الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــــة

Faculté : Sciences de l'ingéniorat

Département : Electronique

Domaine : Sciences et Technologie

Filière : Electronique

Spécialité : Réseaux et Télécommunications

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème:

IOT IN HEALTH –HEART MONITORING

Présenté par : *DJENDLI Maroua*

Encadrant : *FIZZARI Mohamed* *Prof* *BADJI MOKHTAR ANNABA*

Jury de Soutenance :

BOUKARI Karima	MCA	BADJI MOKHTAR ANNABA	Président
FIZZARI Mohamed	Prof	BADJI MOKHTAR ANNABA	Encadrant
MESSADEG Djemil	Prof	BADJI MOKHTAR ANNABA	Examinateur

Année Universitaire : 2019/2020

Dedication

Words can never express my deep love and gratitude to the two men of my life: my father **"DIABI"** and my little brother **"BILLEL"**.

My deep love and profound affection go to my dear mother **"LEILA"** to whom I owe a great debt.

To my sisters **"CHAIMA"**, who believe in me and surround me with her care and love and to who I wish successful live.

To my sister **"KHAWLA"** and her husband **"HAMZA"** whom I thank for giving me love, support and encouragement.

My deep gratitude, respect and thanks are dedicated to my supervisor **Mr. FIZZARI**.

To my best friend **"MIRIEM ZROUG"** who never let me down.

To the two family **"DJENDLI"** and **"LABIDI"**, without forgetting my sisters' family **"BOUNASRI"**

To all my **colleges promotion 2015-2020**, specially my friends whom I share great moments

To every person who will have the occasion to read this modest work.

Djendli Marwa

ACKNOWLEDGMENTS

I thank our Almighty God who granted me the will, the patience, and especially the health throughout our course. .

I also thank **DJEMIL MESSADEG** for having accepted to chair the jury of our brief and I would like to thank for **BOUKARI KARIMA** the honor that we have to accept to judge this work.

I would like to thank all the family of the Electronic department and administrative body of the Faculty of Engineering Sciences and the University, for their support and encouragement with us during our study.

Djendli Marwa

ABSTRACT

In English

Our project is to make an IoT node for heart monitoring using a stethoscope and raspberry pi. This node collects the heart sound and sends it to our web application to visualize it in real-time. Also it is able to save a wave file in order to classify it with a Matlab into a normal or abnormal heart sound.

Key words : IoT, python , raspberry pi ,matlab ,pcg , node js

En Français

Notre projet est de créer un nœud IoT pour la surveillance cardiaque à l'aide d'un stéthoscope et d'un raspberry pi. Ce nœud collecte le son du cœur et l'envoie à notre application web pour le visualiser en temps réel. Il est également capable de sauvegarder un fichier wave afin de le classer avec un Matlab dans un son cardiaque normal ou anormal.

Mots clés: IoT, python, raspberry pi, matlab, pcg, node js

بالعربية

مشروعنا هو إنشاء عقدة إنترنت الأشياء لمراقبة القلب باستخدام سماعة الطبيب و Raspberry Pi. تجمع هذه العقدة صوت القلب وترسله إلى تطبيق الويب الخاص بنا لتصوره في الوقت الفعلي. كما أنه قادر على حفظ ملف الموجة لتصنيفه مع Matlab إلى صوت قلب طبيعي أو غير طبيعي.

الكلمات الأساسية: إنترنت الأشياء ، بيثون ، رازبيري باي ، ماتلاب ، بي سي جي ، عقدة جي إس

Table of contents

CHAPTER III :

CHAPTER VI :

FIGURE LISTE

TABLE LISTE

ABRIVIATION LISTE

IoT: Internet of things

IoMT: The Internet of Medical Things

EEG: electrocardiography

PPG: Photoplethysmography

PCG: Phonocardiography

LV : Left Ventricle

HF: Heart Failler

M1: Mitral

T1: Tricuspid

A2: Aortic

P2: Pulmonary

S1: First heart sound

S2:Second heart sound

S3:Third heart sound

S4:Forth heart sound

MFCC: Mel frequency cepstral coefficients

HS:Heart sound

KNN : k-nearest neighbor

SVM :vector machines

ML : Machine lesrning

NCA: Neighborhood component analysis

ANNABA UNIVERSITY

CHAPTER 1

General introduction

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as a pill to something as big as an aero plane, into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise unintelligent, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us smarter and more responsive, merging the digital and physical universes.

