



Design and Implementation of a Telemedicine System.

Case study: Atrial Fibrillation detection using Machine Learning Technique

By the student : Khalil BEN KALBOUSSI

supervised by

Prof. Lazhar Khriji

Dr. Farah Barika Ktata

Jury's president : Kais BOUALLEGUE

Report examiner : Salma BELGACEM

A Thesis in the Field of Electronics and Computer Science

for the Degree of Master in Intelligent Pervasive Systems

Higher Institute of Applied Sciences and Technologies Sousse, Tunisia

Sultan Qaboos University Muscat, Oman

October 2019

Dedication

I dedicate my dissertation work to my family, my friends and to the soul of our lost beloveds. A special feeling of gratitude to my loving parents, Fethi Kalboussi and Zannouba Belhaiza whose words of encouragement and push for tenacity ring in my ears. My sister and my brother Khawla and Kheireddine, have never left my side. I also dedicate this dissertation to my many friends who have supported me throughout the process specially. I will always appreciate all they have done, especially Oussama Ammar and Ali Al Burtumani for helping me like brothers.

I dedicate this work to the souls of my dear uncle who passed away during the research, may the lord rest his soul, and my beloved friends, Mustafa and Dalel, who have been lost in a blink, but never forgotten.

Acknowledgments

I would like to appreciate all those who provided any contributions in completing this project. I would like to give special thanks to my supervisor Prof. Lazhar Khriji, for his great effort and helpful guidance to complete my project successfully. Furthermore, I would like to acknowledge Prof. Abdulnasir Hussein, Director of Communication and Information Research Center in SQU for his support and helpful suggestions and for giving me all facilities in the center. Dr. Farah Barika Ktata, my Supervisor for the help and the patience. Special thanks and acknowledgment go to the members of the team, Mr. Iftikharullah and Mr. Abbas for their continued support and important contribution in the project. I wish to specially thank and acknowledge the important amount of guidance, help and contribution provided by Mr. Nabil Hamza who has never hesitated to help.

Special thanks go to Sultan Qaboos University, Communication and information Research Center as well as the College of Engineering, Department of Electrical and Computer Engineering for allowing me to conduct my research with them and for the help in providing the necessary tools, equipment, and all licensed software.

Table of Contents

Dedication	iii
Acknowledgments.....	iv
List of Tables	vii
List of Figures	viii
<u>Chapter 1.</u> <u>General Introduction</u>	1
1.1. Introduction	1
1.2. Thesis organization	3
<u>Chapter 2.</u> <u>State of the art</u>	4
2.1 Context	4
2.1.1 Sleep apnea	4
2.1.2 Cardiovascular diseases	5
2.1.3 ECG analysis	7
2.1.4 Telemedicine	7
2.1.5 Wireless Body Area Networks (WBANs)	8
2.2 Related works	9
2.2.1 platform	9
2.2.2 Machine Learning Model	10
<u>Chapter 3.</u> <u>Contribution</u>	16
3.1 System architecture version 1	16
3.1.1 ZigBee module	18

3.1.2	Processing framework	20
3.1.3	User web interface.....	23
3.1.4	Conclusion	25
3.2	System architecture version 2	25
3.2.1	MySignals Platform	25
a-	Electronic analysis	26
b-	Sensors	27
3.2.2	Conclusion	28
3.2.3	Overall architecture description.....	29
3.3	System architecture version 3 (final).....	30
3.3.1	ESP32 Card	30
3.3.2	Sensors	31
3.3.3	Grafana Framework.....	36
3.3.4	Conclusion	40
3.3.5	Overall architecture description.....	42
3.4	Machine Learning Model	43
3.4.1	Data description	43
3.4.2	Feature extraction methods	44
3.4.3	Tests and results.....	49
<u>Chapter 4.</u>	<u>Experimental results</u>	66
4.1	Platform	66
4.2	Machine Learning Model	67
<u>Chapter 5.</u>	<u>Conclusion and perspectives</u>	73

List of Tables

Table 2.1: literature review-comparative study	15
Table 3.1: XBee packet.....	19
Table 3.2: Mysignals supported Sensors	27
Table 3.3: BLE Sensors profile examples.....	33
Table 3.4: R-R intervals table	49
Table 4.1: Comparative study between the platform versions.....	64
Table 4.2: Test Results.....	71

List of Figures

Figure 3.1: System architecture (Data Path) illustration.....	17
Figure 3.2: Processing program	21
Figure 3.3: Video Frames Grouping	22
Figure 3.4: graph display example.....	24
Figure 3.5: System architecture	24
Figure 3.6: MySignals platform and sensors	26
Figure 3.7: Real Test.....	29
Figure 3.8: system architecture V2	30
Figure 3.9: BLE architecture.....	31
Figure 3.10: BLE architecture.....	32
Figure 3.11: nRf connect MAC @ of the sensor	32
Figure 3.12: nRf connect UUID of the service and characteristic	32
Figure 3.13: Respiration force sensor	34
Figure 3.14: ESP32 ADC attenuation.....	35
Figure 3.15: Accelerometer's values	36
Figure 3.16: Graph example	37
Figure 3.17: Grafana Framework.....	38
Figure 3.18: Grafana data source setup.....	39
Figure 3.19:Memory partitioning of ESP32 card	41
Figure 3.20: System architecture V3	42

Figure 3.21: ECG signal example.....	44
Figure 3.22: Discrete Fourier Transform illustration.....	45
Figure 3.23: 1D 3 level DWT illustration.....	47
Figure 3.24: 2D DWT illustration.....	47
Figure 3.25: PCA application	50
Figure 3.26: Peak detection error example	51
Figure 3.27: Applying a Butterworth filter on Noisy signals (a) sample of noise in ECG signals, (b) signal sample after filtering.....	52
Figure 3.28: Applying a Normal Distribution & Standard Deviation filtering technique (a) ECG signal before filtering, (b) ECG signal after filtering	53
Figure 3.29: method's flowshart	55
Figure 3.30: Heart Rate variability between (a) patient 1 and (b) patient 2	56
Figure 3.31: Non-uniform R-R interval vectors (a) Resulting Data-Frame, (b) Non- uniform Length – Missing Data.....	57
Figure 3.32: Data Preprocessing (2nd Phase) (a) Amputation,(b) Adjusted Data-Frame	58
Figure 3.33: Real case data-frame After amputation and filling	58
Figure 3.34: R-R interval with Added values	59
Figure 3.35: Periodic B-Spline Extrapolation (2nd order, over 8 samples).....	60
Figure 3.36: Periodic Cubic Spline Extrapolation (over 5 samples)	60
Figure 3.37: Periodic Cubic Spline Extrapolation (over 8 samples)	61
Figure 3.38: Window Periodic Cubic Spline Extrapolation (over 8 samples).....	62
Figure 3.39: Neuron Cell - CNN equation.....	63
Figure 3.40: Proposed CNN's architecture	64

Chapter 1. General Introduction

1.1. Introduction

“For more than a decade, cardiovascular disease has been the number one killer of Omanis. More people die from cardiovascular disease than diabetes and cancer combined, and it is all preventable, according to officials.

Data from 2012 indicated an 18 per cent chance that adults will contract one non-communicable disease in their lifetime, such as cardiovascular disease. Experts, such as Dr. Ahmed Al-Busaidi, Secretary for the National Non-Communicable Committee, does not see a decrease in this morbidity rate anytime soon.

“I don’t think the data from 2016 will be much different than in the past years. It is a trend from the past five years. Within the last ten years, there is still a high rate of deaths, and that is the problem with non-communicable diseases,” Dr Al Busaidi said.” [1]

Cardiovascular diseases, such as Atrial Fibrillation, are one of the leading death causes in Oman. The main factors that prevents Omanis from getting medical assistance; early disease diagnosis or rescue and reanimation, are the long distances to the hospitals and the lack of cardiology doctors. As most of the hospitals are in the cities, the geographical distance between cities themselves and between cities and countrysides is an important and critical factor in this issue. The distance doesn’t only prevent patients from getting medical rescue, but also affects the results of the diagnosis tests; as the

patient has to travel a long distance in order to arrive to a specialized hospital, the physical condition relatively changes and the tests are questioned to be accurate.

Julio Marti-Almor has proved that Sleep apnea is a disease that is strongly related to cardiovascular diseases. “A higher incidence of significant AF was reported in patients with severe SA than in patients with non-severe SA” [2]

The ministry of health (Sultan Qaboos University Hospital - sleep apnea laboratory) in coordination with **Sultan Qaboos University** (communication and information research center and college of engineering) and **OmanTel**, have launched a project to limit the deaths caused by cardiovascular diseases and sleep apnea. The objective is to use technology to eliminate the distance factor, and make it easier for the patient to be checked or to be rescued and revived without having to travel hundreds of kilometers and without the need to even see the doctor.

Our contribution in this project is illustrated in three different levels:

1) Hardware design and implementation of an electronic platform,

The electronic platform is responsible of collecting the vital signals from a distant patient and immediately store it in a distant database.

2) Setup of a webserver,

Responsible of storing the data (database) and running a graphical web interface to illustrate the signals.

3) A novel Machine Learning Model,

Responsible of automatically detecting cardiovascular diseases and immediately alerting experts.

In this report, we will explain the different sections of the platform following the systematic order of the project. The electronic device is to be introduced at first, as it is the first part to activate in order for the system to function, secondly the setup of the web server, as it is mandatory to log the data, save it and plot it, then we'll explain the Machine Learning model.

1.2. Thesis organization

This thesis is organized as follow: Chapter 2 contains the state of art of this work; the main research papers this work is based on will be discussed along with an introduction of the big topics this project touches. Chapter 3 describes the contributions to the work and the evolution of the project. Experimental results and discussions are given in Chapter 4. Finally, Chapter 5 concludes this research and gives some suggestion for future work that can improve the project.

Chapter 2. State of the art

2.1 Context

2.1.1 Sleep apnea

Obstructive sleep apnea is a sleep disorder in which breathing is briefly and repeatedly interrupted during sleep. The "apnea" in sleep apnea refers to a breathing pause that lasts at least ten seconds. Obstructive sleep apnea occurs when the muscles in the back of the throat fail to keep the airway open, despite efforts to breathe. Another form of sleep apnea is central sleep apnea, in which the brain fails to properly control breathing during sleep. Obstructive sleep apnea is far more common than central sleep apnea.

Obstructive sleep apnea, or simply sleep apnea, can cause fragmented sleep and low blood oxygen levels. For people with sleep apnea, the combination of disturbed sleep and oxygen starvation may lead to hypertension, heart disease and mood and memory problems. Sleep apnea also increases the risk of drowsy driving.

Chronic snoring is a strong indicator of sleep apnea and should be evaluated by a health professional. Since people with sleep apnea tend to be sleep deprived, they may suffer from sleeplessness and a wide range of other symptoms such as difficulty concentrating, depression, irritability, sexual dysfunction, learning and memory difficulties, and falling asleep while at work, on the phone, or driving. Left untreated, symptoms of sleep apnea can include disturbed sleep, excessive sleepiness during the day, high blood pressure, heart attack, congestive heart failure, cardiac arrhythmia, stroke or depression. Sleep apnea diagnosis can be in two forms:

- **Nocturnal polysomnography:** During this test, the patient is hooked up to equipment that monitors his heart, lung and brain activity, breathing patterns, arm and leg movements, and blood oxygen levels while sleeping.
- **Home sleep tests:** The doctor might provide the patient with simplified tests to be used at home to diagnose sleep apnea. These tests usually measure the heart rate, blood oxygen level, airflow and breathing patterns.

Depending on the level of diagnosis the doctor decides which diagnosis method to use, three diagnosis levels exists:

- **1st level:** a technician attendance is required. A minimum of 7 channels of data (but typically ≥ 16), including respiratory, cardiovascular and neurologic parameters. A video is captured along with snoring sound recordings.
- **2nd level:** the same signals are captured except electromyogram (EMG), also without the need for a medical technician.
- **3rd level:** includes at least 3 channels of data (e.g., oximetry, airflow, heart rate) without the need for video streaming nor EMG

In [2] [3] [4] sleep apnea and cardiovascular diseases have been proven strongly related, one can cause the other. Generally, people with severe sleep apnea end up with strong cardiovascular diseases, while it is also one of the symptoms of the condition.

2.1.2 Cardiovascular diseases

Cardiovascular diseases are split into three types:

- **Circulatory:** High Blood Pressure and coronary artery disease (causing blockages in the pipes (arteries) that supply blood to the heart) are the main causes of blood vessel disorders. They can result in a stroke or heart attack,

which can be devastating. Fortunately, there are many preventative and treatment options.

- **Structural:** Heart muscle disease (cardiomyopathy) and congenital abnormalities (problems in the development of the heart and blood vessels which are present from birth) are two problems that can damage the heart muscle or valves.
- **Electrical:** Abnormal heart rhythms (arrhythmias) are caused by problems with the electrical system that regulates the steady heartbeat. The heart rate may be too slow or too fast; it may stay steady or become chaotic (irregular and disorganized). Some arrhythmias are very dangerous and cause sudden cardiac death, while others may be bothersome but not life threatening. [5]

As cited in the previous section, Sleep apnea and arrhythmia are strongly related.

Arrhythmia is divided into multiple categories, depending on which part of the heart that suffers a problem. In this work we will be interested in the **Atrial Fibrillation (AF or AFib)**, which is a disease that hits the Atria (upper chamber of the heart). More than 2 million people in the U.S. have atrial fibrillation. In AF, the heartbeat is irregular and rapid due to disorganized signals from the heart's electrical system. The upper chambers of the heart may beat as often as 300 - 400 times a minute, about four times faster than normal. Though AF isn't life threatening, it can lead to other rhythm problems, feeling tired all the time, and heart failure (with symptoms such as filling up with fluid, swelling of the hands, legs and feet, and shortness of breath). People with AF are five times more likely to have a stroke than people without the condition. Doctors often prescribe blood thinners (anticoagulants) to patients with AF to reduce this higher risk of stroke. [5]

2.1.3 ECG analysis

The advancement in technology has also offered the possibility to use the processing power of machines to automatically analyze long and complicated signals and detect abnormalities, specifically talking about electrocardiograms (ECG). A big amount of researches have been conducted in the field of automatically detecting cardiovascular diseases using mathematical procedures combined with machine learning algorithms. A second contribution has been proposed also in this field. ECG signals shows the plot of the electronic signals that reflects the activity of the heart and is used by physicians to diagnose and treat various cardiovascular diseases. The R peaks detection is a very important step in ECG signal processing. The methods used to detect R peaks in ECG signals are Pan Tompkins algorithm and wavelet transform based R peaks detection.

2.1.4 Telemedicine

The advancement in technology has offered the possibility for the patient to be monitored at home, and to detect in real time medical stress situations. Telemedicine is becoming a major improvement on patients' diagnosis procedures, especially for detecting life threatening health problems in real time. In recent years, the big improvement in information and communication technologies, along with mobile Internet, offering access to the web anywhere and anytime, automatically improved modern healthcare solutions. In this context, mobile healthcare systems (m-Health) offers medical services, eliminating geographical, temporal and even organizational problems. The third level in the diagnosis procedure doesn't oblige the patient to be present in a testing room for the night, where he can be monitored and diagnosed while sleeping at home. multiple

research subjects in this matters have been conducted. Multiple solutions have been studied, an efficient and performant solution has been proposed.

As technology is now mixed with our daily life routine, it is becoming more and more calm and discrete. Mark Weiser and John Brown have used the term calm technology in [6] to describe the calm and smooth flow of information that don't need the full attention of the user yet it offers data in an easy way. Ubiquitous technology has advanced, not only offering information to users but also quietly collecting it. Micro sensors, wearable technology and mobile embedded systems have opened a door of possibilities for ubiquitous technology to evolve rapidly. Mardonova and al. in [7] and R. Seshadri in [8], have discussed and compared the use of smart watches, fitness trackers, wearable cameras and other wearable technologies. Furthermore, collecting the data wirelessly gives the user more freedom and space and make the technology around him invisible.

2.1.5 Wireless Body Area Networks (WBANs)

A wireless body area network (WBAN) is a type of network that connects independent sensors situated in the clothes, on the body or under the skin of a person. The range of this wireless network reaches over the whole human body and the sensors are connected through a wireless communication channel for more movement freedom and discretion. This technology is mainly applied in sports and activities evaluations and health monitoring systems. WBANs usually use low power systems like BLE in order to maintain high efficiency. The architecture of WBANs differs from one application to another, a typical application is to assign one master controller node, and a few slave nodes. The controller in this case is in charge of coordination between nodes and

gathering of the data. The coordinator node needs to have a sufficient processing power in order to be able to keep up with the amount of data coming thru. Another application is to set the nodes in mesh architecture. In this architecture all the nodes can be equal. A routing protocol can be applied to deliver the data from one sensor to the receiving device (mobile phone, computer, custom device, etc..).

2.2 Related works

2.2.1 platform

Ayaskanta Mishra and al. proposed an architecture for collecting, storing and displaying the heart rate values. They used an AD8232 ECG sensor to acquire and filter the signal. Then an ADS1115 low-power, 16-bit, high precision analog to digital converter was used to convert the analog signals coming from the sensor and send it to a raspberry pi 3 card. The card puts the data in a frame and sends it over to a Message Queuing Telemetry Transport (MQTT) cloud server using a MQTT protocol [9]. In a previous work Ayaskanta Mishra and al. proposed another architecture, similar to the previous yet less advanced. The same AD8232 sensor was used to detect heart beats. An Arduino Uno card was utilized to collect the signals from the sensor then transmit it to a cloud storage via an ESP8266 WIFI module [10]. A web interface was developed to make it possible to download the data in a csv file yet displaying the data is not possible. The work did not discuss power usage, nor the cost of the project. Also patient privacy rules are not respected.

Smartphones have been introduced also into telemedicine domain, as the performance of smartphones today competes with the performance of computers, they

have been used is a variety of researches in the field of telemedicine. Bhaskar N. and al. developed a system recording ECG pulse by connecting the mobile phone to a sensor, the acquired data is stored for analyzing by medical staff. [11]. The solution does not introduce any mean of plotting the data. R.Harini and al. have developed a platform based on Arduino Uno card and E-Health Shield that contains the necessary amplifiers and conditioning blocks for most of the sensors like ECG and Pulse/Oximeter sensors [12]. Although this solution is able of saving data to a local database, a remote saving is impossible with the absence of a WIFI module. An android application has been developed to view the graphs over Local Area Network. the graph shows the very low resolution of the signal, as the sampling frequency is pretty low. Another research conducted by A. Maradugu describes a mobile health monitoring system detecting SPO2, heart rate and temperature. The system is based on a controller circuit based on MSP430 micro-processor, that communicates with all the sensors and with a mobile phone. The data is uploaded to the web using the phone's internet connectivity [13].

2.2.2 Machine Learning Model

In order to automatically detect cardiovascular diseases an architecture has been proposed implementing a machine-learning algorithm along with a pre-processing stage. As it is important for the signals to be treated, to successfully extract features, and to be classified, a deep study has been conducted in this field. We will explain some of the related work in connection with both stages in this section.

Karol Antczak and al. have discussed in [14] the possibility of denoising ECG signals using recurrent neural networks. Denoising is a very important task that affects the precision of the algorithm. Fernando A. and al. have used Matlab tools to filter the

signal using a Butterworth filter then detect QRS complex. A total of 169 features were extracted based on classical time domain, frequency domain, non-linear HRV metric and morphological features such as P-wave power and QT-interval. [15]. A Residual Network (ResNet) have been implemented on the same dataset we have used. An F1 score of 79% was reached. The QRS detection algorithm developed by Pan, J. and Tompkins, W.J. in 1985 [16] is used in many research work like by [17] Essam H. and al. the ECG signals have been normalized using a mean of zero and standard deviation technique before passing it thru a band-pass filter then extracting features using FFT and DWT. Also morphological and time domain features were extracted. The method proposed in this work has been used to classify each beat of the heart apart. A Support Vector Machine classifier was used and an accuracy of 93%. Shenda H. and al. in [18] have introduced a signal splitting technique into QRS complexes then calculate statistical features like count, mean, maximum, minimum, range, variance, skewness, kurtosis, percentile and so on, frequency related features like power, frequency band power, Shannon entropy, SNR (Signal Noise Ratio) and so on and then medical features appropriate to the signal like heart rate variability, median absolute deviation, variation and density histograms, interval, duration, amplitude, location, slope and area and so on. A model based on Recurrent Neural Networks (RNN) and one Dimensional Neural Networks (1DCNN) has been proposed in order to automatically detect features from the signal via 1DCNN filters and classify the signals. An accuracy of 86,55% has been reached on MIT database. Muhammad Z. and al. [19] also injected the raw ECG signals into a neural network, after passing each ECG signal through a band pass filter for denoising, to make use of the

internal filters to automatically extract features. An accuracy of 92.7% has been reached on MIT database.

Because many of the previously cited work did not use the same dataset as we have, the number of classes is practically not the same yet we aim to discover the techniques and methods used in classification, feature extraction and signal filtering. In the next part on this section we will be interested in papers that have worked on the same dataset as ours, in order to be able to compare the results.

