger than 20 %
{
    nVar = 1;
    //comparing with the bigger as reference
    if (diff(rrvec[nRR], rrvec[1]) > 20)
    {
        nVar = 2;
        if (diff(rrvec[nRR], rrvec[2]) > 20)
        {
            nVar = 3;
            if (diff(rrvec[nRR], rrvec[3]) > 20)
            {
                nVar = 4;
                if (diff(rrvec[nRR], rrvec[4]) > 20)
                    nVar = 5;
            }
        }
    }
}

//comparing with the smaller as reference. if variations > 4 there's no point in continue.
if ((diff(rrvec[nRR-1], rrvec[0]) > 20) && (nVar < 5))
{
    nVar++;
    if ((diff(rrvec[nRR-2], rrvec[0]) > 20) && (nVar < 5))
    {
        nVar++;
        if ((diff(rrvec[nRR-3], rrvec[0]) > 20) && (nVar < 5))
        {
            nVar++;
            if ((diff(rrvec[nRR-4], rrvec[0]) > 20) && (nVar < 5))
                nVar++;
        }
    }
}

}

/*CONCLUSIONS*/
if ((HeartRate < 40) || (HeartRate > 100))
    detection = 1; // heart rate ALARM
if (nVar > 4)
    detection = 2; // Atrial fibrillation detection ALARM
if (((HeartRate < 40) || (HeartRate > 100)) && (nVar > 4))
    detection = 3; // both produce an alarm
if (((HeartRate > 40) || (HeartRate < 100)) && (nVar < 4))

```

```
        detection = 10;
    nSamples = 0;
    nRR = 0;
}
return detection;
}
float diff (unsigned int big, unsigned int small)
{
    float dif;
    dif = big - small;
    dif = dif / big;
    dif = dif * 100;
    return dif;
}
```

Appendix C – How to Run the Program on the PC

1 – Installing software:

cygwin (version 1.5.25-15) with libcurl and WFDB libraries. Installation steps are detailed in [28].

2 – Inside the directory a copy of `qrsSensitivity.tar` should be saved.

3 – Unpacking the files by typing on the command line:

```
tar xfvz qrsSensitivity.tar.gz
```

This creates the directories 'median_mean', 'square' and 'absolute'. Each one containing the source codes necessary to test its behavior.

'median_mean' have two different detectors implemented, one using the mean of the QRS and Noise values to calculate the detection threshold, and the other using the median. 'square' and 'absolute' have similar detectors but in 'square' the signal is squared after the filtration process while in absolute the signal takes its numerical value without regard to its sign, both calculate the detection threshold recurring to the mean.

4 – Entering the directory (for example '*median_mean*') by typing:

```
cd median_mean
```

5 – Compiling the sources by typing:

```
Make
```

There should be no compilation errors or warnings.

6 – Running the software by typing:

```
./easytest            [or './easytest2']  
./bxbep
```

'easytest2' is only used inside 'median_mean', and it's a version of 'easytest' that performs the detection recurring to the mean instead of the median used in 'easytest'.

The output of 'easytest' is a set of annotation files with the suffix (annotator name) 'atest', one for each record processed.

The output of 'bxbep' is a pair of files (adtstat.txt and testrpt.txt). In 'testrpt.txt' the results of sensitivity and positive predictive can be seen.

All of the output files are written to the current directory.

7 – The steps 4 to 6 are repeated for 'square' and 'absolute'.

The records used are present inside each one of the folders mentioned, in the folder 'mitdb'; if other records need to be analyzed they must be mentioned in the file 'inputs.h', which makes available the use of records present in the 'mitdb' folder, but also the use of MIT-BIH Arrhythmia Database files directly from PhysioNet, which can be slower since it is reading directly from the PhysioNet master server at MIT.

In these tests the sampling frequency is 200Hz, but it's possible to change it. In 'qrsdet.h', variable `SAMPLE_RATE` is defined 200, changing this value changes the sampling frequency for the entire program.

Recompiling should be made after any change.

Appendix D – How to Run the Program on the board.

1 – Installing the software:

First unpack the files inside 'software.rar'. 'EE_143.zip' should be the first one to be installed; at the end of this step RT-Druid should be installed. Run the application 'MPLAB_C30_v3_02-StudentEdition.exe' and after 'Install_MPLAB_v8.exe'.

2 – Configuring RT-Druid:

Open RT-Druid and configure it to recognize microchip C30 compiler and the MPLAB ASM 30 assembler, as follow:

"Window" -> "Preferences" -> "RT-druid" -> "Oil" -> "PIC30" ->...

...->"Gcc path" refers to the installation directory of microchip C30 Compiler;

...->"Asm path" refers to the installation of ASM30 assembler provided with MPLAB IDE;

The default location is often:

C:\Program Files\Microchip\MPLAB C30

And

C:\Program Files\Microchip\MPLAB ASM30 Suite

3 – Configuring MPLAB IDE:

"Configure" -> "Select Device..." -> "Device"

And select 'dsPIC33FJ256MC710';

"Programmer" -> "Select Programmer"

And select '2 MPLAB ICD 2'.

(Attention before this configuration is done, the programmer – MPLAB ICD 2 – should be already connecting the FLEX Full base board and the computer. Also, the board should be powered on)

4 – Uploading the compiled program to the dsPIC:

After compilation on RT-Druid, a file named 'pic30.cof' is created. Open MPLAB IDE:

"File" -> "Import"

And choose the file mentioned (pic30.cof), and then select:

"Programmer" -> "Program"

5 – Running the program:

Once the program is uploaded, it can be executed, but first open one of the terminals supplied by the cd in "software.rar" (Terminal or Putty) or any other terminal, configure the serial port connected to the RS232 module. When ready, on MPLAB IDE:

"Programmer" -> "Release from Reset"

The program starts running.

Appendix E – CD

This CD includes the following files:

- **ADCinput.rar**: containing the files necessary to run the program on the board with the input provided by an analogical sensor (although it hasn't been tested, all configurations were done for a voltage between 0 and 3 Volts).
- **qrsSensitivity.tar**: this contains the tests that are run to test sensitivity and positive predictive (5.2.1.1 Sensitivity and Positive Predictive Test).
- **QRSboard.rar** and **QRSpC.rar**: containing the programs that produce the results on 5.2.2 Performance, Delays and Errors and on 5.2 Test Atrial Fibrillation Algorithm.
- **Software.rar**: with a list of programs necessary to run all the programs above.
- **MATLAB.rar**: all graphs and tests analysis were made resorting to MATLAB® help. The files used for that analysis are made available in this file.

Bibliography

- [1] F. INSTITUTE. (September 2009). Available:
<http://www.fi.edu/learn/heart/systems/circulation.html>
- [2] (September 2009) Available:
<http://www.fi.edu/learn/heart/development/development.html>
- [3] F. INSTITUTE. (September 2009). Available:
<http://www.fi.edu/learn/heart/development/development.html>
- [4] M. K. Tanaka H, Seals DR, "Age-predicted maximal heart rate revisited.," J Am Coll Cardiol., pp. 37: 153-156, 2001.
- [5] HEARTSITE. (2009, September). Available:
http://www.heartsite.com/html/electrical_activity.html
- [6] A. H. Association. (2009, September). Available:
<http://www.americanheart.org/presenter.jhtml?identifier=34>
- [7] S. L. s.-R. H. center. (2009 September). Available:
http://www.arrythmia.org/all_about_rhythms.html
- [8] G. V. N. Peter Kowey, in ATRIAL FIBRILLATION, ed.
- [9] A. J. C. Johan Waktare, in ATRIAL FIBRILATION, ed.
- [10] PhysioNet. (2009, September). Available:
<http://www.physionet.org/cgi-bin/ATM>
- [11] C. s. H. Specialists. (2009, September). Available:
<http://mykentuckyheart.com/services/ECG.htm>
- [12] P. Richard E. Klabunde. (2009, September). Cardiovascular Physiology Concepts. Available:
<http://www.cvphysiology.com/Arrhythmias/A013c.htm>
- [13] M. S. Thaler, in The Only EKG Book You'll Ever Need, paperback: fifth edition ed: Lippincott Williams and Wilkins, 2009.
- [14] (2009, September). Available:
http://library.med.utah.edu/kw/ecg/ecg_outline/Lesson1/lead_dia.html

- [15] N. E. H. Tomas B. Garcia, in Introduction to 12-lead ECG: the art of interpretation, ed.
- [16] F. Johnson. (2009, September). 12 lead ECG. Available: http://academic.cuesta.edu/fjohnson/PowerPoint_PDF/12leadecg.pdf
- [17] M. Homobono Calleja, "Willem Einthoven and the Electrocardiogram," presented at the Sukaman Lecture, 14th ASEAN Congress of Cardiology, Kuala Lumpur, Malaysia, 2002.
- [18] T. H. INSTITUTE. Available: <http://www.texasheart.org/HIC/Topics/Diag/diholt.cfm>
- [19] T. W. J. PAN J. A Real-Time QRS Detection Algorithm. IEEE Trans. Biomed. Eng. 230-236.
- [20] R. M. Rangayyan, in Biomedical Signal Analysis : A Case-Study Approach, ed: IEEE Press Series on Biomedical Engineering.
- [21] W. J. T. P.S. Hamilton. (1986) Quantitative investigation of QRS detection rules using the MIT/BIH arrhythmia database. IEEE Trans. Biomed Eng. 1157-1165.
- [22] H. H. R. H.G. Goovaerts, T.J. Vanden Akker, H.Schneider. A digital QRS detector based on the principle of contour limiting. IEEE Trans. Biomed. Eng. 154, 1976.
- [23] P. S. Hamilton. (October 2009). Open Source ECG Analysis Software Documentation. Available: <http://www.eplimited.com/osea13.pdf>
- [24] W. J. T. J. L. Urrusti, "Performance evaluation of an ECG QRS complex detection algorithm," in Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 1993, pp. 800–801.
- [25] EVIDENCE. Available: http://www.evidence.eu.com/download/manuals/pdf/flex_refman_1_0_2.pdf
- [26] "Data Sheet dsPIC33FJXXXGPX06/X08/X10," ed.
- [27] (2009, October). Available: <http://www.bmj.com/cgi/content/full/309/6947/102>
- [28] (2009, October). Intalation Guide. Available: <http://www.physionet.org/physiotools/wfdb-windows-quick-start.shtml>