In this project we are interested in the iot in health which is called The Internet of Medical Things (IoMT). It is the collection of medical devices and applications that connect to healthcare IT systems through online computer networks. Medical devices equipped with Wi-Fi allow the machine-to-machine communication that is the basis of IoMT. IoMT devices link to cloud platforms such as Amazon Web Services, on which captured data can be stored and analyzed. IoMT is also known as healthcare IoT.

The major percent of people who needs the IoMT, all over the world , are the unhealthy and elderlly ones . Especially whom have heart problems , cause the heart is the most important mosel in the humain body . It has three main events or we may call them 'Biosignals' , which are : electrocardiography 'EEG', photoplethysmography 'PPG', and phonocardiography 'PCG'.

The last one is the basic biosignal in our project which gives us pure informations on the user's health .First of all ,it will be collected using some electroniques and medecals divisces .Than, it will pass throuat a specific preprocessing and processing channels, in order to give the doctors and the hospital's familly clear data which mean correct diagnosis of the patient's condition .That lead us finally to giving the patient his\her traitement without making a singal move.

I.1 Anatomy of Heart

The heart is a muscular organ, which pumps blood through the blood vessels of the circulatory system. Blood provides the body with oxygen and nutrients, as well as assisting in the removal of metabolic wastes.

In humans, the heart is located between the lungs, in the middle compartment of the chest**[1]**. It has four chambers:

- Two upper chambers are called atrial (Mitral in the left and Tricuspidin the right)
- Two lower chambers are called ventricles (Aortic in the left and Pulmonic in the right).

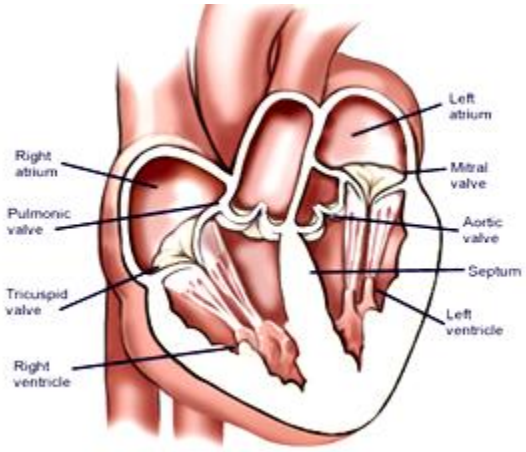


Figure I.1. Human's heart

I.2 The source of heart sounds

Also in human heart , there are four valves - one for each chamber of the heart. The mitral valve and tricuspid valve are located between the atria (upper heart chambers) and the ventricles (lower heart chambers).The aortic valve and pulmonic valve are located between the ventricles and the major blood vessels leaving the heart. These valves which are one-way doors , opens and closes periodically to permit blood flow in only one direction. Heart sounds (beating)are noises generated bya specific cardiac event. Specifically, the sounds created when the heart valves snap shut.

In cardiac auscultation, an examiner may use a stethoscope to listen for these unique and distinct sounds that provide important auditory data regarding the condition of the heart.

Figure I.2 Heart's valves

I.3 Normal heart sounds :

Noises S1 and S2 are the two main noises heard in an auscultationof a healthy subject. The interval between the first and second noise defines the systole(ventricular ejection), while the interval between the second and the first noisefollowing defines the diastole (ventricular filling).