Joachim A. in [20] proposed a pre-processing method to extract features after detecting R-peaks. The RR interval time-series was first calculated then the signal quality was estimated on a second-by-second basis and the continuous sub-segment with the highest quality was selected for further analysis. A number of features were estimated: heart rate variability (time domain based, fragmentation, coefficient of sample entropy etc.), ECG morphology (QRS length, QT interval etc.) and the presence of ectopic beats. The features were used to train support an SVM. The included test metric is only F1, which reached 80%. In [21] Lucia B. and al. have used fifty features based on the ECG signal, derived from the RR series and obtained combining QRS morphology and rhythm. After applying a stepwise linear discriminant analysis, only thirty features are used in LSVM classifier. The F1 score reached 81%. Guangyu B. proposed a model in [22] including a method for extracting thirty features based on AF Features, ECG Morphology, RR intervals, Similarity of QRS, Similarity of R amplitude, Ratio of high similarity, and Signal Qualify. A decision tree based classifier model were utilized in the work to obtain an $F1 = 86\%$. Another work conducted by Pietro B. [23] includes a two stages RUSBoost model to classify the signals. The first stage aims to distinguish

between noisy and non-noisy recordings. In the second stage, the recordings not classified as noisy are delineated and the AF features are extracted, and provided as inputs to an ensemble learning classifier. Before that a modified P&T algorithm is implemented to detect QRS peaks then a range of AF is extracted from the recording describing the spectral properties of the hearth rate variability (HRV), the ECG signal morphology, the complexity of both the ventricular and the atrial activity, and a variety of other atrial activity indices for irregularity and variability analysis. An F1 score = 0.75% is reached. In other works like [24] Chandra B. and al. did not extract features. First the signal is normalized to be in $[-1,1]$ then it is passed thru a filter to detect the base line, then that base line is subtracted from the original signal to smooth it. R peaks are detected to form templates then injected into a CNN. An overall F1 score of 71% is reached. Another work [25] by Ivaylo C. and al. used two techniques to eliminate noise and low amplitude in normal ECG signals 1) The signal is stepwise amplified until the detector starts to register QRS. 2) The noise immunity is improved by zeroing the signal during $\pm 300\text{ms}$ around the detected noise and then the QRS detector is restarted. Then HRV features are calculated from RR-Tachogram like mean value, median value, standard deviation, mean deviation, ratio of mean-to-median value, etc. and from dRR-Tachogram like proportion of RR intervals differing by $>50\text{ms}$ from the preceding RR interval, square root of the mean squared differences of successive RR intervals, etc. along with some other features like Average beat, P-waves and amplitudes. A Linear Discriminant Analysis (LDA) classifier has been implemented in this work to give an accuracy of 80%. Erin E. C. and al. [26] proposed a classic features extraction technique based on the ECG characteristics: Ventricular response features, Atrial activity features and Other ECG

features like Average spectral power, Variance of spectral power, Root mean square fluctuation of time series, Average total power of time series, etc. a decision tree is implemented with a SMOTE (Synthetic Minority Oversampling Technique) for increasing the sample size of the minority classes. F1 score of this model have reached 78,55%. Matthieu D. S. and al. in [27] also proposed a classic method beginning with filtering each ECG signal with cutoff frequencies of 1 and 50Hz. A Pan and Tompkins s QRS detector was then used RR intervals were extracted then features were extracted depending on the heart rate variability. Template-based features were also used along with signal quality features. After that an adaboost classifier was implemented to give an F1 score = 76%

Comparative study

In the table 2.3.1 we list the paper we have analyzed and considered as a base for our work. In the table we list the used filtering technique, the features extraction method, the used classifier, the obtained Accuracy and the F1 score which will be explained in chapter four section two. As some papers are based on the accuracy and others are based on the F1 score we were obliged to use both in table as double standard.

Table 2.1: literature review-comparative study

	Filtering technique	Features extraction	classifier	Acc	F1
Fernando A. and al [15]	Butterworth filter	Features calculation	ResNet		79%
Essam H. and al. [17]	mean of zero STD	FFT DWT Features calculation	SVM	93%	
Shenda H. and al [18]	DWT	QRS detection Feature calculation 1DCNN	RNN	86,5%	
Muhammad Z. and al. [19]	Filter	CNN	CNN	92.7%	
Joachim A. and al. [20]	Filter	R-peaks detection Feature calculation	SVM		80%
Lucia B. and al. [21]	Filter	QRS detection Feature calculation	LSVM		81%
Guangyu B. and al. [22]	Filter	Feature calculation	Decision tree		86%
Pietro B. and al. [23]	Filter	QRS detection Feature calculation	Decision tree		75%
Chandra B. and al. [24]	Normalizing Filter	R-peaks detection Feature calculation	CNN		71%
Ivaylo C. and al. [25]	Adaptive amp Noise detection	QRS detection Feature calculation	LDA		80%
Erin E. C. and al. [26]	Filter	Feature calculation	Decision tree + SMOTE		78,5%.
Matthieu D. S. and al. [27]	Filter	QRS detection Feature calculation	Decision tree		76%

Chapter 3. Contribution

In such a platform, some basic functionalities should be present. The most important task is to successfully collect the data, then to transmit it to the database. The architecture of such a system can vary in big way, a huge amount of solutions can be proposed, some of these solutions are better than others as they fit better to our demands. The characteristics that are needed in such a platform, and which we will evaluate the solution upon, are:

Energy consumption, Mobility, Wireless capabilities, Real time capabilities, Data usage, Security, Cost and the Options available.

3.1 System architecture version 1

In this chapter, we will explain the different versions of the electronic platforms we have built and tested. We will go through the explanation chronologically explaining the differences between them.

The main purpose of the platform is to collect the vital signals and send it to storage. It needs to satisfy certain criteria, speaking of energy consuming, cost, wireless capabilities, data and traffic usage, real time constraints, mobility and other criteria that will be tested in the three platforms.

The proposed architecture ensures a complete and reliable healthcare system. Different types of users use this system, like patient, doctor, and administrator. The first requirement of this system is that every user must have Internet connection and an account according to its role. To use this system every user connects to the server through the interface.

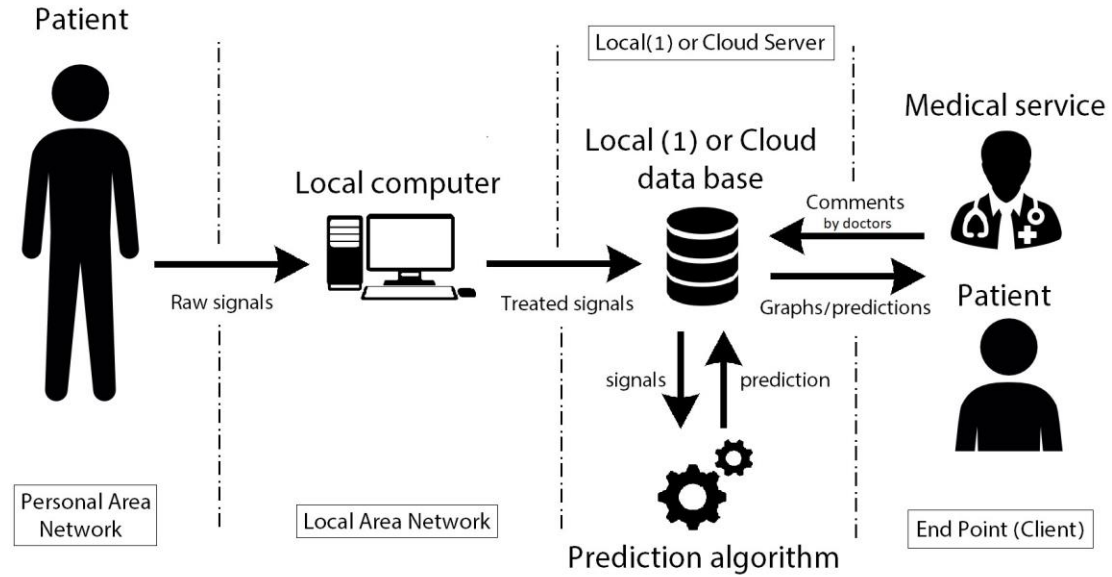


Figure 3.1: System architecture (Data Path) illustration

The architecture proposed has been studied and carefully tested, the idea is to take the data path as an axis (Figure 3.1), and build a modular architecture around it. Having a reliable electronic platform is the starting point of our work. The first node is the signal acquisition circuit. It is highly important to collect clean (not noisy) and real (right values with no amplitude shift) signals. These signals are passed to a local computer via Bluetooth. The second node is composed of two Zigbee Bluetooth modules; it is responsible of the transmission of the signals from the acquisition circuit to the computer. The choice of the modules has been studied carefully to achieve the best efficiency. In such a mobile system, a minimal power consumption is a priority. The modules we have used offers the possibility to choose the wanted range and power consumption. The data received on the computer is forwarded via Serial Port (USB) to an installed program, to process the packets received. This program is built with JAVA on "Processing" framework. Processing is an open source framework that offers a variety of libraries and

tools to work with hardware modules such as Bluetooth and Wi-Fi. After reading the packets, a secure connection is established with a database that can be either hosted on a cloud platform or on a local server located in the Sultan Qaboos University Hospital. If the data is stored on cloud servers, patients' privacy should be taken into consideration, and the geographic location of the cloud servers should be determined, so that the data is not stored outside the country borders. After storage, the data is accessible to the doctors and medical staff via the dedicated web interface. Authentication is required, and the website displays vital signals' graphs for the selected patient, and gives the possibility to comment and feedback on the patient's health status.

3.1.1 ZigBee module

ZigBee is an open global standard for wireless technology designed to use low-power digital radio signals for personal area networks manufactured by Digi International. It is a low-power circuit, which is useful in mobile and portable projects like ours. Xbee S2C offers some very useful options concerning the transmitting power and the current consumption. The user can choose, while programming the module, the level of transmitting and power consumption to use. The maximum transmitting current is 33 mA at 3.3 VDC and 45 mA in boost mode [28].

The module can handle multiple anthologies like star connection, mesh and peer to peer. The data sent from a module is encapsulated in a packet, which we explain its parts in the table below.

Table 3.1: XBee packet

Frame data fields	Offset	Description
Frame type	3	0x92
64-bit source address	4-11	MSB first, LSB last. The sender's 64-bit address.
16-bit Source network address	12-13	MSB first, LSB last. The sender's 16-bit address.
Receive options	14	Bit field: 0x01 = Packet acknowledged 0x02 = Packet is a broadcast packet Ignore all other bits
Number of samples	15	The number of sample sets included in the payload. Always set to 1.
Digital channel mask	16-17	Bitmask field that indicates which digital I/O lines on the remote have sampling enabled
Analog channel mask	18	Bitmask field that indicates which analog I/O lines on the remote have sampling enabled
Digital samples (if included)	19-20	If the sample set includes any digital I/O lines (Digital channel mask > 0), these two bytes contain samples for all enabled digital I/O lines. DIO lines that do not have sampling enabled return 0. Bits in these two bytes map the same as they do in the Digital channel mask field.
Analog sample	21-22	If the sample set includes any analog I/O lines (Analog channel mask > 0), each enabled analog input returns a 2-byte value indicating the A/D measurement of that input. Analog samples are ordered sequentially from ADO/DIO0 to AD3/DIO3.

We have set the connection type to peer-to-peer connection as, in this stage; we are just testing the connection reliability and feasibility. The Xbee Module is connected to a simple conditioning electronic card that acquires the temperature using an LM35 sensor. In addition, the battery level has been acquired as an analog signal using a simple two resistors voltage divider. The ZigBee modules are power efficient. In such a project, modules like these can be very useful, but in the other hand the performance and the hardware configuration of this module does not satisfy the projects objectives. ZigBee has only four analog channels, which is less than the number of sensors we plan to use.

Also, the used ZigBee modules (Xbee) have a maximum sampling frequency of 50 Hz which less that the average ECG sampling frequency (260Hz).

The data received on the computer, by the other Xbee module and thru the serial port, should contain the two signals. *Processing* framework, explained in the next subsection, is used to read and truncate the data packet then display the graphs.

3.1.2 Processing framework

Processing is framework, language and IDE, based on JAVA language, built for graphical and electronics design communities. Similar to Arduino framework, Processing offers for hobbyists a large amount of libraries and support due to the large community that it has. The graphical capabilities of processing are just below the professional level, yet very powerful options are available. Like in Arduino, Processing is able to communication with the serial ports and the camera and it is very light in size, which makes it easy to install and run. Another option was suggested and tested instead of Processing; LabVIEW Framework but the results have shown that this solution is more practical and useful, because LabVIEW is approximately 15 Gb in size and demands important resources to run normally, which can be unavailable. Yet Processing is much less performant than LabVIEW, in terms of latency, graphical performance etc.

Processing has been used to fulfil three main objectives: receiving and truncating the packets to get the data, acquiring video stream and uploading the data to the internet. At this point, Processing has been able to acquire two signals, battery level and temperature along with video stream, yet some issues came to the surface.

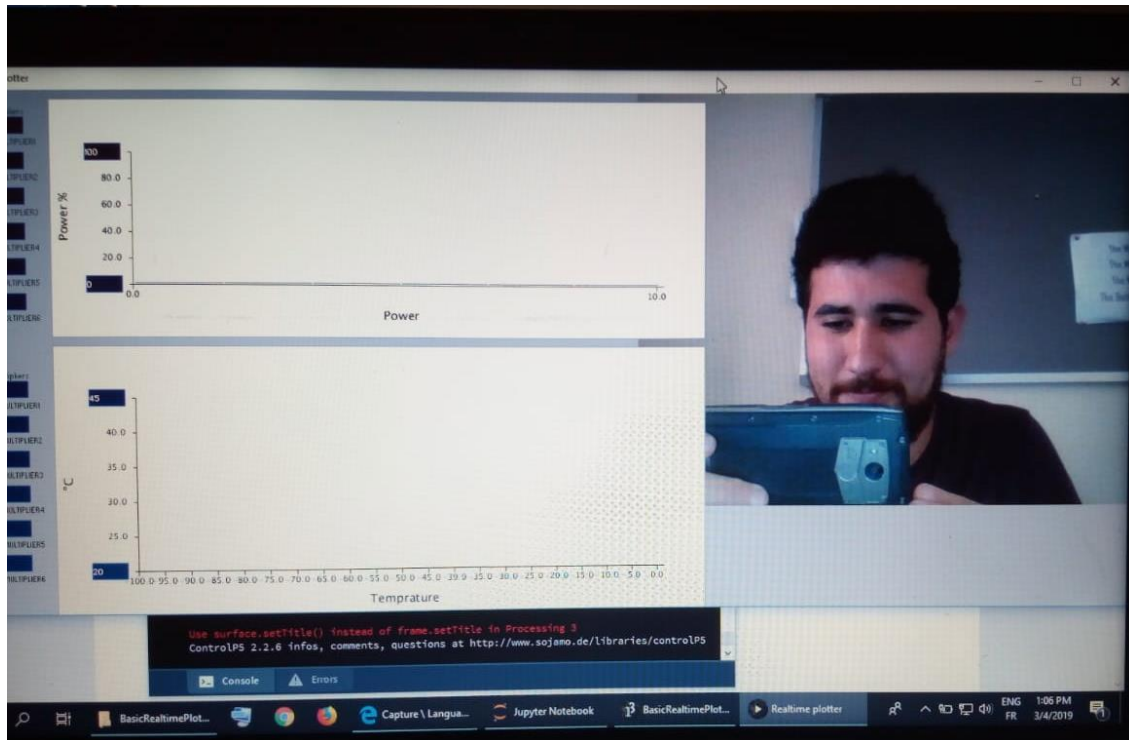


Figure 3.2: Processing program

The video stream is saved as a set of frames under the name of “time-stamp.jpg” in a local folder. Figure 3.2 is a picture taken by a mobile phone of the developed graphical interface. It shows the temperature, battery level graphs and the video acquisition panel.

```
still.copy(cam, 0, 0, cam.width, cam.height, 0, 0, still.width, still.height);
still.save("data/img"+m+" "+s+" "+ms+".png");
```

This line of codes shows the grouping method of the video frames with the title (minutes seconds milliseconds.png).

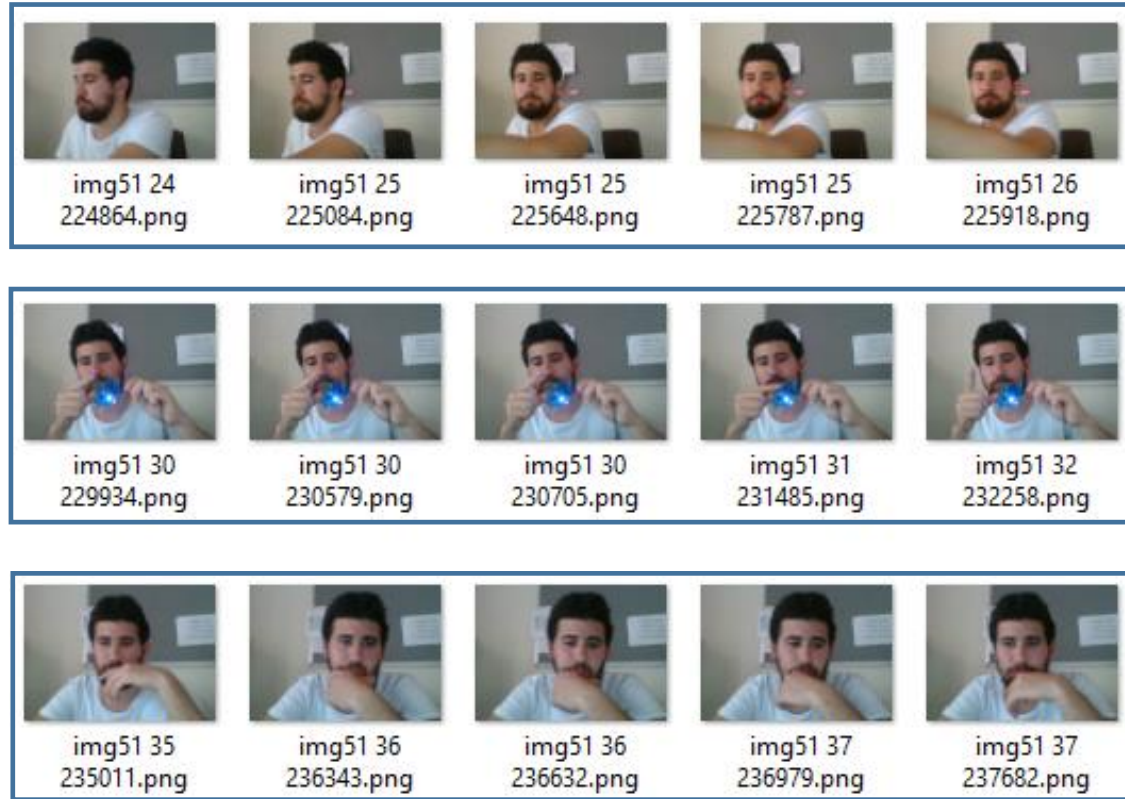


Figure 3.3: Video Frames Grouping

We confirm that all these frames were taken at the minute 51, the first set is from the seconds 24 to 26, the second from 30 to 32 and the third from the second 35 to 37. We can see that the duration between frames varies from 100ms to 700ms. The performance of the computer has a big role in this variation.

In addition, the method of calling the procedure that is responsible of the frame capture has also affected the capture frequency.

We confirm here that the saving of the signals (*savedb*) and the frame capture (*save*) are called in parallel inside the *serialEvent* procedure, which is called whenever serial data is present. The frame capture procedure contains of course a control mechanism that

prevents the frame capture from executing if it is already running, in order to eliminate redundancy.

3.1.3 User web interface

The graphical interface is one of the most important parts of the project, where it is extremely necessary to plot the graphs with high resolution and precision in order to help the doctors and the medical staff understand and take correct decisions. As a first attempt, we tried to use some JavaScript ready modules and integrate them in a web site. Figure 3.4 shows the zooming capability of one module and the real-time plotting capability of another, yet a lot of missing options are necessary in the project but not provided in this module, like time selection, multi graph plotting, etc. In addition, the modules use graph properties and styles from external sources, which can cause privacy problems in another aspect.

Figure 3.5 describes the system architecture, which is in our case composed of the sensors that are connected to a conditioning electronic circuit which transmits the data wirelessly via Xbee modules to a computer, the pc will format the data and send it to a database (for testing purposes we have used a local database) and at last a graphical interface is being developed to show the data.

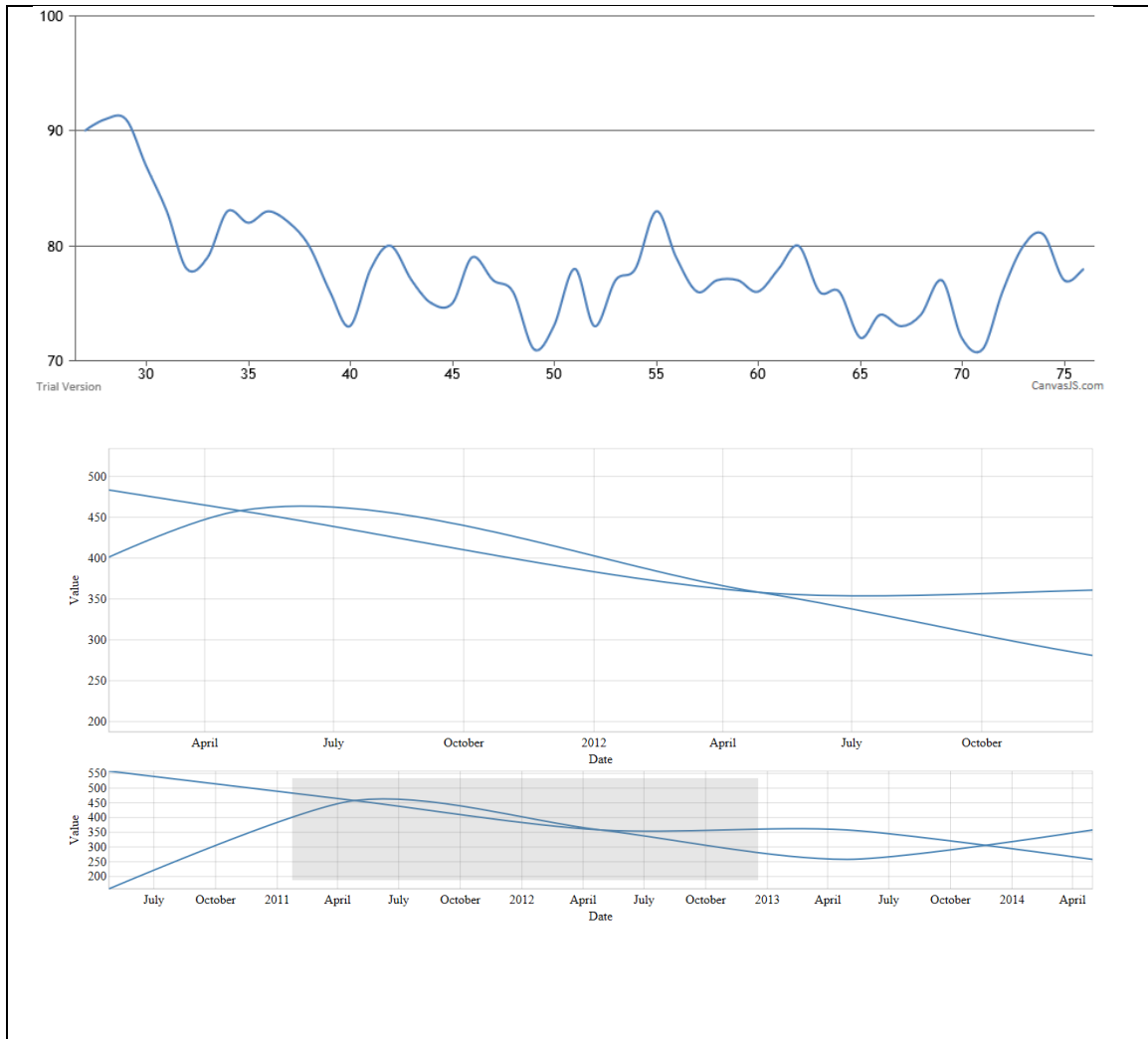


Figure 3.4: graph display example

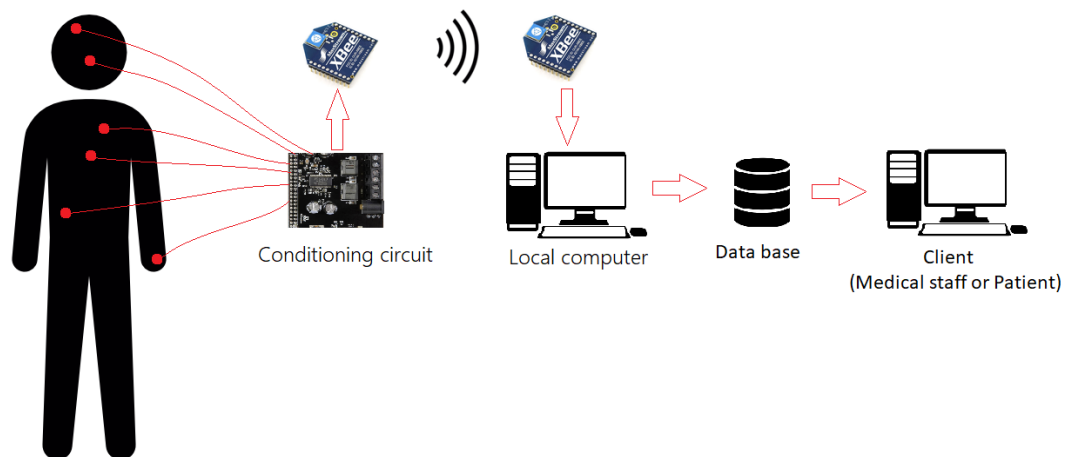


Figure 3.5: System architecture

3.1.4 Conclusion

The link between the sensors and the central unit have been established wirelessly, while the energy consumption of the wireless units (Xbee modules) is very low its performance is relatively low: Xbee has only four ADC channels, with 50Hz maximum sampling frequency. Which interferes with our objectives.

3.2 System architecture version 2

In this chapter, we will discuss a second version of the project, after slightly modifying the objectives and the target project specifications. For the first level of sleep apnea diagnosis, an overnight stay in a sleep laboratory with a technician in attendance is required. A minimum of 7 channels of data (but typically ≥ 16), including respiratory, cardiovascular and neurologic parameters, is mandatory to produce a comprehensive picture of sleep quality. Yet the 3rd level of diagnosis only includes at least 3 channels of data (e.g., oximetry, airflow, heart rate). The project we aim for is to establish a remote diagnosis device on the third level diagnosis standard.

3.2.1 MySignals Platform

Shown in Figure 3.6 a MySigals card with sensors. It is an eHealth and Medical IoT Development Platform, manufactured by Libelium Distributed Communications. The Platform includes a number of professional medical sensors, and electronic signal acquisition card, a TFT screen and an Arduino Uno card.

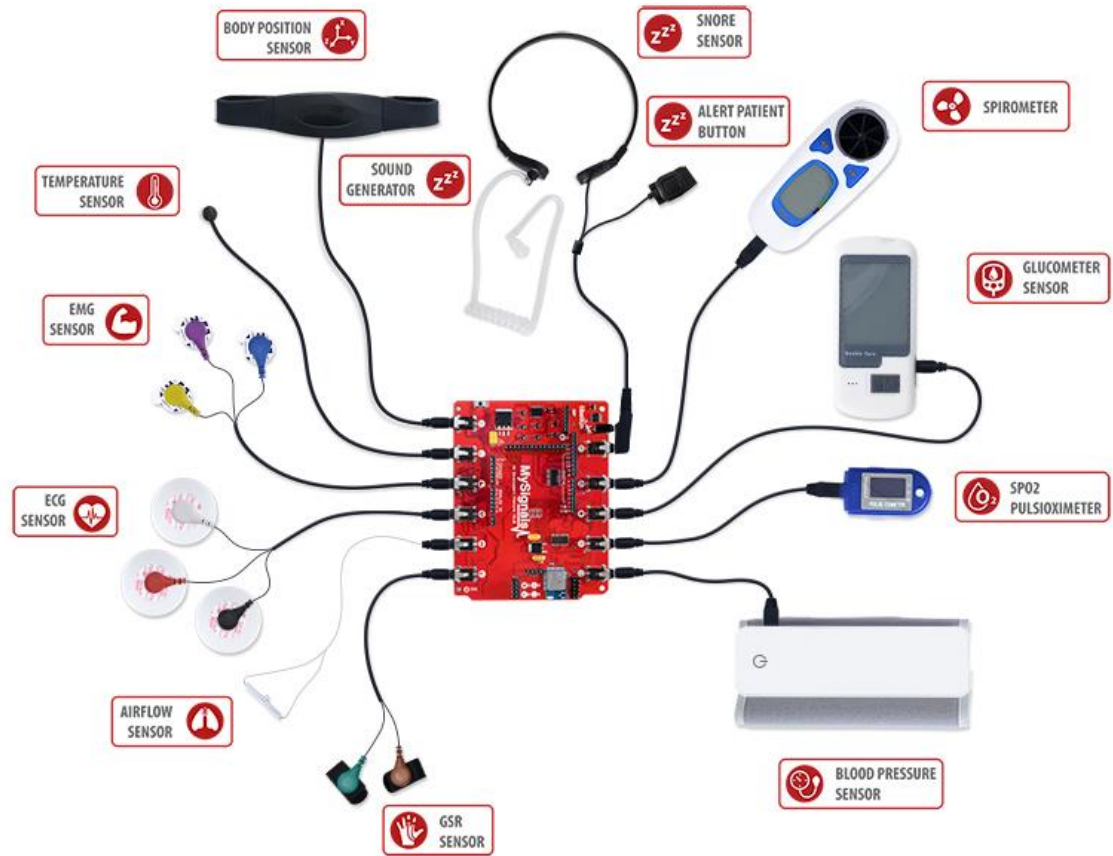


Figure 3.6: MySignals platform and sensors

We have tested a number of sensors on this platform, we got some perfect results, as the conditioning block is perfectly calibrated, yet we discovered some basic problems.

a- Electronic analysis

MySigals is an open source signal-conditioning card based on Arduino Uno card that is built around an 8bits Atmel Atmega328 Microcontroller, functioning on 16MHz frequency [29]. The limits MySigals card are caused by the electronic architecture itself. Because the used microcontroller itself has a limited number of inputs and outputs, the designers have used a multiplexing technique, using HC4051chips to multiplex the input from the sensors with the UART port of the microcontroller, which limits the sampling

frequency of the device. The card has BLE connection capabilities, along with WIFI capabilities, yet the WIFI module is not included and needs to be added separately. The WIFI and BLE modules are both connected to the UART port of the microcontroller, along with the USB driver chip, which explains the need for a multiplexer.

b- Sensors

We have been able to test and number of sensors; SPO2 and Oximeter sensor (BLE), Airflow sensor (Wired), and body position sensor (Wired). Also, the blood pressure sensor (BLE) has been proven to function on the platform but a systematic problem occurred. In order for the blood pressure to function, a connection have to be established between the sensor and the card, then a request needs to be sent to the sensor to activate it and to launch the sensing procedure (pumping air into the arm band and testing blood pressure for approximately 30 seconds) then the resulting data will be sent back to the card, and the sensor is shut down automatically. The systematic functioning of this sensor gives many problems; first the sensor has to be turned on manually before the connection is made, and secondly during almost 30 seconds the card is holding and waiting for data to get back, which makes it impossible for this sensor to work in parallel with other sensors.

Table 3.2: Mysignals supported Sensors

Control and Monitoring	Available Data Transmission
EMG signals	Wired/Bluetooth/WiFi
ECG signals	Wired/Bluetooth/WiFi
Snore signals	Wired/Bluetooth/WiFi
Airflow	Wired/Bluetooth/WiFi
Body temperature	Wired/Bluetooth/WiFi
Galvanic skin response measurements	Wired/Bluetooth/WiFi

Body position detection	Wired/Bluetooth/WiFi
Pulse and oxygen functions	Wired/Bluetooth/WiFi
Blood pressure	Wired/Bluetooth
Glucometer monitor	Wired/Bluetooth
Spirometer monitor	Wired/Bluetooth/WiFi
Body Scale	Bluetooth

To be able to use many sensors at in parallel, we have chosen to use multiple wired sensors (Airflow and Body position) and use the internal multiplexers of the card, along with one BLE sensor (Oximeter). The results were acceptable, yet under the hoped standards. To test this platform, we have used Processing framework mentioned in the first Section Subsection 2, As MySignals card is based on Arduino Uno it is open source and fully programmable and configurable, which means that the data arriving on the serial port is not framed. The user/programmer has the freedom to put the data in a suitable frame. We have chosen to put the data a frame like follows: HR SPO2 AIRFLOW with spaces between each value to detect as a delimiter

```
String[] nums = split(myString, ' ');
```

This instruction is used to cut the data stored in myString (incoming data) into sub strings and store them into a table of strings nums whenever a space is found.

3.2.2 Conclusion

As a reliability confirmation test we went to Sultan Qaboos University Hospital, the medical team in the sleep apnea laboratory set up a professional yet complicated and very expensive vital signals acquiring device, to compare with the proposed device.



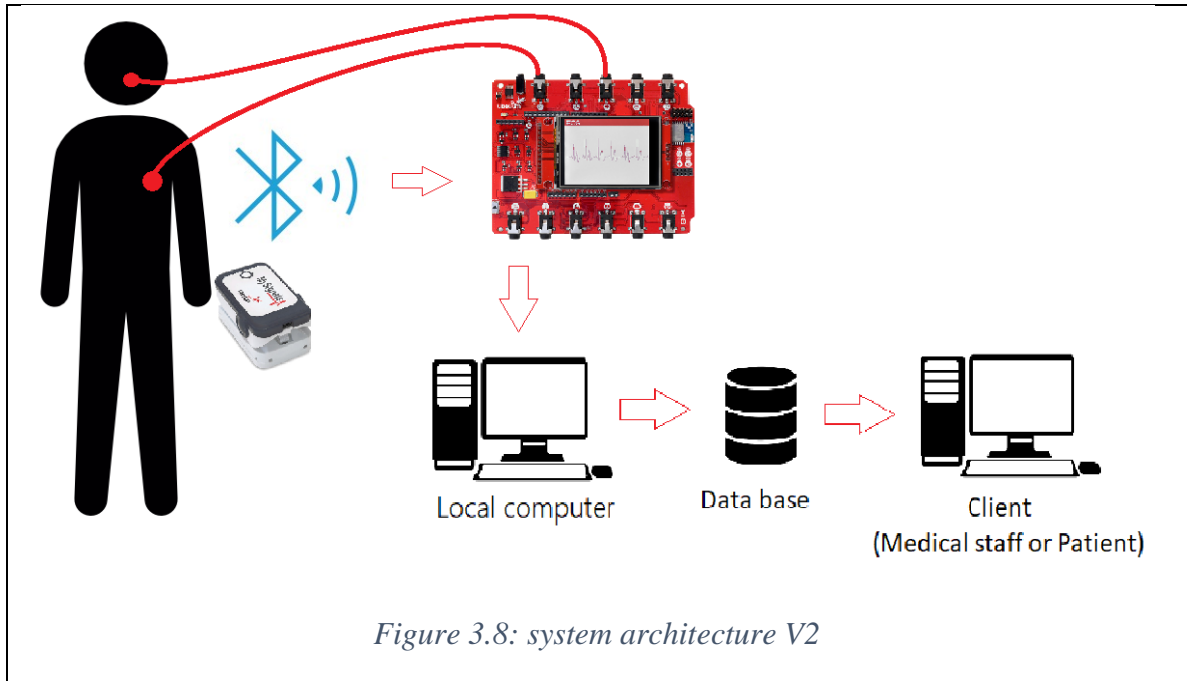
Figure 3.7: Real Test

Figure 3.7 shows the sensors from two different systems connected to my body, the developed system is connected to the computer in front of me, and the professional system from SQUH is located on the table beside me (white device). Two sensors from each device are activated, Airflow and SPO2/HR. the SPO2 sensor we used in our device is wireless as shown in the Figure 3.7 (right hand) whereas the other device's sensor is wired. The test was on for almost 20 minutes. After the doctors' confirmation, the results are very similar.

3.2.3 Overall architecture description

Figure 3.8 illustrates the second architecture we proposed. We used a BLE SPO2 sensor and some other wired sensors. The card still needs a computer to treat the data and log it to the database over the web because although the card has Wi-Fi capabilities it

does not have the WIFI module integrated. An overall evaluation is conducted and the architecture is proven non-responding to the specifications needed.



3.3 System architecture version 3 (final)

3.3.1 ESP32 Card

A computer may not always be available near the patient, so it is better if the architecture does not include a computer. The ESP32 card is a high performance electronic card, with a dual core 32bit processor running a real-time operating system RTOS. It integrates 14 ADC channels. The card also has WIFI and BLE capabilities integrated. It can be programmed using the Arduino IDE, and using the Arduino MySQL library, we have been able to eliminate the computer and directly sample the signals and send them to a database. A WIFI / Bluetooth coexistence protocol is used on the card, which allows using BLE sensors and WIFI connectivity at the same time.

```

#include "BLEDevice.h"
#include <WiFi.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <WiFiClient.h>

```

3.3.2 Sensors

We used two types of sensors in this platform, wired and wireless. We will explain in this section how we handled each type. The wireless sensors communicate using the BLE protocol not the normal Bluetooth protocol. A research on this matter has been conducted in order to understand the functioning of this protocol. The wired sensors have been connected and calibrated using hardware and software methods, in order to get the optimal results.

a- BLE sensors

The BLE Protocol Stack has the functionality of GATT (Generic Attribute Profile) to communicate application data after establishing a connection [30]. Each BLE sensor has a specific profile, which contains the relevant services necessary for operation.

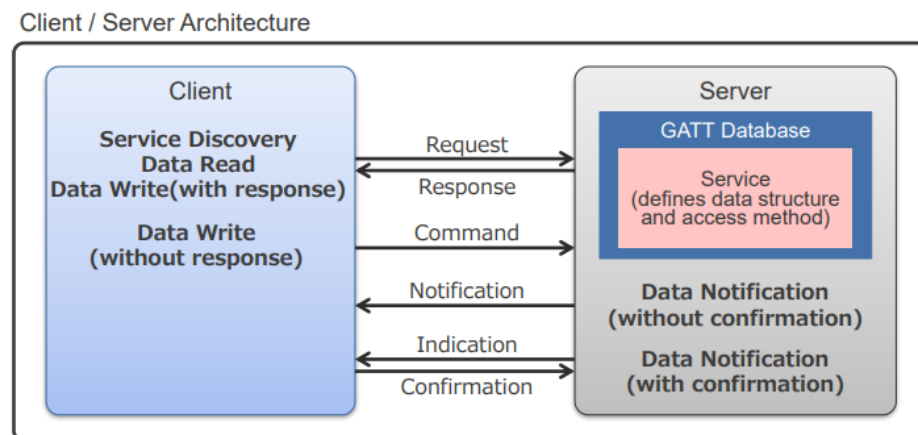


Figure 3.9: BLE architecture

Oximeter sensors for example are used along with calling the Oximeter profile. The Pulse Oximeter profile defines two roles: Sensor (a pulse oximeter device) and Collector (a device that collects data from a pulse oximeter). The Sensor role has instances of the Pulse Oximeter service, the Device Information service and the Battery service. [31]

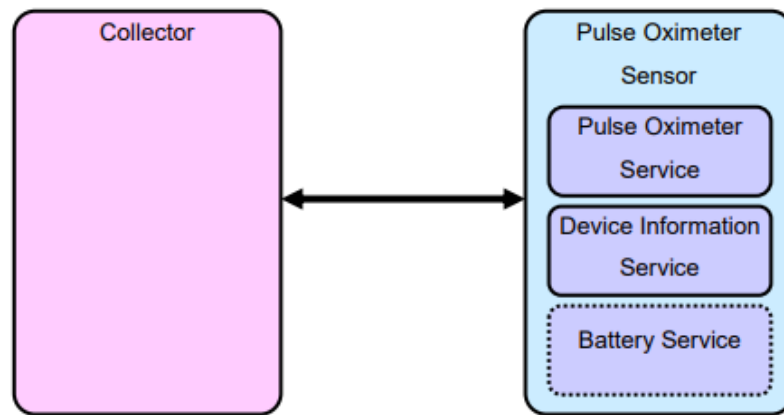


Figure 3.10: BLE architecture

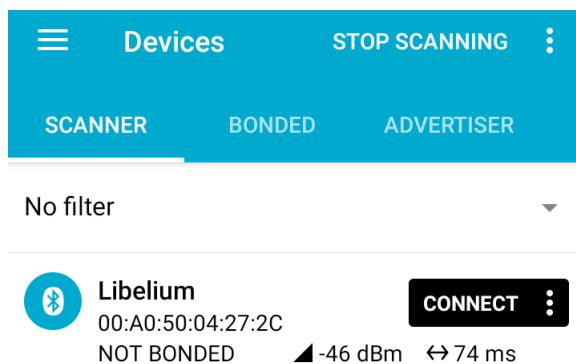


Figure 3.11: nRf connect MAC @ of the sensor

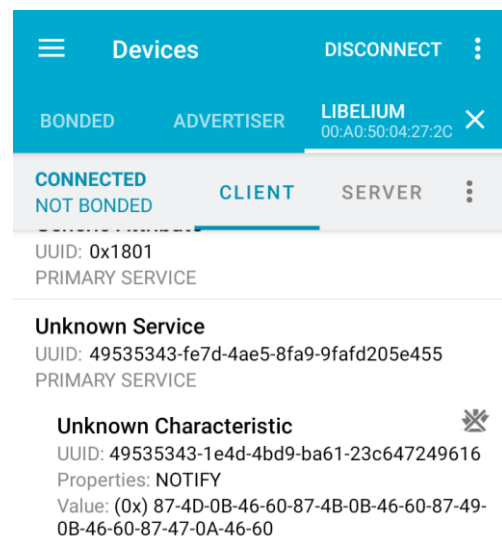


Figure 3.12: nRf connect UUID of the service and characteristic

The BLE SPO2 sensor from the previous platform was used to collect SPO2 and Heart Rate signals. Using *nRf connect* android application the BLE Sensors information have been extracted as shown in Figure 3.11, the mac address is necessary for the connection, also the service's Universal Unique IDentifier (UUID) is shown after connecting to the sensor using the application as illustrated in Figure 3.12. The characteristics UUID is also needed to get the data, we have run a test using the application the see the response of the sensor, when hitting the receiving button data starts to be displayed.