I.3.1 First heart sound :

The first heart sound "**Lub**"corresponds to the myocardium'scontractionat the start of ventricular systole.It results from the closing of the mitral and tricuspid valves.This noise is characterized by its frequency content of low frequencies , compared toto the S2 noise of the same cardiac cycle.Its large proportion in termsof energy, generated by the closure of the atrioventricular valves (mitral andtricuspid).Cardiac noise S1 is made up of four groups of vibrations, as illustratedin figure I.3 :

1. Low amplitude and low frequency oscillations, not audible, theycorrespond to the onset of ventricular contraction.
2. Large amplitude, audible , oscillations corresponding to the closings of the mitral (M1) and tricuspid (T1) atrioventricular valves.
3. Large amplitude, audible , oscillations corresponding to the opening ofsigmoid(Sigmoid valves (Anat) the valves at the beginning of the aorta and of the pulmonary artery which prevent the blood from flowing back into the ventricle).
4. Low amplitude and low frequency oscillations, not audible, due toturbulent blood flow in the aorta and pulmonary artery, suddenly dilatedat the start of ventricular ejection.

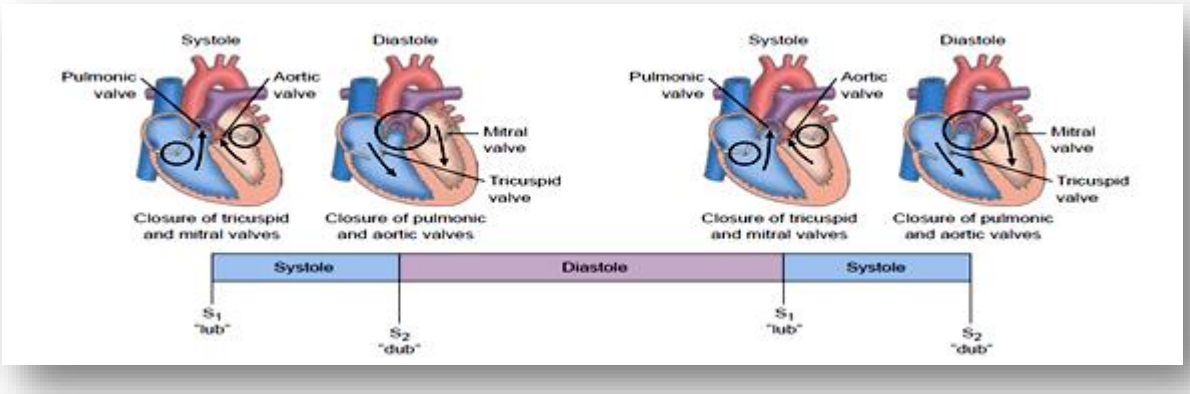


Figure I.3 First cardiac noise

I.3.2 Second hreart sound :

The second heart sound "**Dub**" is produced by the closure of the aortic and pulmonic valves. It is characterized by its acoustic content of acute frequencies. He is drier and more slamming than noise S1, and of higher tone and can reach frequencies up to 200 Hz. It corresponds to the sigmoid valve closuresaortic and pulmonary. This second cardiac noise marks the beginning of the ventricular diastole,its duration does not generally exceed 100 ms.Noise S2 is essentially composed of two groups of vibrations:

I.1 low amplitude vibrations , not audible due to blood vorticespreceding the closing of the sigmoid.

I.2 audible vibrations of high frequencies due to sigmoid closures aortic and pulmonary, noted respectively A2 and P2. Each of the A2 componentsand P2 lasts less than 50 ms, and are generally separated by a time interval of 3 to 4 ms which increases during inspiration.