Several attempts have been made to connect multiple BLE sensors simultaneously to the card, but the nature of the BLE protocol prevents such behavior as a client (which is the esp32 card) can connect to one server (which is the sensor) at a time. Yet we have been able to time multiplex the connection. Each sensor will have a time window in which the card will connect to it. The solution is still inefficient as the sampling frequency drops dramatically. One BLE SPO2 sensor and multiple wired sensors like airflow and respiration force were then used and tested.

Table 3.3: BLE Sensors profile examples

Data Type	MDEP Data Type(IEEE 11073-10101 Nomenclature Data Type Code)	IEEE 11073 Document Name
Pulse Oximeter	0x1004 (4100 decimal)	Health informatics - Personal health device communication - Device specialization - Pulse oximeter
Basic ECG (heart rate)	0x1006 (4102 decimal)	Health informatics - Personal health device communication - Device specialization - Basic ECG (heart rate)
Blood Pressure Monitor	0x1007 (4103 decimal)	Health informatics - Personal health device communication - Device specialization - Blood pressure monitor

b- Wired sensors

The ESP32 doesn't have a preprocessing block like MySignals platform, that's why we had to use an instrumentation amplifier to be able to acquire the respiration force signal.



Figure 3.13: Respiration force sensor

The primary tests have shown that the sensor is piezo electric, generating a very small voltage whenever a tension is put to the belt. The current peaks, measured with a voltmeter and an oscilloscope, does not exceed 2mV. An amplifier with a very high gain (x1000) is set to convert this value into usable data. Yet the esp32 offers a functionality on its ADC ports that is useful in this matter. Attenuation range is configurable in the ESP32 ADCs, which gives the programmer a manual selection over the desirable data range to convert which gives a higher resolution.

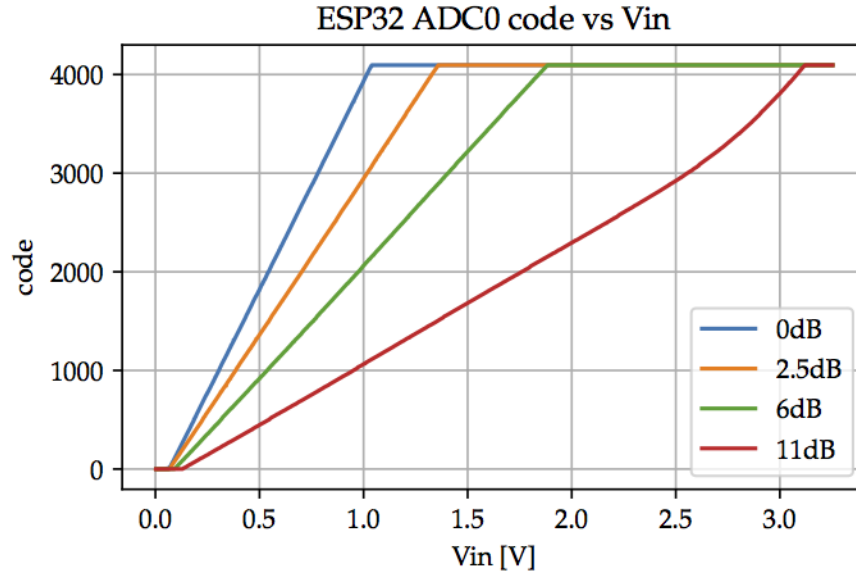


Figure 3.14: ESP32 ADC attenuation

Figure 3.14 explains that if an attenuation of 0 dB is set the practical ADC range will be restricted in approximately 0V to 1V and mapped to 4096 values which can be very useful to increase resolution. Also by using the offset parameter of the ADC we can choose from where to start the conversion range so basically we can choose the proper range and resolution to use with each sensor.

Also an accelerometer has been set up and connected to the card to detect the patient's position, as we have migrated now to a fully mobile solution, the position of the electronic card will reflect the position of the patient's body. The positions should be: Supine, Prone, Right Lateral Decubitus, Left Lateral Decubitus, Standing or Sitting and a non-defined position if anything else. The exact position is calculated using a sum vector of the three axis on the accelerometer [32], the position is deduced after a test on the vector's value. Figure 3.15 is a graph of the values of the three accelerometer's axis. It

has been recorded while moving from one position to another, in order to visualize and describe the data received from the accelerometer.

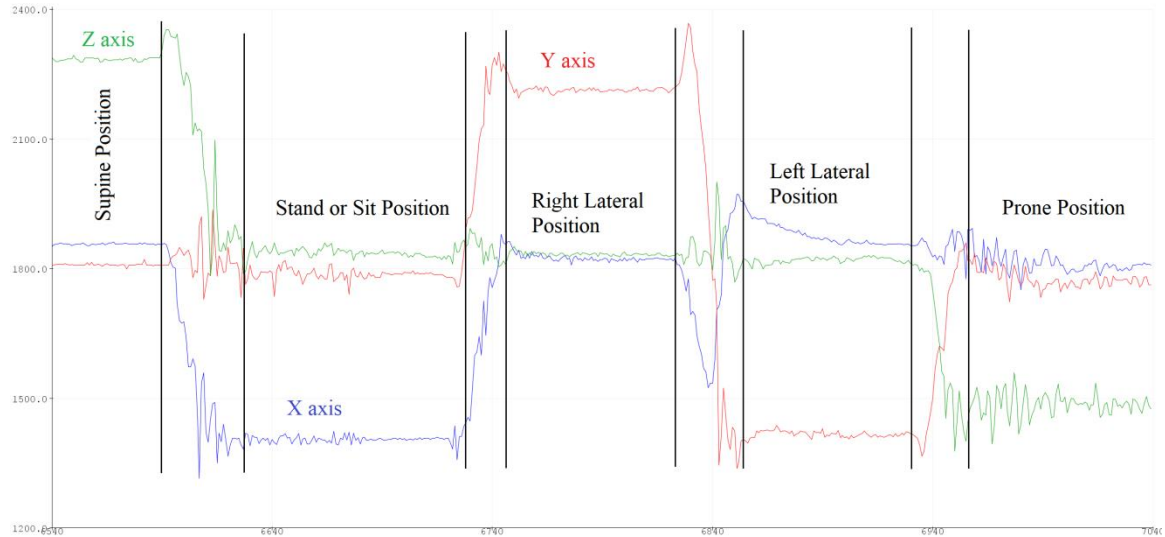


Figure 3.15: Accelerometer's values

3.3.3 Grafana Framework

In order to visualize the data, we realized a web graphic mechanism, based on a framework called “Grafana” which communicates directly with the database and produces an interactive interface with graphs with a minimum refresh rate of one second.



Figure 3.16: Graph example

Figure 3.16 shows a plotted graph of an ECG signal. Note that this signal is not really acquired using the platform, although it is a real ECG signal, it is downloaded from the web to use it in testing and illustration.

Grafana offers some good functionalities like interactive zoom-in, zoom-out, selection, refresh rate, etc. it also integrates a useful tool that helps to directly connect easily to any kind of database. The setup of the interface is also user friendly. As soon as the framework is installed on the server, a username and a password is demanded to login. Multiple accounts can be created with different privileges (Doctor, Medical assistant, patient). As soon as a user is logged in an interface shown and the user is able to display or create graphs like shown in the Figure 3.17.

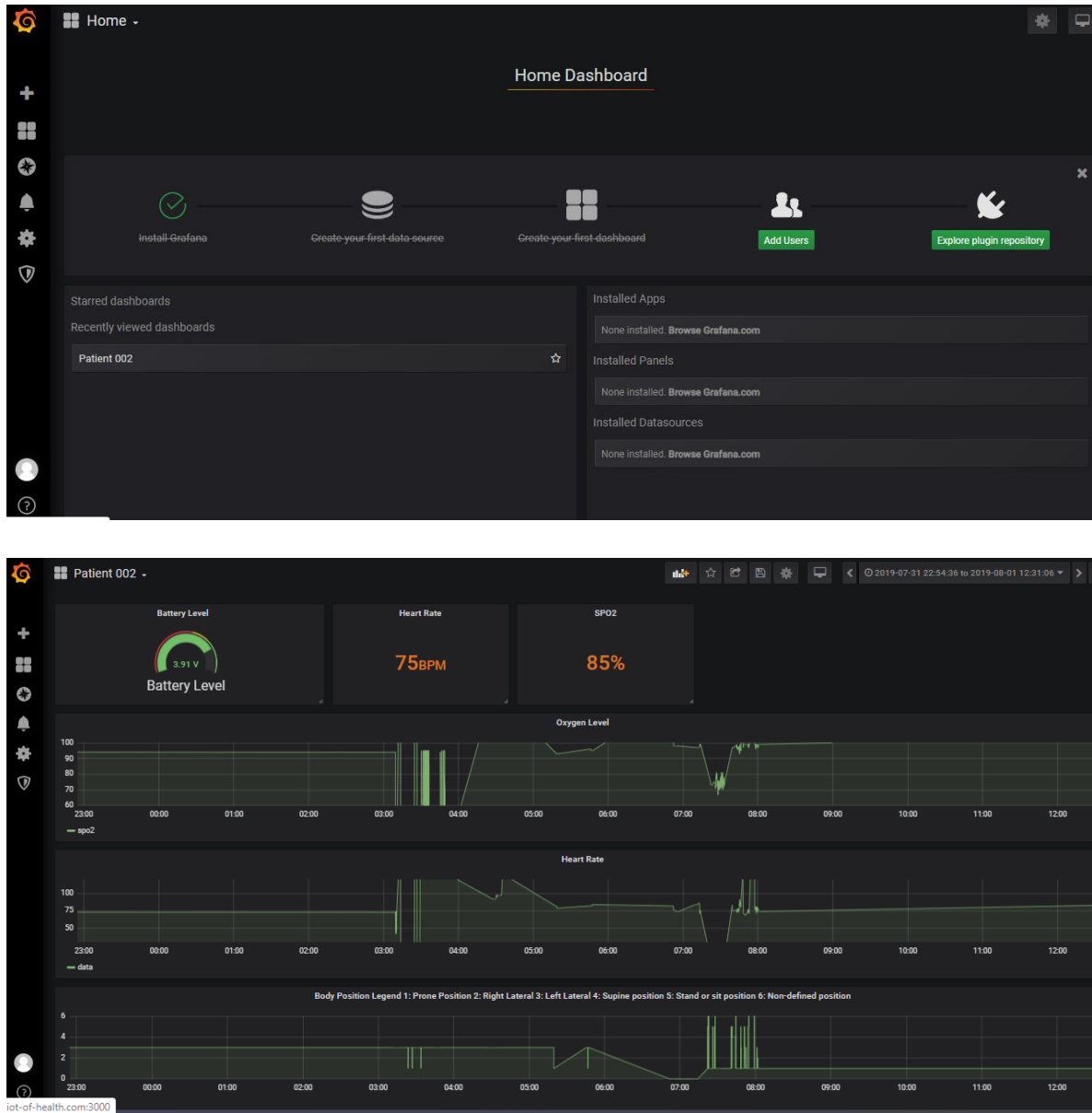
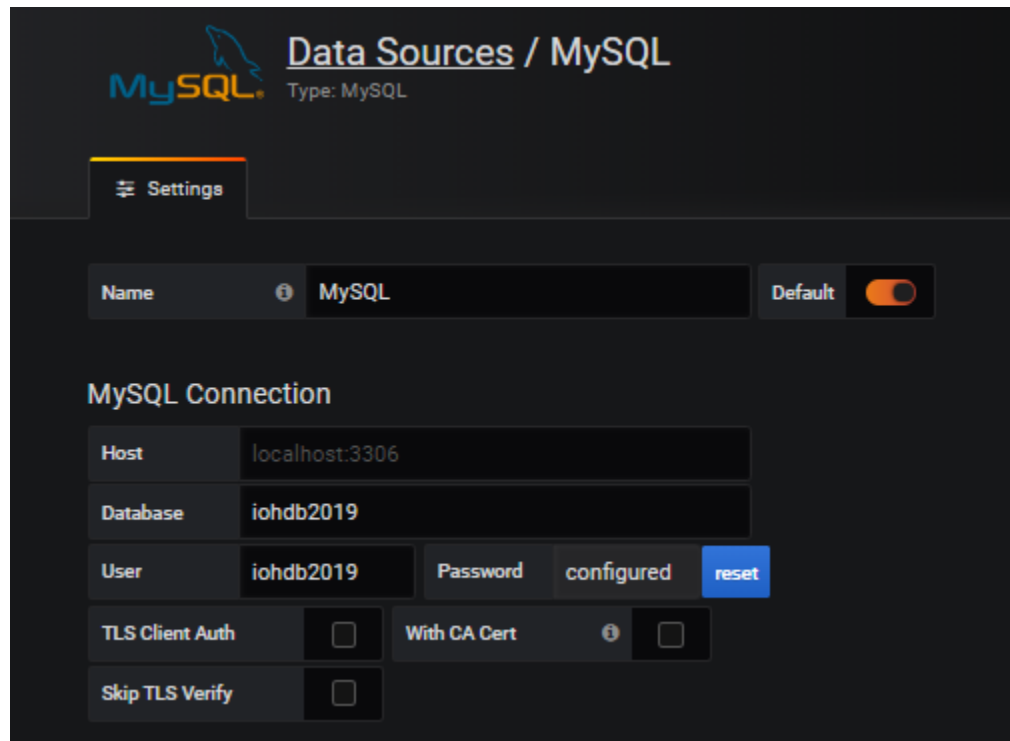


Figure 3.17: Grafana Framework

As a beginning, only four signals are transmitted and plotted on the interface: oxygen level (SPO2), Heart Rate (HR), Body Position and battery level. Figure 3.17 above shows the graphs of the three vital signals in a time window, along with the current or last received values in the boxes above.

The user can add a data source (database) from the data source panel on Grafana interface. The MySQL database and Grafana framework has been deployed side to side on an internet sever. The picture underneath illustrates the parameters needed to set up a data source.



The screenshot displays the 'Data Sources / MySQL' configuration page in Grafana. At the top, the MySQL logo and 'Type: MySQL' are shown. Below this is a 'Settings' tab. The main configuration area includes a 'Name' field set to 'MySQL' and a 'Default' toggle switch that is turned on. Under the 'MySQL Connection' section, the 'Host' is 'localhost:3306', the 'Database' is 'iohdb2019', and the 'User' is 'iohdb2019'. The 'Password' field is marked as 'configured' with a blue 'reset' button. There are also checkboxes for 'TLS Client Auth', 'With CA Cert', and 'Skip TLS Verify', all of which are currently unchecked.

Figure 3.18: Grafana data source setup

Although a minimum refresh rate of 5s is set by default, a changing in Grafana's main core have been made to have the possibility of setting a minimum of 1s refresh rate. Although the communication time between the database and Grafana is mostly more than 1s so it may cause problems loading data.

3.3.4 Conclusion

We have run multiple power consumption tests, using batteries, in order to define power consumption in real use cases. The results were promising; a full charged 1500 mAh battery lasts from 8 to 12 hours continuously running.

Even with such powerful resources and promising results, some problems still exist. The card is now uploading data directly to the database, although the processor frequency is high, the sampling frequency is still relatively low, as it depends on the internet connection. The problem is that the MySQL connection protocol demands to receive an acknowledgment after sending data to the database. Therefore, the duration that takes data to arrive to the DB will be doubled. The durations between samples in the database varies from 100 to 500ms depending on the quality of the internet. We tried to eliminate this problem by temporarily stocking the data in the FLASH memory of the ESP32. The sampling frequency was acceptable but the memory could not stock more than one minute of samples. We tried to use the dual cores in parallel, affecting the first for sampling and stocking and the second for uploading to the database, using the flash memory as a pipeline, again the low capacity of flash memory is a holdback. After approximately five minutes, the pipeline gets full and the low speed-uploading task is not able to keep up. The idea was to use the internal memory of the ESP32 card as a pipeline for samples, where data will be stored in a memory section called Serial Peripheral Interface Flash File System (SPIFFS) and uploaded to the data base from that memory at the same time. This partition allows permanent storage of data even after a restart. According to the modules, its size varies between 512kB to 4Mo. The memory access

coordination is managed by the Real Time Operating System core (RTOS) already deployed on the card.

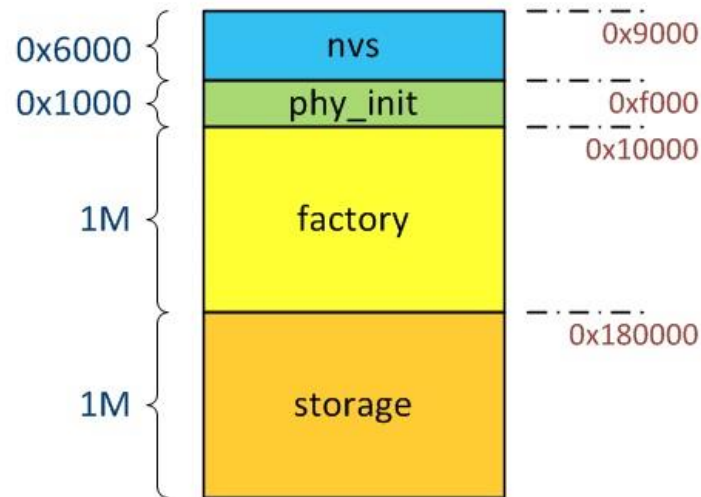


Figure 3.19: Memory partitioning of ESP32 card

We have made a test for this method, acquiring the mentioned vital signals along with the local timestamp of the samples. The data is stored in a SQLite table on the SPIFFS partition. At this point the sampling rate was fast 8-10ms (125Hz) but the reading rate was quite high 70-80ms (15Hz). The first CPU core was assigned to sample and log data while the second core was programmed to upload data to the distant database. The memory space used as a pipeline got saturated in approximately three minutes. The problem is that the upload rate is still very slow comparing to the sampling rate, due to the slow reading rate. This problem is mainly because of the nature of the memory partition, as it is physically situating on the same peripheral with the program code. This delay combined with the transmission time depending on the internet connection quality makes the uploading frequency very slow.

Also some BLE connection problems show up sometimes, due to interferences and to the sensor's poor quality. The BLE connection is interrupted and the device is stuck and get

back on only if we manually push the reset button. To fix the problem we added an auto reconnection mechanism in the program. For that, we had to add a *get_error* method in the basic *BLEDevice* library and use it to get the error number, if the error is *BLE_disconnected*, the mechanism is triggered.

The setup has been tested in real conditions over night, promising results were given. A test of seven hours is conducted and perfect signals were shown, the battery life is confirmed good and reconnection actions were made automatically during the test. At this point, the project is confirmed good for professional use for level three diagnosis (home auto-analysis).

3.3.5 Overall architecture description

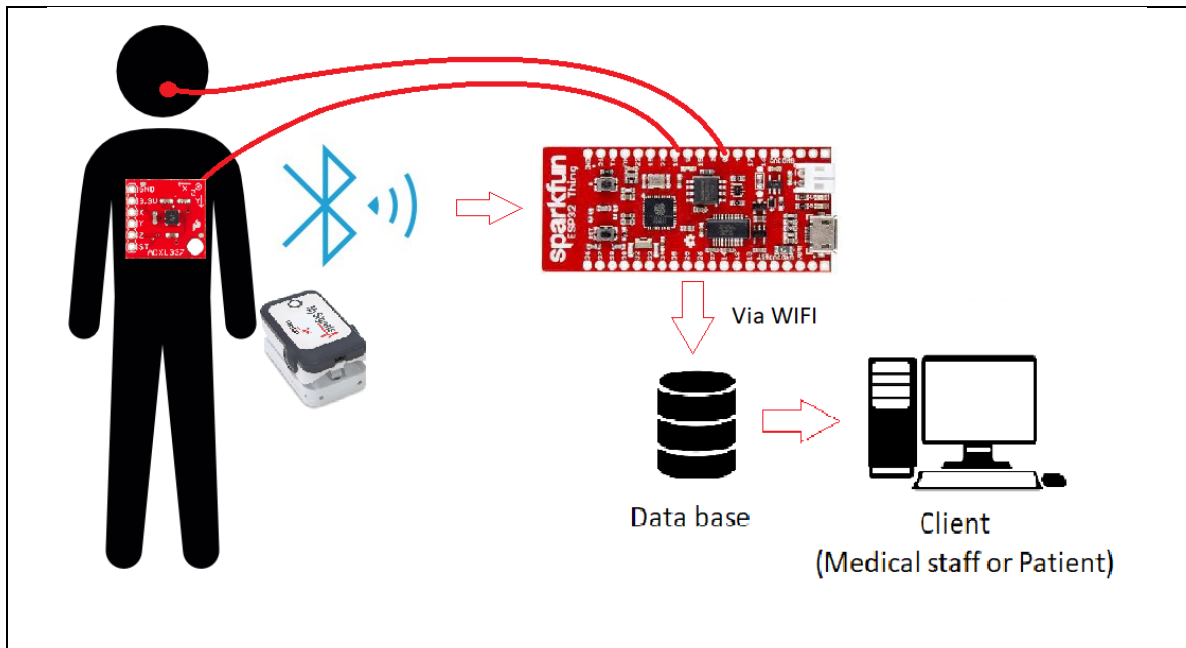


Figure 3.20: System architecture V3

We notice that the new architecture eliminates the need for a local computer. An accelerometer is attached to the patient's body along with several sensors and the wireless SPO2 BLE sensor.

3.4 Machine Learning Model

The most important task in this project is to classify ECG signals using AI. Several attempts have been made to classify an ECG signal into three classes, Normal, Noisy and Atrial Fibrillation. The second and the first level sleep apnea diagnosis demands an ECG signal acquisition and analyzing. Going thru eight hours of graphs looking for cardiac disorder can be tiring for medical staff. For that, we tried to build a machine-learning model that will dynamically analyze the ECG signal and notify the doctor if anything shows up. The idea is to analyze the signal part by part, of 60 seconds for example, and if the signal partition contains health problems, the medical staff is notified.

3.4.1 Data description

The electrocardiogram (ECG) signals we used are one lead, short signals collected from real patients, downloaded from physionets website. These signals have been put to the public in the frame of a medical computing challenge "PhysioNet CinC Challenge 2017" [33]. The signals files are in MAT extension files, which is developed by Mathworks for use with the MATLAB software. In this work, these files were read and treated using Python. The data inside the files are in the form of a matrix describing the sample number and the signal amplitude. The length of signals varies from one to another

(from thirty to ninety seconds). The signals have been sampled with a 300 Hz frequency. 8528 Records are available on the site. After removing the "Other" class, leaving only AF, N and ~, a data set of 5971 records were used in the training. A separate 300 records were downloaded, 230 are used for testing and verification.

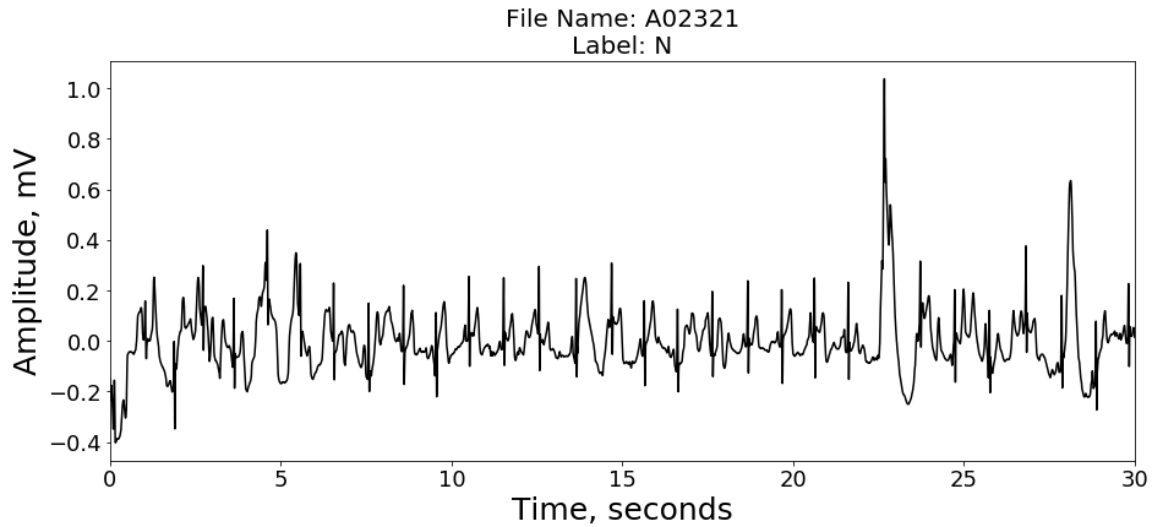


Figure 3.21: ECG signal example

3.4.2 Feature extraction methods

a- Fast Fourier Transform FFT

The Fourier transform (FT) decomposes a function of time (a signal) into its constituent frequencies. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The FT equation is defined as follows:

$$F(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-2j\pi\omega t} dt$$

In the above equation x stands for frequency, t stands for time, and $x(t)$ denotes time domain signal. The signal $x(t)$, is multiplied with an exponential term, at some certain frequency “ x ”, and then integrated over all the times. This integral is calculated for every value of “ x ”. If the value of this integration is large, then this means that the signal has a major component of “ x ” in it.

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT).

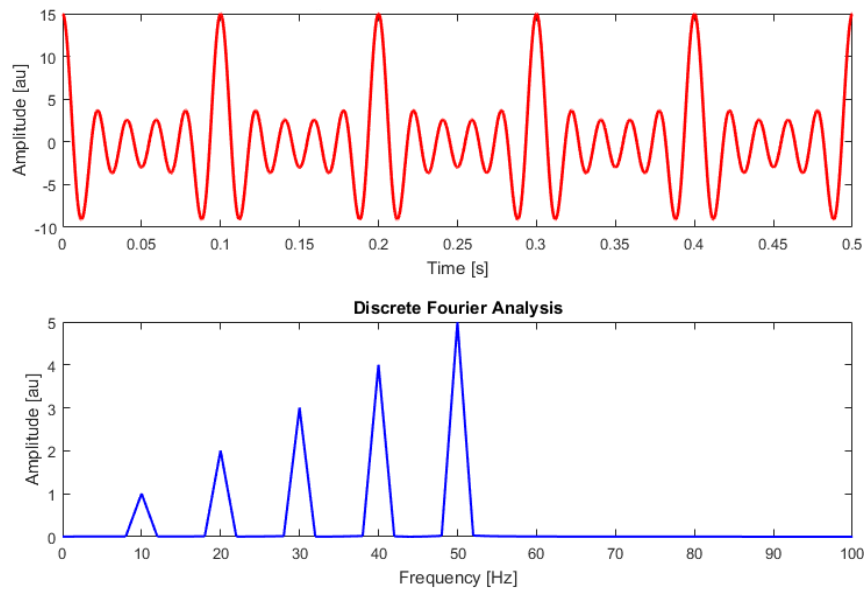


Figure 3.22: Discrete Fourier Transform illustration

The picture above illustrates a discrete Fourier analysis of a sum of cosine waves at 10, 20, 30, 40, and 50 Hz. If the FT of a signal in time domain is taken, the frequency–amplitude representation of that signal is obtained. That is, we have a plot with one axis being the frequency and the other being the amplitude. This plot tells us how much of each frequency exists in the raw signal. However, it does not tell about what spectral component exist at any given time instant, i.e., the time information is lost.

The idea here is to choose some frequencies, for example 50Hz,75Hz,100Hz,and 125Hz and calculate the FFT for each one, then the calculated FFT value becomes the feature. The number of features here is statically chosen, equals to the number of frequencies we set.

b- Wavelet Transformation

The wavelet representation gives the optimal resolution on time domain and frequency domain by adaptively partitioning the time–frequency plane, using a range of window sizes.

$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \phi(Sx - k)$$

The equation above defines the Discrete Wavelet Transform, where S is a scaling factor (usually chosen as 2).

At high frequencies, the WT gives up some frequency resolution compared to the FT. The WT breaks the signal into its wavelets (small wave) which are scaled and shifted versions of the original wavelet so-called mother wavelet. In contrast to the Fourier transform, the wavelet transform deals with the limitations of uncertainty principle in a way that is more convenient for most real-world signals. The Discrete Wavelet Transform (DWT) is probably the most popular type of the wavelet transform in the signal and image-processing field. This transform has become successful primarily in compression, as it became part of the JPEG2000 standard. As displayed in Figure 3.23, Figure 3.24. at each decomposition stage, the transform produces detail coefficients and

approximation coefficients corresponding respectively to the upper half and the lower half of the input signal spectrum. The approximations then become an input of the next level. [34]

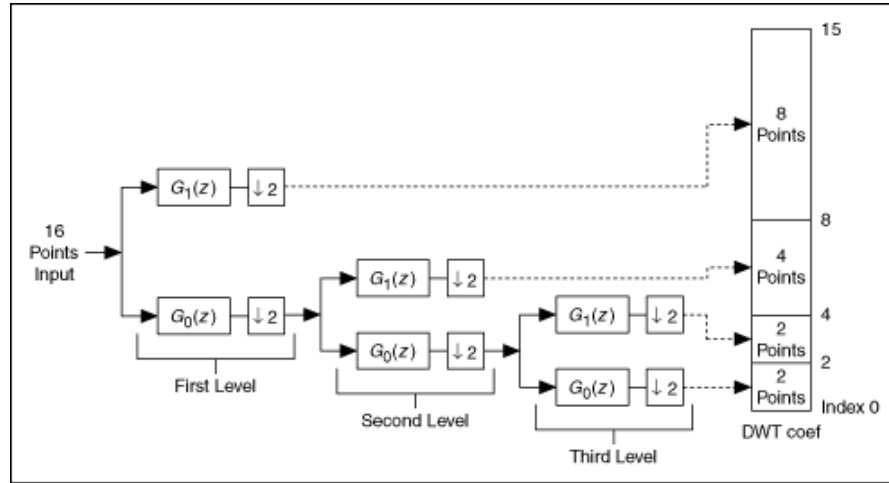


Figure 3.23: 1D 3 level DWT illustration

As the signal can be considered as a matrix of two dimensions (time and amplitude), a two-dimensional DWT can be applied to obtain better results. Two-dimensional DWT applies the filters symmetrically on high and low frequencies as shown in the picture below.

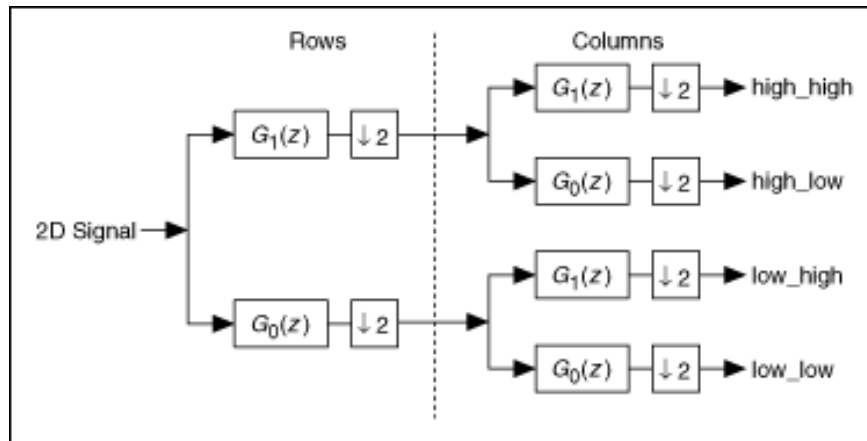
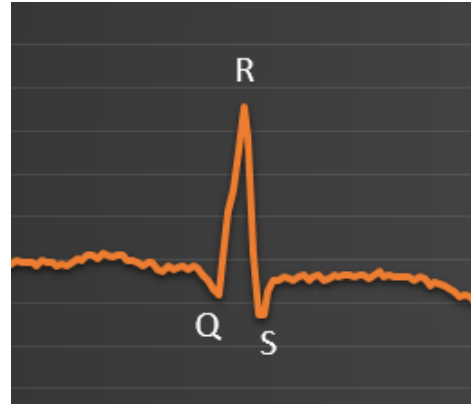


Figure 3.24: 2D DWT illustration

DWT is also applied on frequencies depending of the filters we set. It is correct to say that the number of features in this method is also static depending on the number of levels we set and the configuration of filters.

c- R-R Interval and variability Method

The electrocardiogram contains a repetitive shape that can be analyzed in order to diagnose a cardiac illness. The QRS complex is the most important part of the ECG signal. This complex defines and hold the properties of the signal (frequency, amplitude, etc.). R peaks are the most



important component in ECG signals and are the most important agents that are used to diagnose cardiovascular disease. Detecting R peaks is a critical task, which led us to implement a library “BioSPPy” which offers many bio-signals processing tools like filtering, reading, plotting, detecting R peaks and many other functions. We used this library to detect R peaks, and then with a simple procedure, calculate each interval between two peaks. Each signal has its duration and each one contains a number of heart beats and R-R intervals different from the other which leads to a non-uniform data set.

Table 3.4: R-R intervals table

Records	0	1	2	3	4	5	6	7	8	9	...	99	100	101	102	
R1	0	215	218	237	243	231	239	244	242	234	240	...	NaN	NaN	NaN	NaN
R2	0	306	301	305	318	305	315	321	308	108	213	...	NaN	NaN	NaN	NaN
R3	0	223	231	232	236	236	227	214	211	217	223	...	NaN	NaN	NaN	NaN
R4	0	286	277	323	358	309	342	403	253	215	224	...	NaN	NaN	NaN	NaN
R5	0	251	255	275	297	310	316	306	305	313	316	...	NaN	NaN	NaN	NaN
....	0	309	305	301	300	301	292	285	287	291	302	...	NaN	NaN	NaN	NaN
	0	145	139	204	175	212	173	261	216	159	194	...	NaN	NaN	NaN	NaN

Supposing that S is the ECG signal with $x = \{x_0, x_1, x_2, \dots, x_n\}$, with $n + 1$ the number of samples inside the signal, n can be calculated by $n = \text{signal duration} \times \text{sampling rate}$. After detecting R peaks another vector is obtained $p = \{p_0, p_1, p_2, \dots, p_m\}$ where $m + 1$ is the number of elements in the vector, m cannot be calculated because it depends on the patient's heart rate and it is not static. R-R intervals are also stored in a vector defined as follows $i = \{i_0, i_1, i_2, \dots, i_k\}$ where $k = m - 1$ and i is defined as a sequence $i_n = (p_{n+1}) - p_n$. The difference in length will lead to empty (NaN) cells inside the data-frame as shown in Table 3. because the width of the data frame will be determined by the longest signal. To treat this issue we suggest dropping the columns that exceeds a certain threshold and replace the NaN values in the columns that did not reach the threshold with a chosen value.

3.4.3 Tests and results

a- Feature extraction method

The first attempt was to extract signal parameters to use in learning. Calculated parameters are based on heart rate: `hr_max`, `hr_mean`, `hr_median`, `hr_std`, `hr_skew`, `hr_kurtosis`. The heart rate itself is calculated using the R peaks in the annotations. Six features were extracted as a beginning. After the construction of the model, which is based on a Convolutional Neural Network, the accuracy was very low. Therefore, other parameters that depends on the signal itself have been computed `full_waveform_min`, `full_waveform_max`, `full_waveform_mean`, `full_waveform_median`, `full_waveform_std`, `full_waveform_skew`, `full_waveform_kurtosis`, `full_waveform_duration`. Also with the application of a Wavelet Transform algorithm for three frequency bands and four levels that gives these parameters:

<code>swt_d_level (n) _low_power_ratio</code> ,	<code>swt_a_level (n) _low_power_ratio</code> ,
<code>swt_d_level (n) _med_power_ratio</code> ,	<code>swt_a_level (n) _med_power_ratio</code> ,
<code>swt_d_level (n) _high_power_ratio</code>	<code>swt_a_level (n) _high_power_ratio</code>
<code>swt_d_level (n) _energy_entropy</code>	<code>swt_a_level (n) _energy_entropy</code>
<code>swt_d_level (n) _higuchi_fractal</code>	<code>swt_a_level (n) _higuchi_fractal</code>

Which gives 48 parameters, the accuracy of the proposed CNN model did not reach a reasonable accuracy using these parameters.

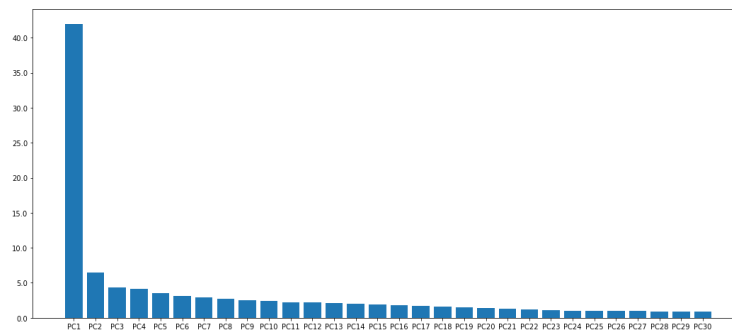


Figure 3.25: PCA application

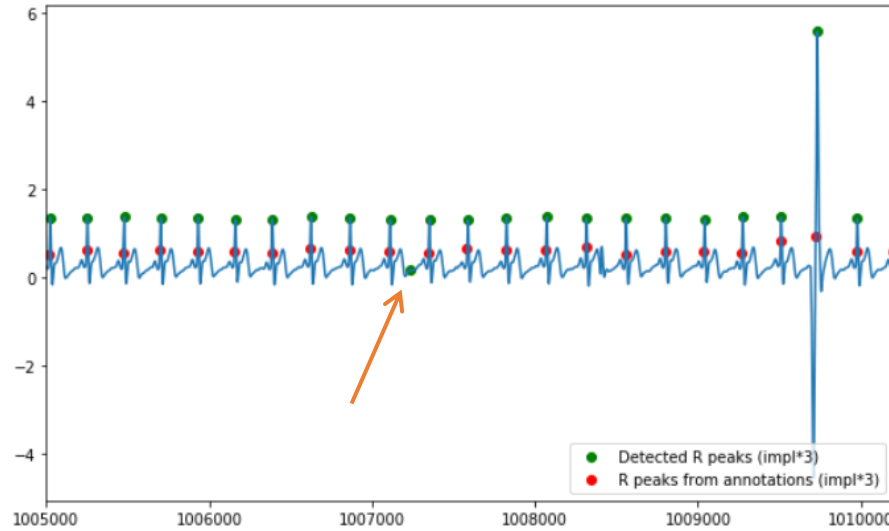


Figure 3.26: Peak detection error example

A Principal Component Analysis parameter selection algorithm was applied on the features, the Figure 3.25 shows the results, the first three components were chosen for classification. The accuracy increases slightly but remains below acceptable. As this method did not show any results we tried to inject the R-Peaks directly into the CNN, for that we tried to use a peak detection algorithm. The method we've used to detect R-peaks uses a sliding window thru the signal with the size of 200 samples on the right of the R-peak. The first window was set after detecting the first peak in the signal, which is the highest value reached before dropping. From that, the window is set on the right side of the detected peak to detect the next peak. Inside the window, the highest value is considered an R-peak. And from that the window moves and the same highest value detection mechanism is used. This did not work in good and promising results. The algorithm used is not precise, we can always observe errors. This is due to the fact that the R-peaks frequency varies from one signal to another, while the size of the window is set to be static in this algorithm. So an ECG signal that is fast enough can have two or more R-peaks inside the windows that we've set, in this case the first peak will be detected while the second will be passed undetected. Yet a slow enough signal can cause

the opposite problem, the window that is set can be small enough not to reach the next R-peak, so the highest value is detected which can be another type of wave (p wave for example) as shown in Figure 3.26, The detected peaks are displayed in green; the peaks from annotation are displayed in red. We can see a miss placed green dot in the figure.

Figure 3.27 illustrates the noise in some of the signals. This noise can be due to low quality sensors, movement of the patient's body, electrostatic charges or any of other causes.