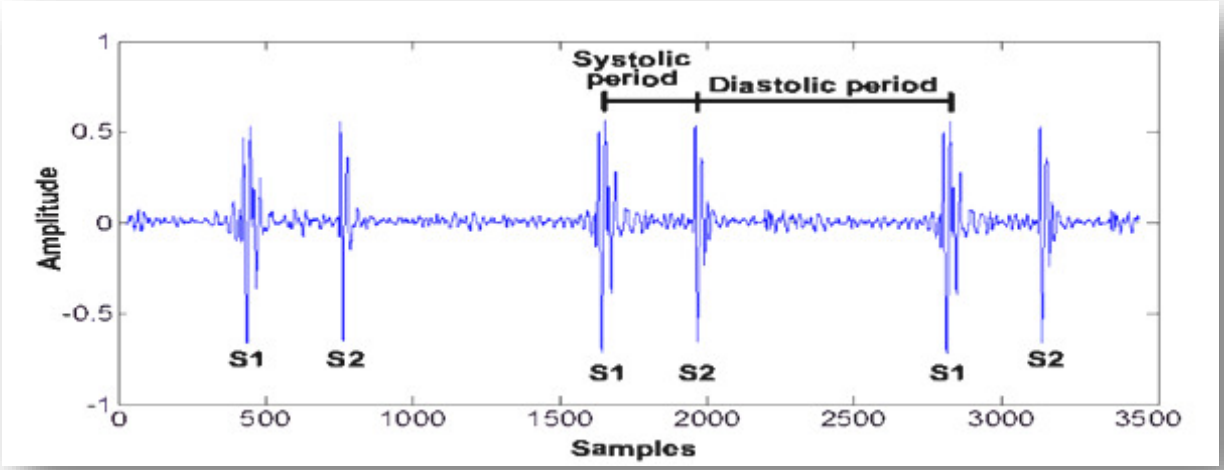


Figure I.4 Waveform of S1 heart sound lubb and S2 heart sound dupp

I.4 Abnormal heart sounds:

I.4.1 Third heart sound :

The third heart sound"Kentucky" (S1=Ken; S2=tuck; S3=y), also known as the "ventricular gallop", occurs just after S2 when the mitral valve opens, allowing passive filling of the left ventricle. The S3 sound is actually produced by the large amount of blood striking a very compliant LV.It is usually of low frequency .It is normally audible in children and young adults**[3]**.

It occurs 0.12-0.16 seconds after S2 in the early diastole, and has a frequency of 25-50 Hz. In young humans, an S3 can occur in a healthy heart: because the ventricle is not fully grown yet and thus relativelysmall, it is filling too rapidly. Over time, it will get a larger diameter and thus filling capacity. The sound iscaused by a sudden halt to the ventricular filling and it is thought to depend on the ventricular wall complianceand thickness.In a healthy heart, the amplitude of S3 is lower than S2, but an S3 greater than S2 canoccur. This was pointed out in a 2011 study concerning athletes with cardiac fatique.

I.4.2 Forth hreart sound :

The fourth heart sound "Tennessee" with the S4 being the "ten," the S1 being the "nes," and the S2 being the "see" **[3]**. Also knows as the "atrial gallop"; occurs just before S1 when the atria contract to force blood into the LV. If the LV is non-compliant and atrial contraction forces blood through the atrioventricular valves, a S4 is produced by the blood striking the LV. An S4 heart sound is often a sign of diastolic HF, and it is rarely a normal finding (unlike a S3).