This noise is interfering with the machine learning process and needs to be filtered, a bandpass Butterworth filter with order=8, with cut-off frequencies of 3Hz and 40Hz has been applied and a Standard Deviation/Normal Distribution technique has been used on the signals. The results are illustrated in figure

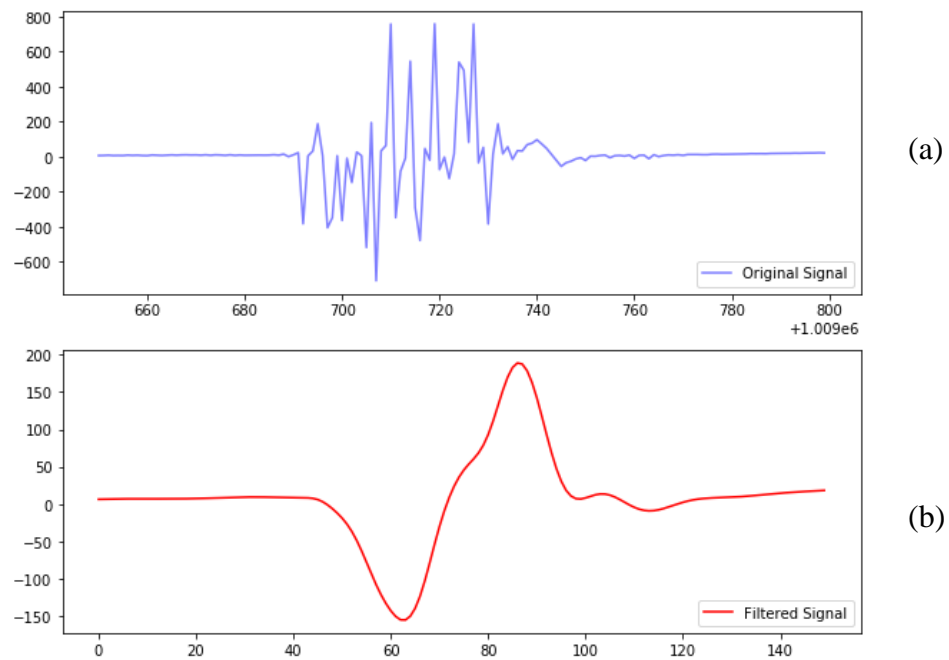


Figure 3.27: Applying a Butterworth filter on Noisy signals (a) sample of noise in ECG signals, (b) signal sample after filtering

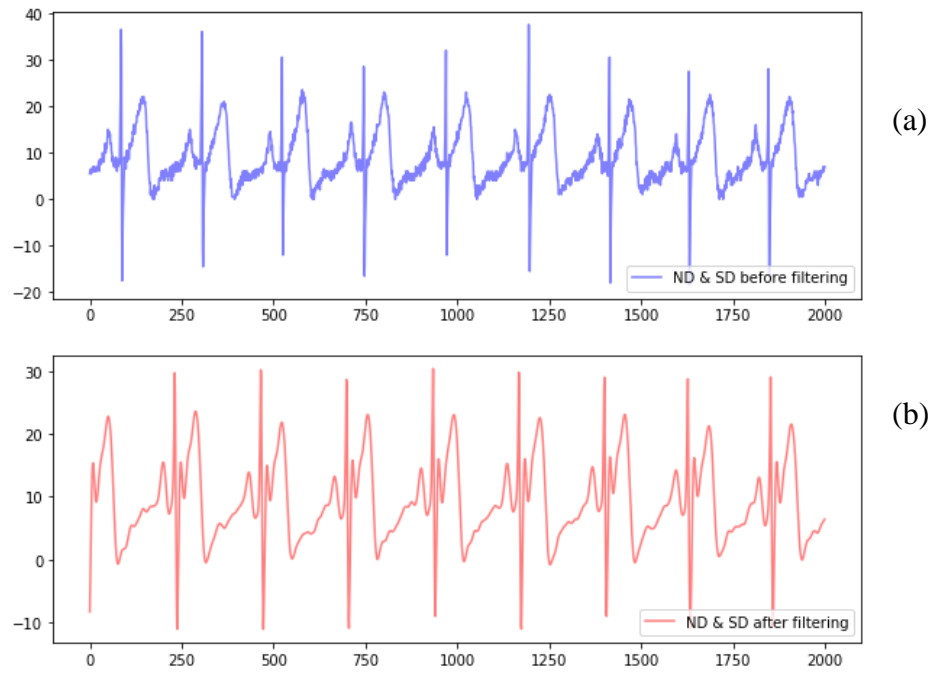


Figure 3.28: Applying a Normal Distribution & Standard Deviation filtering technique (a) ECG signal before filtering, (b) ECG signal after filtering

As we tried to inject the signals from different patients to the CNN, a problem has arisen regarding the form of the data. The signals have different lengths and contain different numbers of peaks, while it is mandatory to inject data of the same form (array of the same length) for each patient for the CNN to work. To solve this, we tried to cut the number of peaks for the minimum present in the dataset. This method gave no results.

At this stage, a library named BioSppy on python has been used, it offers tools for the processing of bio-signals, such as filters, R-peaks detection algorithms, etc. The implemented R-peak algorithm is based on Hamilton P. approach proposed in 2002 [35] this QRS complex detector is based on a bandpass filter with cutoff frequencies of 16Hz-8Hz, then working on time domain an sliding window of 80ms is used to detect peaks. Only peaks that satisfies a list of rules is kept. The algorithm reached an accuracy of

97.83%. After R-peaks detected, R-R Intervals have been calculated. Every signal has its properties, so the obtained RRI table is definitely not uniform. Figure 3.31 (a) and (b) illustrate a sample of the obtained table and the missing values in it. To eliminate this problem, we have tried two techniques 1) cut off the data exceeding the length of the minimum array 2) cut off the data from a manually chosen threshold (800 for example) and fill in the other missing values with a manually chosen value (250). This pseudo code describes the algorithm used to pre-process the data.

```

For Raw_ECG in files
  Filtered_ECG <- Butterworth(Raw_ECG)
  RP_vector <- Detect_R-peaks(Filter_ECG)
  RRI_vector <- Calculate_RRi(RP_vector)
  Data_frame[patient index]<- RRI_vector
End for
For column in Data_frame_colmuns
  If NaN_values > Thresh then
    Drop_column
  Else
    NaN_values <- 250
  End for

```

The difference between technique 1) and 2) is that the first technique sets the threshold variable **Thresh** to 1, so the first column containing a missing value will be amputated along with all successor columns, but the second technique uses a different threshold value = 800 (also tested later with 750) to amputate the columns that reached the threshold and leaving the columns that didn't. those columns containing missing values will be treated, and the missing values will be replaced by a constant number = 250 which is the average *This method gave very high results, with accuracy ~97% for learning and ~95% for testing.*

The method architecture we used is illustrated in the following flowchart:

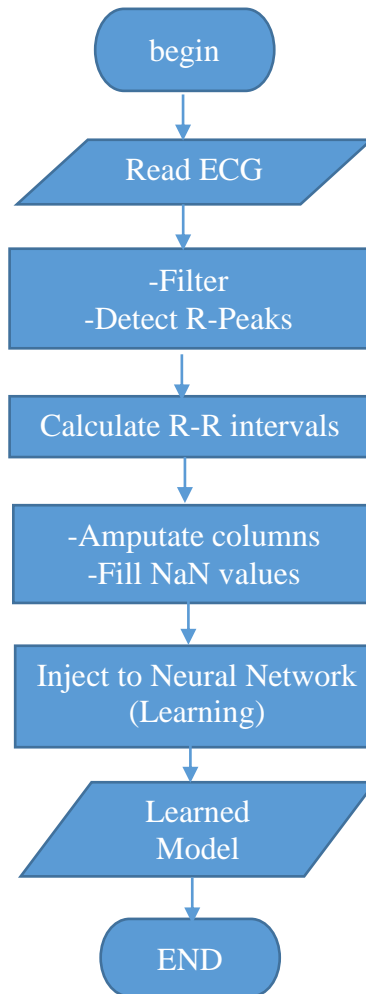


Figure 3.29: method's flowchart

The ECG signals varies in Length (duration) and in Heart Rate (number of R peaks) Figure 3.30, after going thru the first part of signal preprocessing; detecting R peaks, calculating R-R intervals and recording it in a data-frame, we will face a non-uniformity problem. The R-R intervals vector will not have the same length from one

patient to another. Patient X Figure 3.30 (a) will have much longer R-R intervals vector than patient Y Figure 3.30 (b) as he has a much higher heart rate.

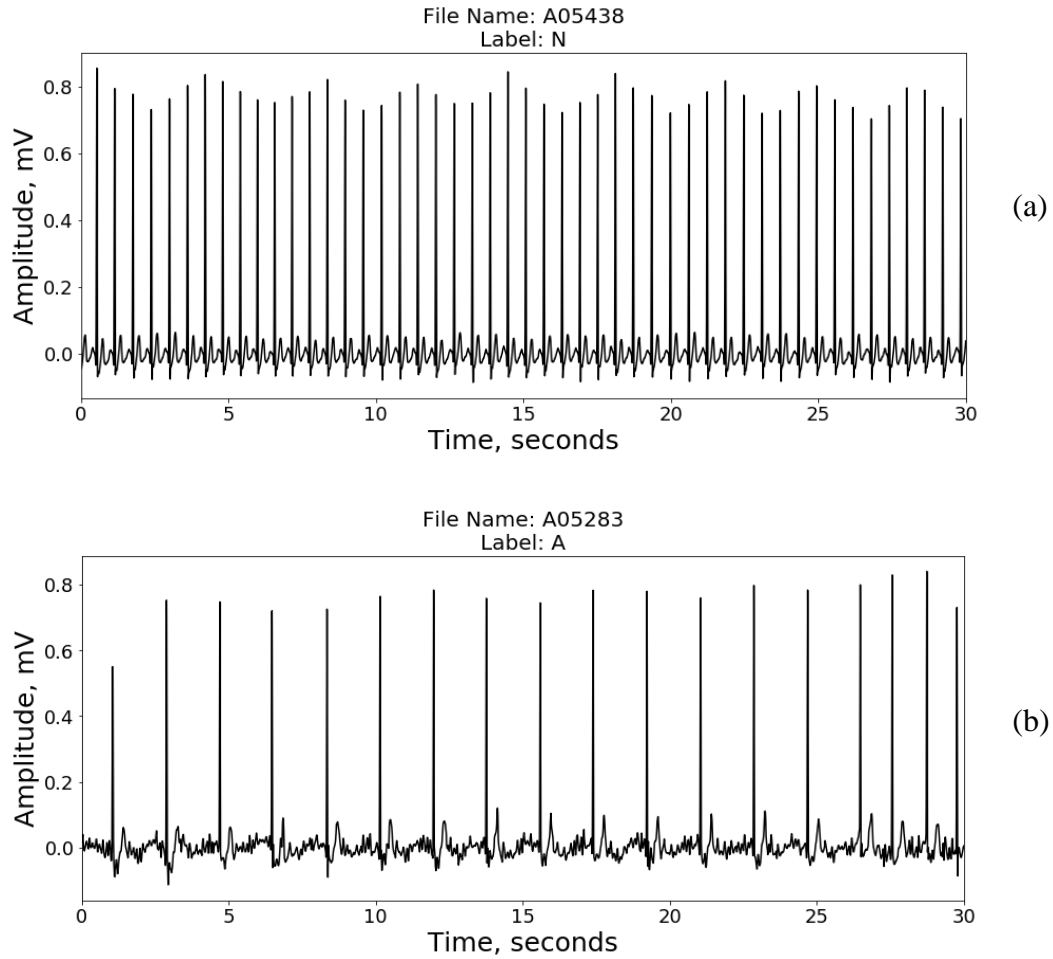


Figure 3.30: Heart Rate variability between (a) patient 1 and (b) patient 2

The data-frame dimensions are defined by the length of the longest R-R vector n_{max} times the number of records. The patients who has an R-R vectors which length is less than n_{max} will result in a missing data in the data-frame Figure 3.31(b).

	0	1	2	3	4	...	130	131	132	133	134	135	136	137	138	139
P1	334.0	363.0	345.0	365.0	345.0	366.0	337.0	307.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
P2	207.0	210.0	280.0	206.0	272.0	276.0	238.0	324.0	205.0	208.0	238.0	195.0	234.0	245.0	NaN	NaN
P3	76.0	77.0	150.0	243.0	55.0	117.0	143.0	129.0	87.0	114.0	185.0	158.0	221.0	214.0	355.0	207.0
P4	268.0	274.0	276.0	282.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	286.0	290.0	292.0	298.0	290.0	289.0	294.0	294.0	298.0	295.0	293.0	297.0	295.0	299.0	292.0	290.0
	231.0	230.0	234.0	235.0	240.0	238.0	223.0	214.0	212.0	218.0	217.0	214.0	209.0	214.0	224.0	230.0
	250.0	257.0	245.0	241.0	244.0	262.0	258.0	263.0	270.0	261.0	246.0	238.0	NaN	NaN	NaN	NaN
	391.0	323.0	311.0	314.0	353.0	317.0	294.0	319.0	345.0	403.0	343.0	318.0	333.0	360.0	311.0	328.0

(a)

	0	1	2	3	4	...	130	131	132	133	134	135	136	137	138	139
P1	334.0	363.0	345.0	365.0	345.0	366.0	337.0	307.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
P2	207.0	210.0	280.0	206.0	272.0	276.0	238.0	324.0	205.0	208.0	238.0	195.0	234.0	245.0	NaN	NaN
P3	76.0	77.0	150.0	243.0	55.0	117.0	143.0	129.0	87.0	114.0	185.0	158.0	221.0	214.0	355.0	207.0
P4	268.0	274.0	276.0	282.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	286.0	290.0	292.0	298.0	290.0	289.0	294.0	294.0	298.0	295.0	293.0	297.0	295.0	299.0	292.0	290.0
	231.0	230.0	234.0	235.0	240.0	238.0	223.0	214.0	212.0	218.0	217.0	214.0	209.0	214.0	224.0	230.0
	250.0	257.0	245.0	241.0	244.0	262.0	258.0	263.0	270.0	261.0	246.0	238.0	NaN	NaN	NaN	NaN
	391.0	323.0	311.0	314.0	353.0	317.0	294.0	319.0	345.0	403.0	343.0	318.0	333.0	360.0	311.0	328.0

(b)

Figure 3.31: Non-uniform R-R interval vectors (a) Resulting Data-Frame, (b) Non-uniform Length – Missing Data

The method we used is to amputate the columns that reaches a certain threshold of missing values (NaN), for illustrating, we take an example of the threshold equals to three. In our case the data-frame will be amputated starting from column 134 as it reached the threshold Figure 3.32(a), and the new data-frame will contain 133 columns only Figure 3.32(b), if we change the threshold the number of columns amputated will change accordingly.

	0	1	2	3	4	...	130	131	132	133	134	135	136	137	138	139
P1	334.0	363.0	345.0	365.0	345.0	366.0	337.0	307.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
P2	207.0	210.0	280.0	206.0	272.0	276.0	238.0	324.0	205.0	208.0	238.0	195.0	234.0	245.0	NaN	NaN
P3	76.0	77.0	150.0	243.0	55.0	117.0	143.0	129.0	87.0	114.0	185.0	158.0	221.0	214.0	355.0	207.0
P4	268.0	274.0	276.0	282.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	286.0	290.0	292.0	298.0	290.0	289.0	294.0	294.0	298.0	295.0	293.0	297.0	295.0	299.0	292.0	290.0
	231.0	230.0	234.0	235.0	240.0	238.0	223.0	214.0	212.0	218.0	217.0	214.0	209.0	214.0	224.0	230.0
	250.0	257.0	245.0	241.0	244.0	262.0	258.0	263.0	270.0	261.0	246.0	238.0	NaN	NaN	NaN	NaN
	391.0	323.0	311.0	314.0	353.0	317.0	294.0	319.0	345.0	403.0	343.0	318.0	333.0	360.0	311.0	328.0

(a)

	0	1	2	3	4	...	130	131	132	133			
P1	334.0	363.0	345.0	365.0	345.0	366.0	337.0	307.0	NaN	NaN	NaN	NaN	(b)
P2	207.0	210.0	280.0	206.0	272.0	276.0	238.0	324.0	205.0	208.0	238.0	195.0	
P3	76.0	77.0	150.0	243.0	55.0	117.0	143.0	129.0	87.0	114.0	185.0	158.0	
P4	268.0	274.0	276.0	282.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	286.0	290.0	292.0	298.0	290.0	289.0	294.0	294.0	298.0	295.0	293.0	297.0	
	231.0	230.0	234.0	235.0	240.0	238.0	223.0	214.0	212.0	218.0	217.0	214.0	
	250.0	257.0	245.0	241.0	244.0	262.0	258.0	263.0	270.0	261.0	246.0	238.0	
	391.0	323.0	311.0	314.0	353.0	317.0	294.0	319.0	345.0	403.0	343.0	318.0	

Figure 3.32: Data Preprocessing (2nd Phase) (a) Amputation, (b) Adjusted Data-Frame

After applying the amputation some missing values still exist in the columns that didn't reach the threshold. We propose two methods to treat this; (1) filling the missing values with a chosen value, (2) applying an extrapolation technique to complete the data vectors. In our real case we have selected a threshold equal to 800 missing values to amputate. A total of 53 columns are left in the data-frame, we replaced then the missing values with the value 250 Figure 3.33, which is the average R-R interval for a healthy adult (counted in samples, sample rate is 300/s).

	0	1	2	3	4	5	6	7	8	9	...	43	44	45	46	47	48	49	50	51	52	
P1	0	215	218	237	243	231	239	244	242	234	240	...	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	
P2	0	306	301	305	318	305	315	321	308	108	213	...	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	
P3	0	223	231	232	236	236	227	214	211	217	223	...	223.0	210.0	206.0	210.0	214.0	216.0	202.0	201.0	209.0	140.0
...	0	286	277	323	358	309	342	403	253	215	224	...	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	
	0	251	255	275	297	310	316	306	305	313	316	...	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	
	0	309	305	301	300	301	292	285	287	291	302	...	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	
	0	145	139	204	175	212	173	261	216	159	194	...	184.0	201.0	213.0	250.0	250.0	250.0	250.0	250.0	250.0	
	0	318	334	343	342	338	348	372	382	365	354	...	378.0	361.0	344.0	330.0	333.0	342.0	250.0	250.0	250.0	
	0	184	240	117	178	180	181	183	184	184	183	...	201.0	197.0	250.0	250.0	250.0	250.0	250.0	250.0	250.0	

Figure 3.33: Real case data-frame After amputation and filling

Using this method, we notice that the value 250 can be irrelevant to the other values in the data frame which causes a problem, when injected to the CNN, of correlation between

samples. Figure 3.34 shows that some patients have different R-R interval average than the chosen value, as it depends on many factors like the age, the health condition and even the emotional condition the patient is going thru, which directly affects the heart rate.

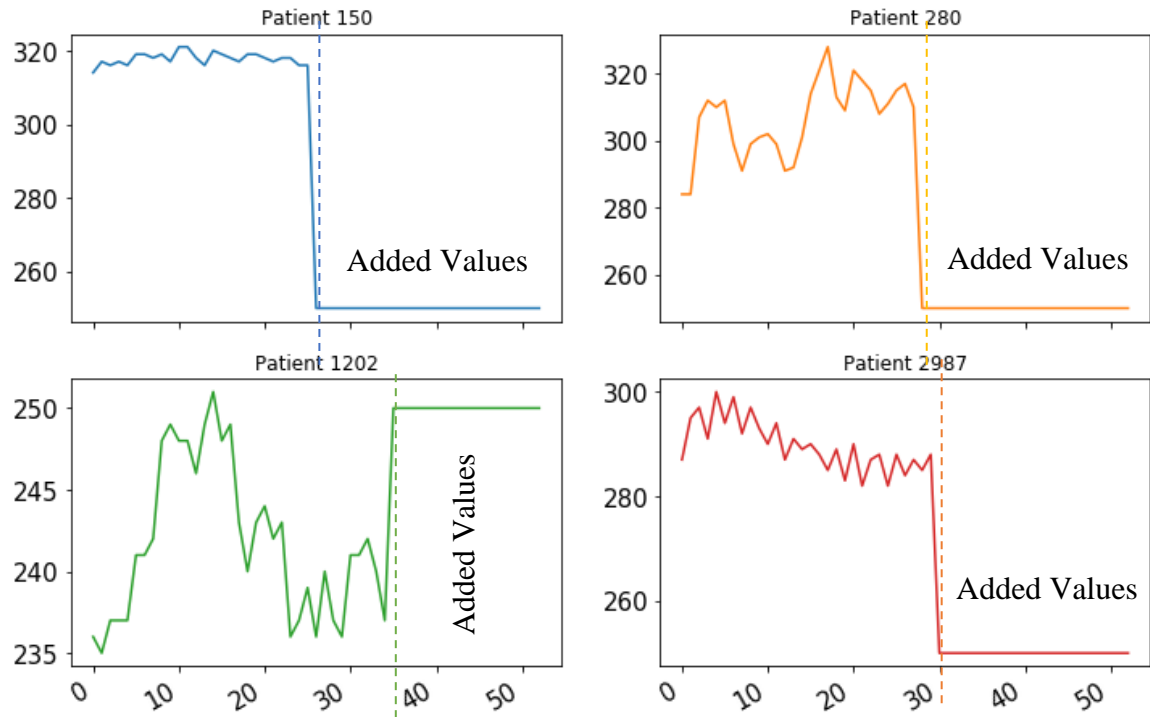


Figure 3.34: R-R interval with Added values

In order to treat this issue, a lot of methods exists. Statistical solutions like replacing with average or mean values of the signal can be applied, but in some cases, it will not be reliable just as the replacement with a given constant number. To overcome this problem, a dynamic solution (extrapolation method) has been proposed.

Extrapolation is an estimation of a value based on extending a known sequence of values beyond the area that is certainly known. To achieve this, many techniques can be used namely the BSpline and the Cubic Spline algorithms.

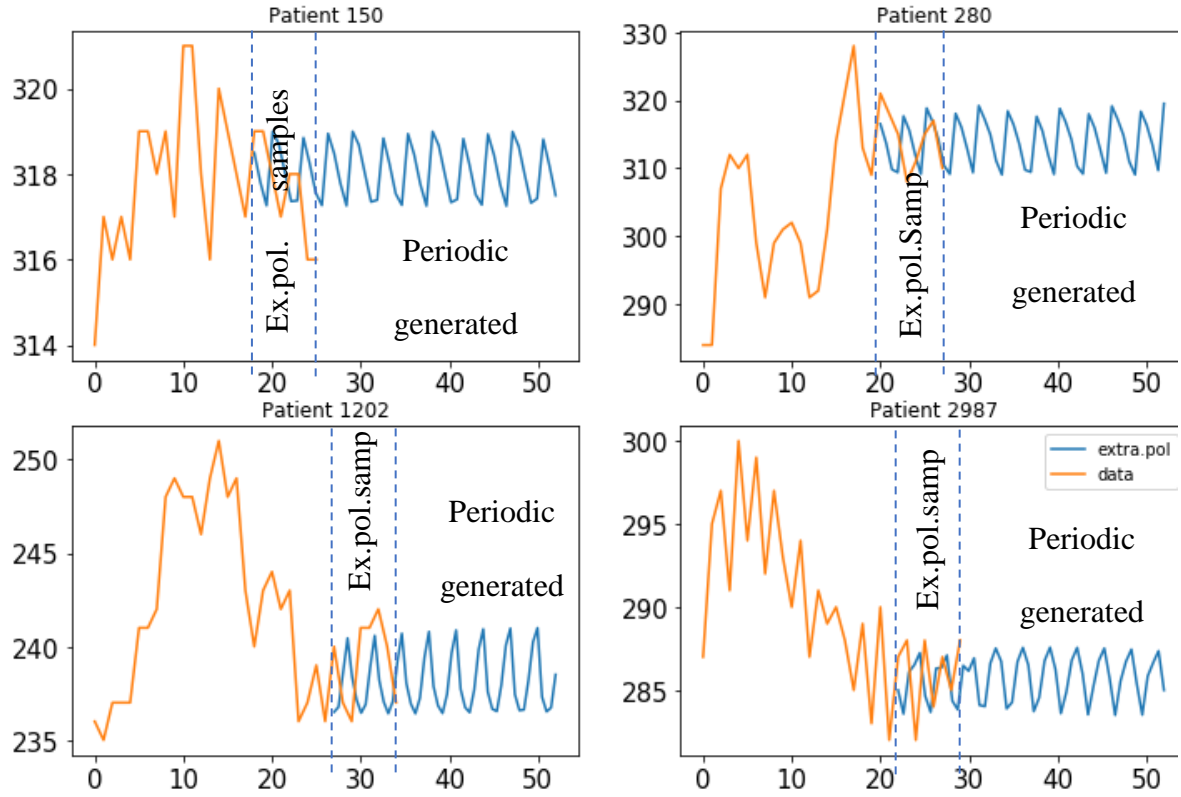


Figure 3.35: Periodic B-Spline Extrapolation (2nd order, over 8 samples)

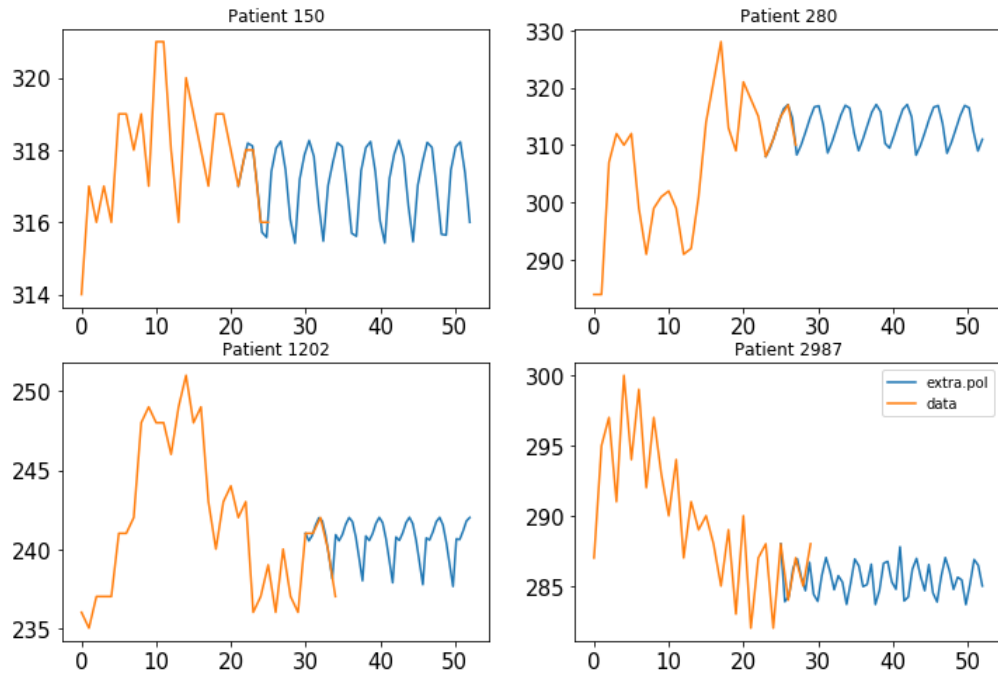


Figure 3.36: Periodic Cubic Spline Extrapolation (over 5 samples)

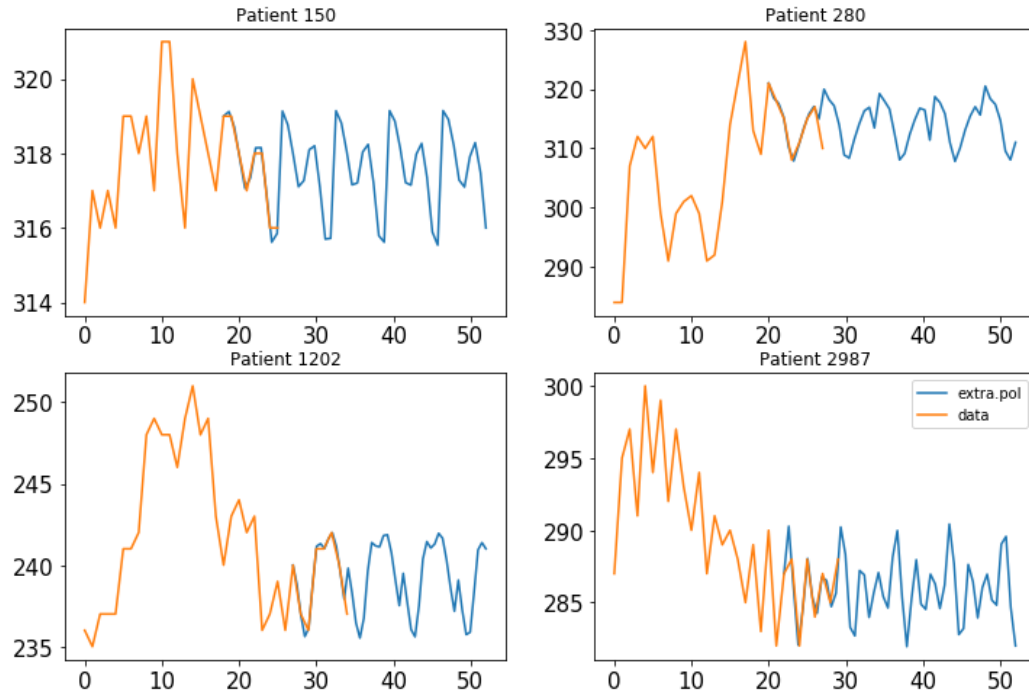


Figure 3.37: Periodic Cubic Spline Extrapolation (over 8 samples)

Furthermore, we have tried to extrapolate the signals using a sliding window technique, we take a fixed number of samples (n) to generate one point at a time, then we slide the window including the new generated point into the vector and releasing one point from the other side, and we calculate another point. The signal we get is illustrated in Figure 3.38

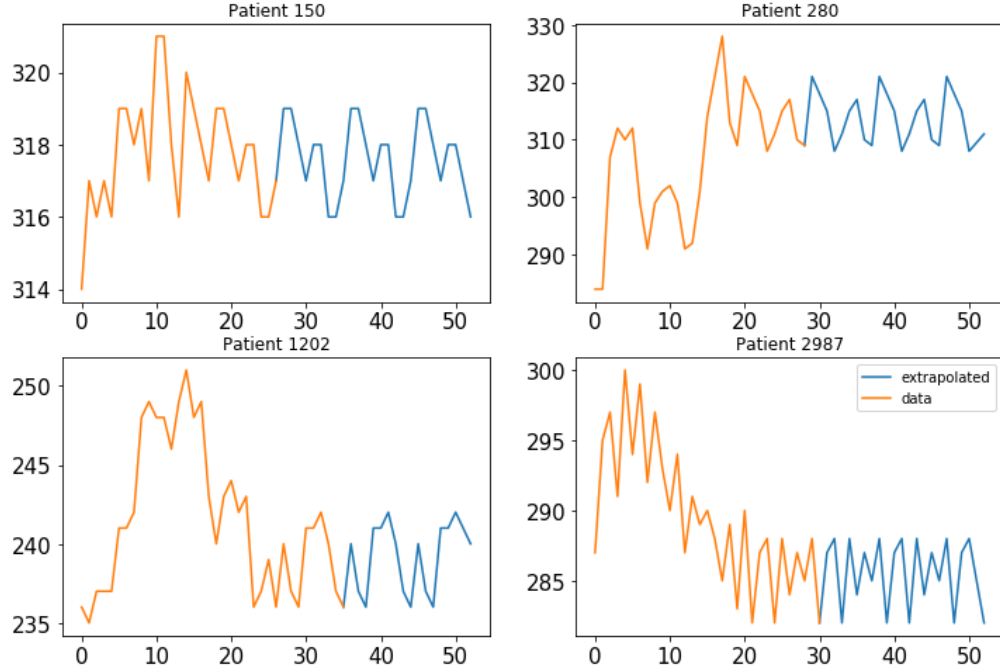


Figure 3.38: Window Periodic Cubic Spline Extrapolation (over 8 samples)

These extrapolated signals have been injected to the CNN but the accuracy have dropped to 85%. We explain this drop by the fact that this difference between the actual data and the manually added values is actually detected by the Neural Network as a feature.

b- Convolutional Neural Network (CNN)

Convolution is a function derived from two other functions by integration which describes how one of them modifies the other. The mathematical expression of convolution is defined as follows:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

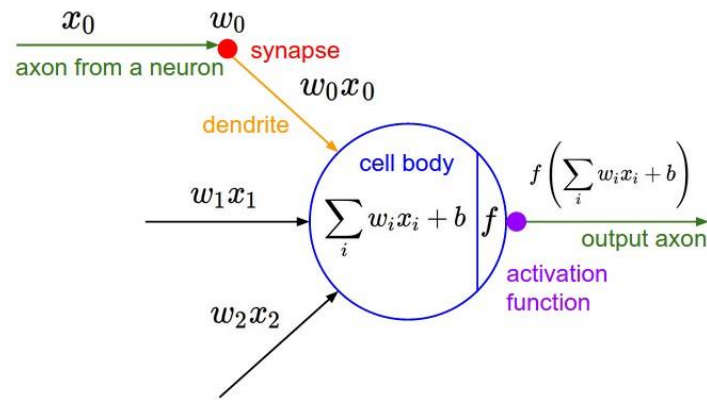


Figure 3.39: Neuron Cell - CNN equation

A Convolutional Neural Network (CNN) applies filters (also called kernels or feature detectors) to the input to detect patterns and extract a feature map. Different types of CNNs exist, depending on the application we can choose between one, two or three dimensional CNN. The key difference is the dimensionality of the input data and how the feature detector (or filter) slides across the data. In the one dimensional CNN the filters run thru the data row by row (assuming that the data is a table) where the selected data's dimension is one by X where X is the number of features, but in two dimensional CNN the filters go matrix by matrix with two dimensions, X and Y.

CNNs have two main parts:

- A convolution/pooling mechanism that breaks up the image into features and analyzes them
- A fully connected layer that takes the output of convolution/pooling and predicts the best label to describe the image

Convolutional Neural Networks have several types of layers:

- Convolutional layer: a “filter” passes over the data (image, data table, matrix, etc..), scanning a few data samples at a time and creating a feature map that predicts the class to which each feature belongs.
- Pooling layer (downsampling): reduces the amount of information in each feature obtained in the convolutional layer while maintaining the most important information (there are usually several rounds of convolution and pooling).
- Fully connected input layer (flatten): takes the output of the previous layers, “flattens” them and turns them into a single vector that can be an input for the next stage.
- Fully connected output layer: gives the final probabilities for each label.

The proposed CNN’s architecture is as follows:

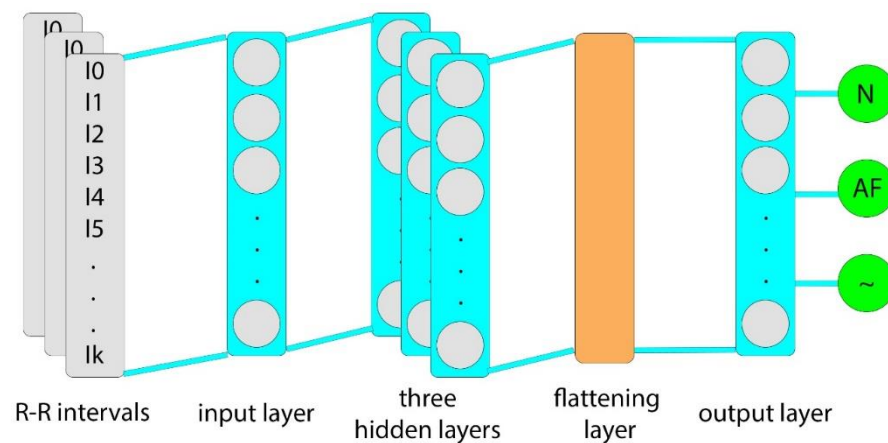


Figure 3.40: Proposed CNN's architecture

The model we are proposing is based on a one dimensional Convolutional Neural Network as shown in Figure 3.40. After the pre-processing and feature extracting method

explained in the next section, the resulting R-R records will be injected into the input convolutional layer. After it three hidden layer are applied with respectively 100, 150 and 100 nodes. The input and the first hidden layer are activated using the Rectified Linear Unit function (ReLU), the second and third hidden layers are activated using nonlinear activation (sigmoid). Then a flattening function is applied to convert the 2 dimensional output of the previous layer to a single dimension array to inject into a fully connected output layer. A pooling layer has been added between the hidden layers to help speed the learning procedure without affecting the accuracy.

Chapter 4. Experimental results

4.1 Platform

Three versions of the electronic platform have been designed and tested in terms of cost, size/mobility, number of supported sensors, scalability, connectivity, frequency and processing power. The following table shows the difference between each version and highlights the upgrades from one version to another.

Table 4.1: Comparative study between the platform versions

	Version 1	Version 2	Version 3
Cost	\$26.95 * 2	\$423,25 (1)	\$21.95
Size	Small	Bulky	Medium
Mobility	Needs a Computer	Needs a Computer(2)	Standalone
Supported sensors	wired	Wired/BLE	Wired/BLE
Scalability	4 channels	12 channels	14 channels
Connectivity	BLE	BLE/WIFI(2)	BLE/WIFI
Frequency	50Hz	16MHz	240MHz
Processor	HCS08	atmega328	Xtensa LX6 32bit Dual Core

(1) Without the Arduino board.

(2) Wifi module is not included and needs to be added separately, coexistence with Bluetooth is not guaranteed.

4.2 Machine Learning Model

We tested the model over different configurations, the accuracy and F1 score results goes from 74% to 96%, which is still reasonable values. The choice of optimizer seems to strongly affect the stability and the accuracy of the model. We decided to run few tests to demonstrate the effect of some optimizers on the performance of the model. We also changed the amputation threshold and the padding value in the pre-processing phase during the test.

Evaluation metrics:

Precision is the number of True Positives divided by the number of True Positives and False Positives.

$$Precision = \frac{TP}{TP + FP}$$

Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV).

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives.

$$Recall = \frac{TP}{TP + FN}$$

Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate.

F1 Score is the $2 * ((precision * recall) / (precision + recall))$. It is also called the F Score or the F Measure.

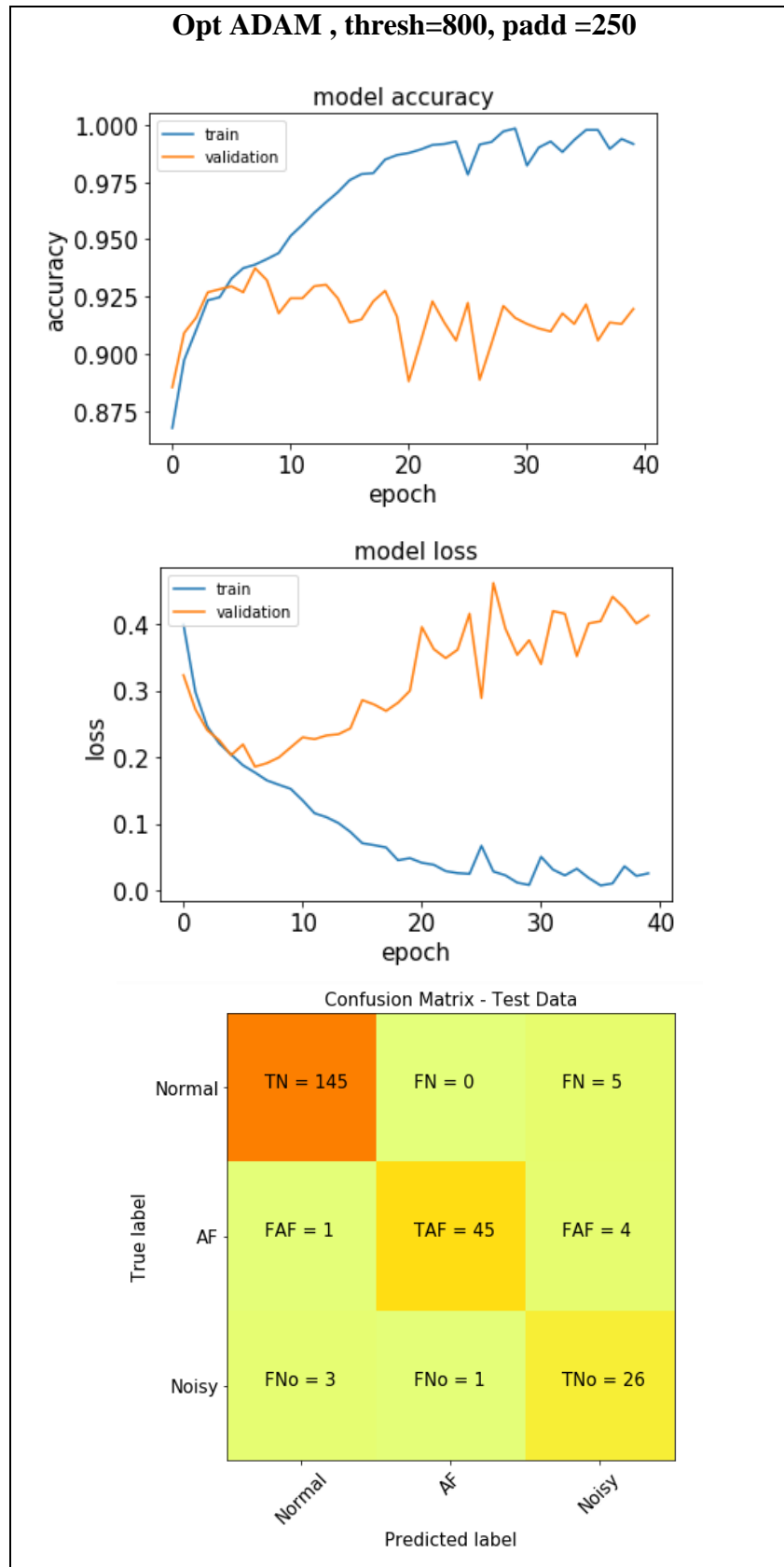
$$F1\ Score = 2 \frac{Precision * Recall}{Precision + Recall}$$