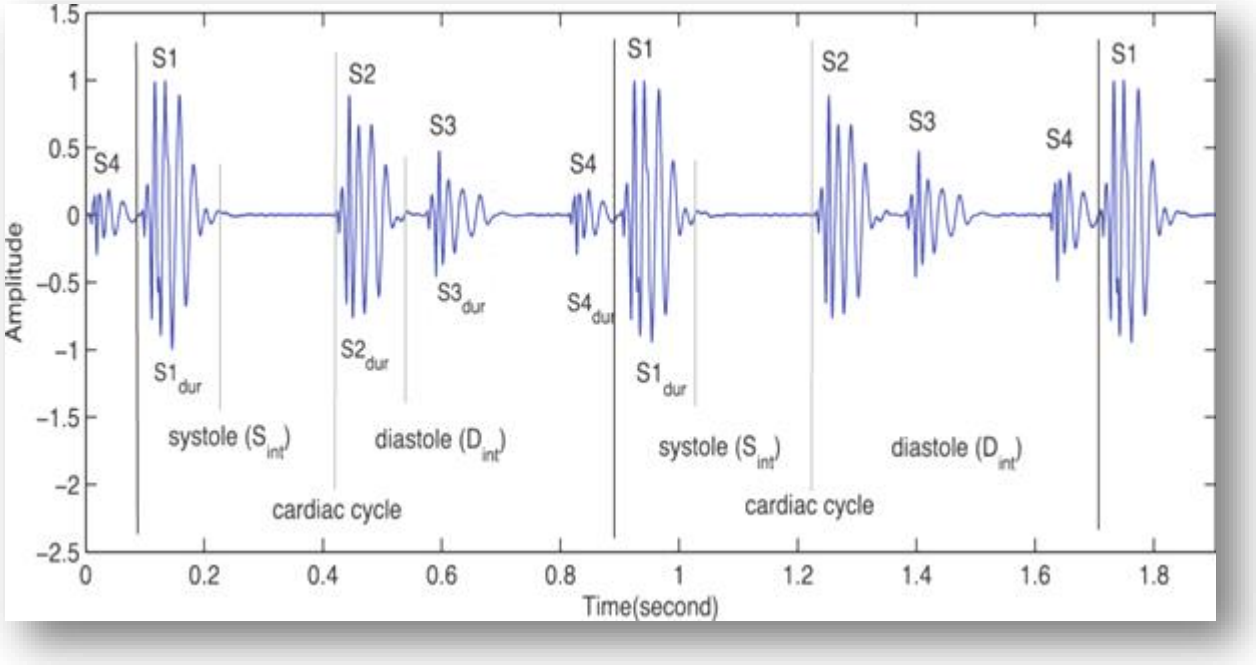


Figure I.5 PCG signal includingheart sounds (S1, S2, S3, S4).

I.4.3 Murmurs:

Heart murmurs are produced as a result of turbulent flow of blood, turbulence sufficient to produce audible noise. They are usually heard as a whooshing sound. The term murmur only refers to a sound believed to originate within blood flow through or near the heart; rapid blood velocity is necessary to produce a murmur. Yet most heart problems do not produce any murmur and most valve problems also do not produce an audible murmur**[]**.

The following paragraphs overview the murmurs most commonly heard in adults who do not have major congenital heart abnormalities.