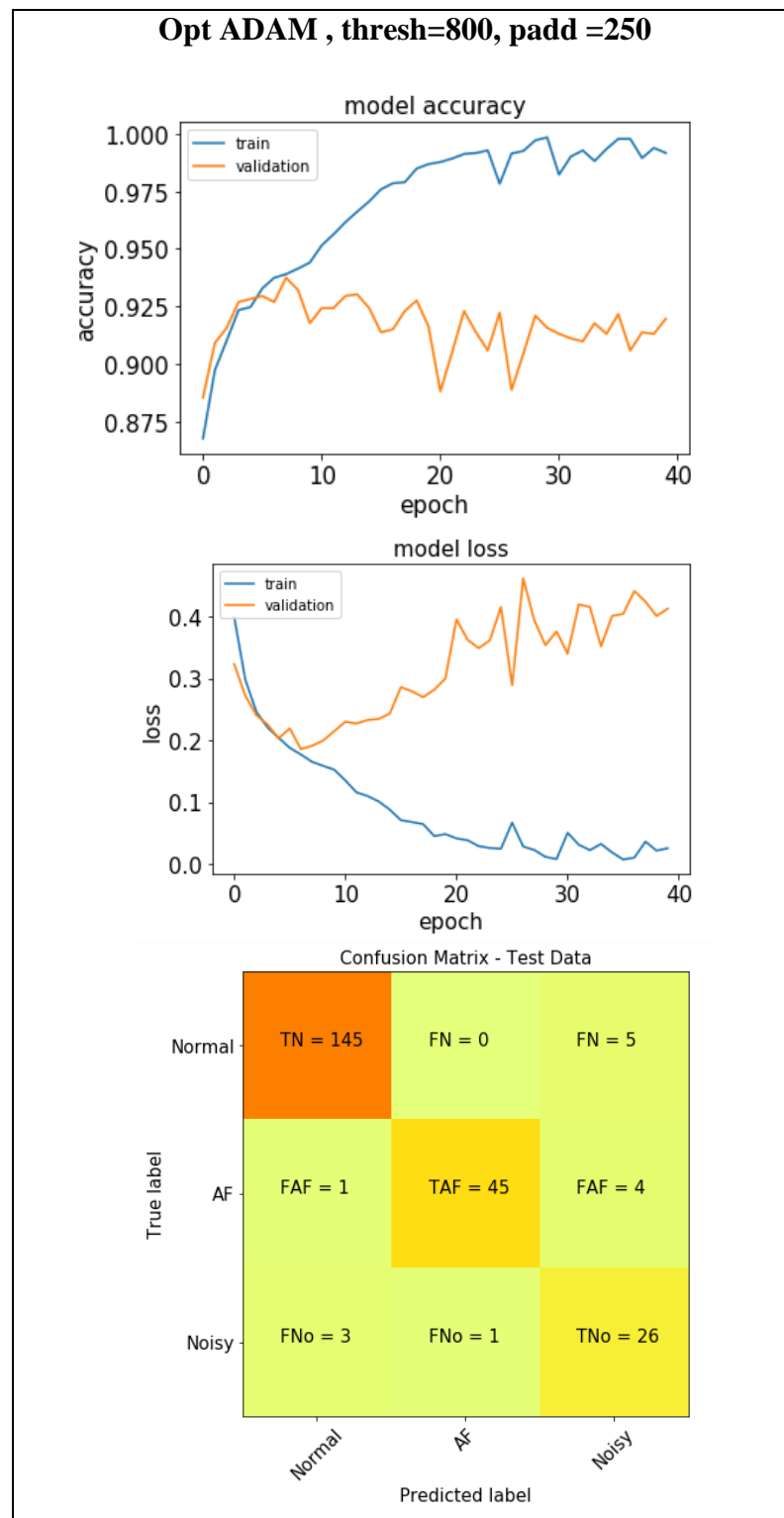
Put another way, the F1 score conveys the balance between the precision and the recall

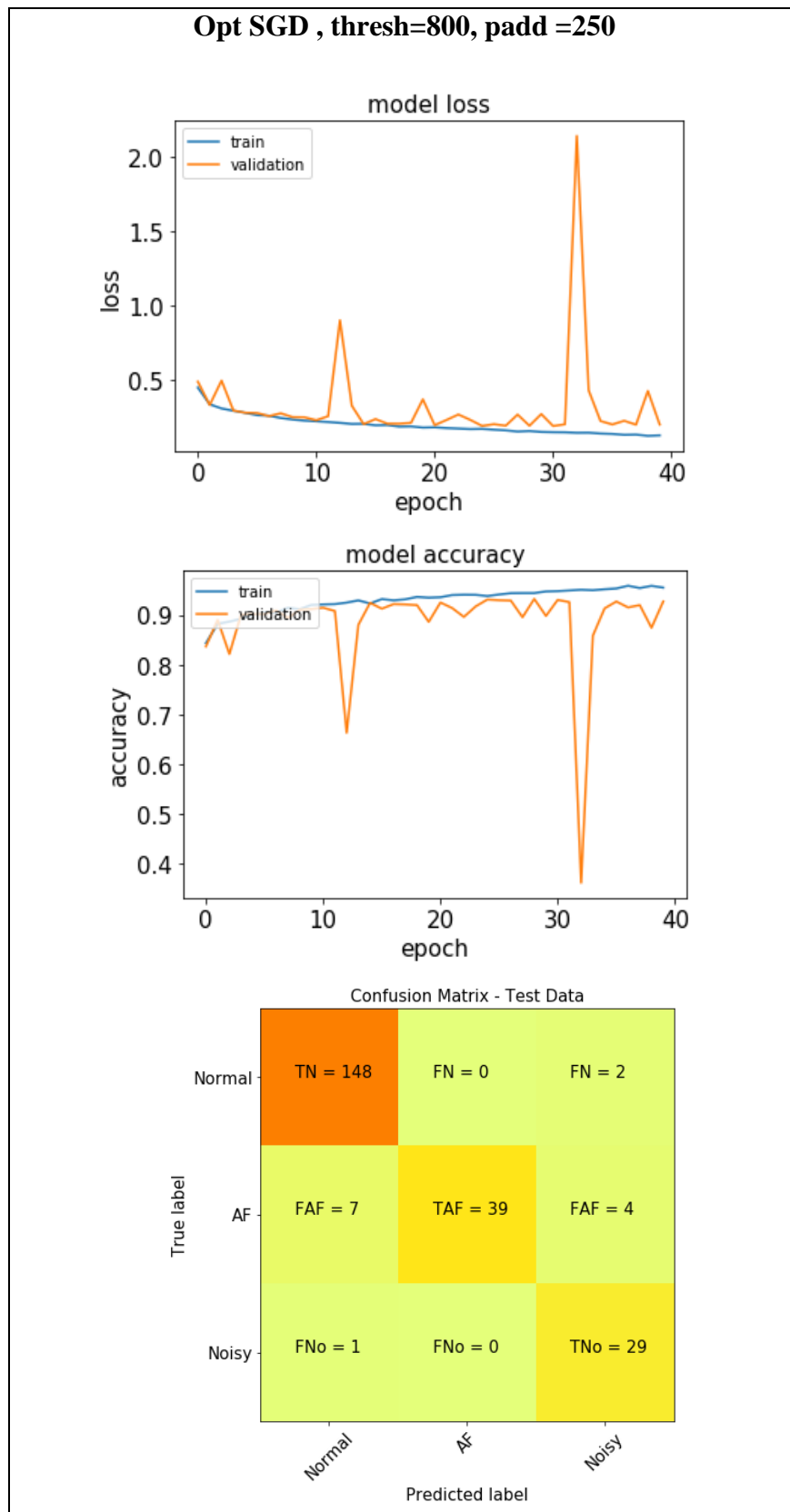
Stochastic gradient descent (often abbreviated SGD) optimizing method for machine learning, It can be regarded as a stochastic approximation of gradient descent optimization, the difference is that it replaces a normal gradient descent (calculated from the entire data set) and calculates an estimate thereof (calculated from a randomly selected subset of the data).

Adaptive Moment Estimation Algorithm (Adam), this algorithm simply estimates moments and uses them to optimize a function. It is essentially a combination of the gradient descent with momentum algorithm (AdaGrad) and the RMS (Root Mean Square) Prop algorithm. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training where in Adam the parameter learning rates are not calculated based on the average first moment (the mean) as in RMSProp, it also makes use of the average of the second moments of the gradients (the uncentered variance).

In the next section we will see the difference between the two optimizers.







In

the first

test using ADAM optimizer, 800 as a threshold and the value 250 for padding, we notice a gap between the training accuracy ~99% and the validation accuracy ~91% which means the model is over fitting. The second test seems to have slightly better results after changing the threshold to 700, the accuracy of training is ~98% and the validation is ~92%. In the third test, and by changing the optimizer to SGD, the threshold to 800 and the padding to 250, the accuracy of training is ~98% and the validation accuracy is ~96% which is acceptable. We notice some spikes in the validation accuracy, the sensitivity of the model is sometimes unstable with this particular optimizer. The F1 scores which represent the well predicted and miss predicted samples are displayed in the confusion matrix.

Also some tests have been made with different number of patients, and with different preprocessing methods and with different classifiers. Table 4.1 illustrates the results:

Table 4.2: Test results

	Training set	Testing set	Classes	PreProc Method	Training Accuracy	Test Accuracy	Classifier
Test 1	589	197	2 Normal AF	Same duration signals (30s) RR Intervals + sample cutting (thresh=1)	99.66%	95.43%	CNN
Test2	605	202	3 Normal AF Noisy	Same duration signals (30s), RR Intervals extraction + sample cutting (thresh=1)	99.83%	90,59%	CNN
Test3	816	272	3 Normal AF Noisy	Different durations signals, RR Interval extraction + sample cutting (thresh=800) + NaN replacing (250)	99.51%	91.18%	CNN
Test4	5971	230	3 Normal AF Noisy	Different durations signals, RR Interval extraction + sample cutting (thresh=800) + NaN replacing (250)		82.93%	SVM
Test5	5971	230	3 Normal AF Noisy	Different durations signals, RR Interval extraction + sample cutting (thresh=800) + NaN replacing (250)	88.58 % 88.36 % 86.96%	85.24% K: 3 85.77% K: 4 85.38% K: 5	KNN

Chapter 5. Conclusion and perspectives

In this work, we showed the developing process of a very cost-effective mobile health monitoring system. The platform has successfully passed the real case usage tests. A machine-learning model has been developed and successfully tested. An accuracy of 96% on the testing data set has been reached yet many improvements can be done.

The platform we have built has been proven to be reliable and efficient, yet it can be modified to be more flexible. As the platform is scalable, many sensors need to be supported by the electronic card. The memory limitation on the card can be solved by adding an external memory. With that done the method of the pipeline data acquisition and uploading will be very effective, and a high sampling frequency will be reached which mean a higher resolution graphs. Also the additional storage on the device will give the possibility to embed another feature, the possibility to record offline will be possible and the patient will not be obliged to have internet connection at the time. The external memory card also opens the doors for major changes of the system architecture, as the data can be logged and stored locally, the system can be set not to transmit at the acquisition time even with the presence of internet connection, yet a specific uploading time can be set and the architecture can be modified to form a data warehouse system where multiple yet similar databases uploads to a large universal database.

A major improvement can be held on the architecture of the device as it is possible to use systems on chips including the Wi-Fi and Bluetooth connectivity and ADC on an electronic chip with the size of a couple of centimeters. Such improvement in size will

definitely improve the power consumption as well. The used esp32 Thing card is actually based on a System On Chip (SOC) ESP32-D0WDQ6 which embeds various modules; CPU, memory and registers, WIFI and Bluetooth, encryption module, ADC, dedicated sensors interfaces. The chip is a couple of millimeters in diameter, but the card around it, which holds the cp2104 USB to UART chip necessary for uploading programs to the chip via USB, a power regulator and filter, a Lipo battery charger chip and a couple of switches with some necessary capacitors and resistors makes it's dimensions much bigger. By adding the accelerometer and the sockets for the wired sensors we make even bigger, of course the prototype we have is just for demonstrating purposes, yet we plan to work more on developing a much smaller card that will be proper for usage.

Speaking of the internal architecture of the ESP32 SOC we notice it embeds an Ultra-Low Power Co-processor connected directly to the ADC and separated from the main cores figure 5.1. The ULP CoCPU has access to the internal memory as well, so it can sample and log data separately from the main CPU. This can give the ability to divide the uploading time by assigning both main cores on the uploading task.

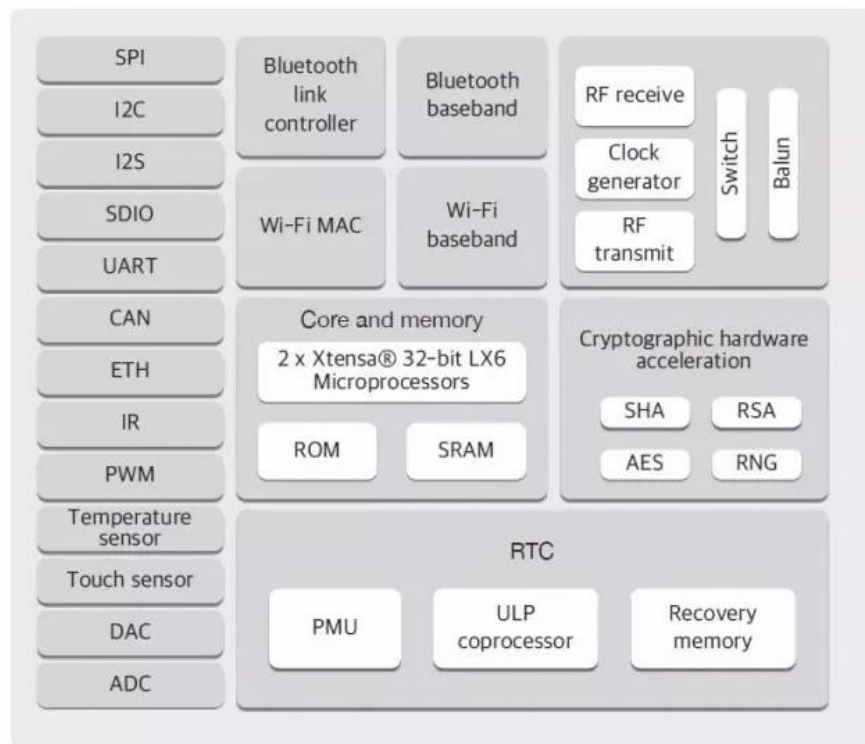


Figure 5.1: ESP32 SOC architecture

References

- [1] K. A., "Oman health: Cardiovascular disease is top killer of Omanis," Times of oman, 4 January 2017. [Online]. Available: <https://timesofoman.com/article/99943>. [Accessed 12 09 2019].
- [2] JulioMarti-Almor, "Incidence of sleep apnea and association with atrial fibrillation in an unselected pacemaker population: Results of the observational RESPIRE study," *Elsevier*, 2019.
- [3] Y. X. Y. Fu, "Meta-analysis of all-cause and cardiovascular mortality in obstructive sleep apnea with or without continuous positive airway pressure treatment," *elsevier*, 2017.
- [4] S. W. and al, "Obstructive sleep apnea is associated with nonsustained ventricular tachycardia in patients with hypertrophic obstructive cardiomyopathy," *elsevier*, 2019.
- [5] "Heart Diseases & Disorders," [Online]. Available: <https://www.hrsonline.org/heart-diseases-disorders>.
- [6] w. Mark and J. B., "Designing Calm Technology," [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.9788&rep=rep1&type=pdf>.
- [7] M. Mokhinabonu and al., "Review of Wearable Device Technology and Its Applications to the Mining Industry," *MDPI Open Access Journal*, 2018.
- [8] D. R. S. and al., "Wearable sensors for monitoring the internal and external workload of the athlete," *US National Library of Medicine*, 2019.
- [9] AyaskantaMishra, "REMOTE WEB BASED ECG MONITORING USIMQTT PROTOCOL FOR IOT IN HEALTHCARE," *International Journal of Engineering Research & Technology (IJERT)*, vol. Volume 5, 2018.
- [10] AyaskantaMishra, "AD8232 based Smart Healthcare System using Internet of Things (IoT)," *International Journal of Engineering Research & Technology (IJERT)*, vol. 7, no. 4, 2018.
- [11] B. N. and al., "Heart Beat Monitoring System And Security Using Android," *International Journal of Advanced Research in Computer Science*, 2017.
- [12] R.Harini, B.Rama and al., "DEVELOPMENT OF ECG MONITORING SYSTEM USING ANDROID APP," *International Journal of Electronics and Electrical Engineering*, 2017.
- [13] A. Maradugu and al., "Android Based Health Care Monitoring System," *IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems*, 2015.
- [14] K. Antczak, "Deep Recurrent Neural Networks for ECG Signal Denoising," *Cornell University*, 2018.
- [15] F. A. and al., "Comparing Feature-Based Classifiers and Convolutional Neural Networks to Detect Arrhythmia from Short Segments of ECG," *IEEE Computing in Cardiology*, 2017.
- [16] J. Pan and W. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on*

Biomedical Engineering, Vols. BME-32, no. 3, p. 230–236, 1985.

- [17] E. H. and al., "ECG signals classification: a review," *International Journal of Medical Engineering and Informatics*, 2017.
- [18] a. Shenda H., "ENCASE: an ENsemble ClASSifiEr for ECG Classification Using Expert Features and Deep Neural Networks," *Computing in Cardiology*, vol. 44, 2017.
- [19] a. Muhammad Z., "An Automated ECG Beat Classification System Using Convolutional Neural Networks," *IEEE*, 2016.
- [20] J. A. and al., "Rhythm and Quality Classification from Short ECGs Recorded Using a Mobile Device," *IEEE Computing in Cardiology*, 2017.
- [21] B. Lucia and al., "Detection of AF and Other Rhythms Using RR Variability and ECG Spectral Measures," *IEEE Computing in Cardiology*, 2017.
- [22] G. B. and al., "Detection of Atrial Fibrillation Using Decision Tree Ensemble," *Computing in Cardiology*, 2017.
- [23] B. Pietro and al., "Detection of Atrial Fibrillation Episodes from Short Single Lead Recordings by Means of Ensemble Learning," *Computing in Cardiology*, 2017.
- [24] B. Chandra and al., "Atrial Fibrillation Detection Using Convolutional Neural Networks," *IEEE Computing in Cardiology*, 2017.
- [25] I. C. and al., "Multi-parametric Analysis for Atrial Fibrillation Classification in ECG," *Computing in Cardiology*, 2017.
- [26] E. E. C. and al., "Atrial Fibrillation Classification from a Short Single Lead ECG Recording Using Hierarchical Classifier," *Computing in Cardiology*, vol. 44, 2017.
- [27] M. D. S. and al., "Combining Template-based and Feature-based Classification to Detect Atrial Fibrillation from a Short Single Lead ECG Recording," *Computing in Cardiology*, vol. 44, 2017.
- [28] DIGI, "Xbee S2C DataSheet," [Online]. Available: <http://www.digi.com/products/xbee-rf-solutions/rf-modules/xbee-zigbee>.
- [29] Arduino, Arduino Uno Datasheet, www.content.arduino.cc.
- [30] Renesas, "Bluetooth low energy Protocol Stack Introduction," 2017. [Online]. Available: <https://www.renesas.com/eu/en/img/solutions/key-technology/connectivity/bluetooth-smart/r01qs0014ej0100-bleintro.pdf>.
- [31] renesas, "Bluetooth Low Energy Protocol Stack Pulse Oximeter Profile," 2017. [Online]. Available: <https://www.renesas.com/us/en/doc/products/mpumcu/apn/r178/002/r01an3738ej0100-g1dplxp.pdf>.
- [32] N. Jia, "Detecting Human Falls with a 3-Axis Digital Accelerometer," 2009. [Online]. Available: <https://www.analog.com/media/en/analog-dialogue/volume-43/number-3/articles/detecting-falls-3-axis-digital-accelerometer.pdf>.
- [33] G. D. and al., "AF Classification from a short single lead ECG recording: the

- PhysioNet/Computing in Cardiology Challenge," *IEEE Computing in Cardiology*, 2017.
- [34] J. Eva and al, "Biomedical Image Volumes Denoising via the Wavelet Transform," *IntechOpen*, 2011.
 - [35] P. Hamilton, "Open Source ECG Analysis," *Computers in Cardiology*, 2002.
 - [36] A. L. and al., "Development of a Cost-Effective ECG Monitor for Cardiac Arrhythmia Detection using Heart Rate Variability," *Biomedical Engineering International Conference*, 2016.
 - [37] A. Hossen and al., "Investigation of heart rate variability of patients undergoing coronary artery bypass grafting (CABG)," *IOS Press Technology and Health Care*, 2017.
 - [38] A. Meliones and al., "Embedded Parallelism Enabling Ultralow-Power Zigbee," *Journal of Computer Networks and Communications*, 2019.
 - [39] J. Li and al., "Deep Convolutional Neural Network Based ECG Classification System Using Information Fusion and One-Hot Encoding Techniques," *Hindawi Mathematical Problems in Engineering*, 2019.
 - [40] B. M. Silva and al., "Mobile-health: A review of current state in 2015," *Journal of Biomedical Informatics* 56, 2015.
 - [41] M. S.I. and al., "Monitoring of the Human Body Signal through the Internet of Things (IoT) Based LoRa Wireless Network System," *MDPI applied science*, 2019.
 - [42] R. Y. R. and al., "Physiological Wireless Sensor Nodes: Lifetime and power Consumption Evaluation for Data Intensive Transmission," *IEEE*, 2017.
 - [43] A. A.E. and al., "ZigBee Network Using Low Power Techniques and Modified LEACH Protocol," *IEEE*, 2017.
 - [44] H. K. and al., "Performance Evaluation of Bluetooth Low Energy Technology under Interference," *13th EAI International Conference on Body Area Networks*, 2018.
 - [45] M. k. and al., "Remote health monitoring of elderly through wearable," *springer*, 2019.
 - [46] a. Yinsheng Ji, "Electrocardiogram Classification Based on Faster," *MDPI SENSORS*, 2019.
 - [47] a. Serkan K., "Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664 - 675, 2016.