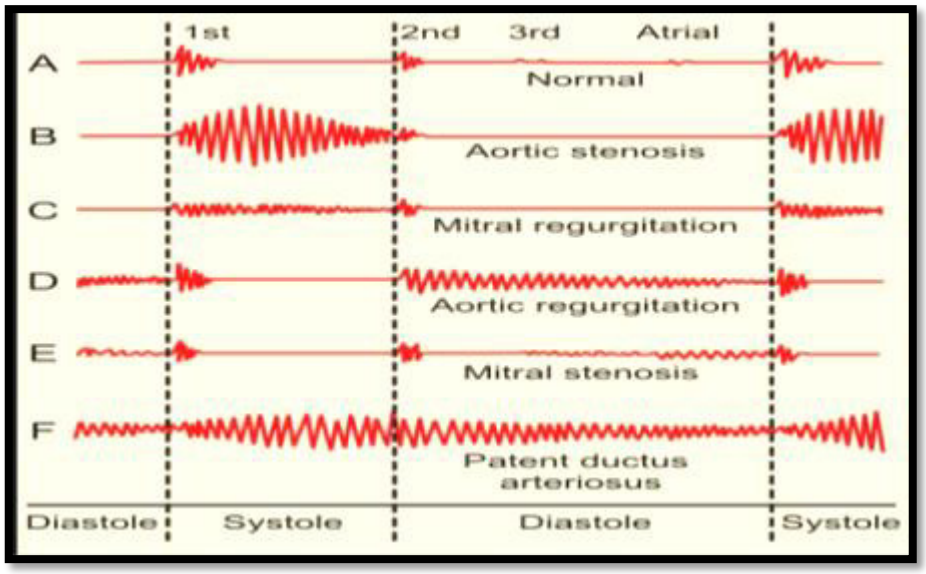


Figure I.6 Heart Sounds and Murmurs

- Regurgitation through the mitral valve is by far the most commonly heard murmur, sometimes fairly loud to a practiced ear, even though the volume of regurgitant blood flow may be quite small. Yet, though obvious using echocardiography visualization, probably about 20% of cases of mitral regurgitation does not produce an audible murmur.
- Stenosis of the aortic valve is typically the next most common heart murmur, a systolic ejection murmur. This is more common in older adults or in those individuals having a two, not a three leaflet aortic valve.
- Regurgitation through the aortic valve, if marked, is sometimes audible to a practiced ear with a high quality, especially electronically amplified, stethoscope. Generally, this is a very rarely heard murmur, even though aortic valve regurgitation is not so rare. Aortic regurgitation, though obvious using echocardiography visualization, usually does not produce an audible murmur **[]**.

- Stenosis of the mitral valve, if severe, also rarely produces an audible, low frequency soft rumbling murmur, best recognized by a practiced ear using a high quality, especially electronically amplified, stethoscope.
- Either regurgitation through, or stenosis of, the tricuspid or pulmonary valves essentially never produces audible murmurs.
- Other audible murmurs are associated with abnormal openings between the left ventricle and right heart or from the aortic or pulmonary arteries back into a lower pressure heart chamber.

Gradations of Murmurs	(Defined based on use of an acoustic, not a high-fidelity amplified electronic stethoscope
Grade	**Description**
Grade 1	Very faint, heard only after listener has "tuned in"; may be heard in all positions
Grade 2	Quiet, but heard immediately after placing the stethoscope on the chest.
Grade 3	Moderately loud.
Grade 4	Loud, with palpable thrill (i.e., a tremor or vibration felt on palpation)
Grade 5	Very loud, with thrill. May be heard when stethoscope is partly off the chest.
Grade 6	Very loud, with thrill. May be heard with stethoscope entirely off the chest.

Table I.1. Murmurs Description

ANNABA UNIVERSITY

CHAPTER 2

II.1 Problematic:

Since 1985, the heart diseases cause millions of death worldwide. Which make it the second mortality after the stroke (cerebrovascular accident). There are deferent layers of our society suffer from this illness.

According to the population statistics, the number of elderly people is increasing with time. Those are more likely to suffer from chronic diseases and to have greater and more complex healthcare needs. Most of them suffer from the transport in order to visit their cardiologist, sometimes forgetting their appointment, having a new treatment from their cardiologist who is out of country… Which lead us to the disable people who know the real and painful feeling of being tired from moving a short distance on their wheel chair, just for take a new treatment.

Also, the babies births with congenital malformations in their hearts which must have a big attention from their parents. Without forgetting that in some cases, the cardiologist can't find out that this baby having a heart problem. This has two causes: the crying of babies which noising the sound of heart beats, or the insensitivity of doctor's ear. Even for a well-trained young cardiologist to auscultate and diagnose cardiac diseases, several years' clinical experience is required. Actually, a well-trained young cardiologist could hear out the pathologic heart murmur very sensitively but it is so difficult to an inexperienced or non-clinical experience person. However, using the stethoscope to screen human's disorder needs a long-term practice and experience.

Since February 2020, corona virus which is dangerous and contagious illness shows up in Chinese Wuhan. It spreads to that point of killing thousands of people in just one country. This serious situation that all of us all over the world are living it ; forcing the healthy and the unhealthy person to stay at home in a completely or partial quarantine .Like everything is closing , also cardiologist's Clinique .

But if heart condition didn't get long-term real-time monitoring and diagnosis, it may induce sudden heart disease and delay the best time to treat, resulting in unpredictable grave consequences, even death. Now-a-days heart problems are arising day by day at a very high rate. Due to these problems, time to time heart monitoring is very essential.

II.2 Synoptic:

To solve the problematic above we propose the synoptic bellow (figure II.1) which describe all the system with specifying the heart sound node we have done the diagrame bellow (figure II.2).

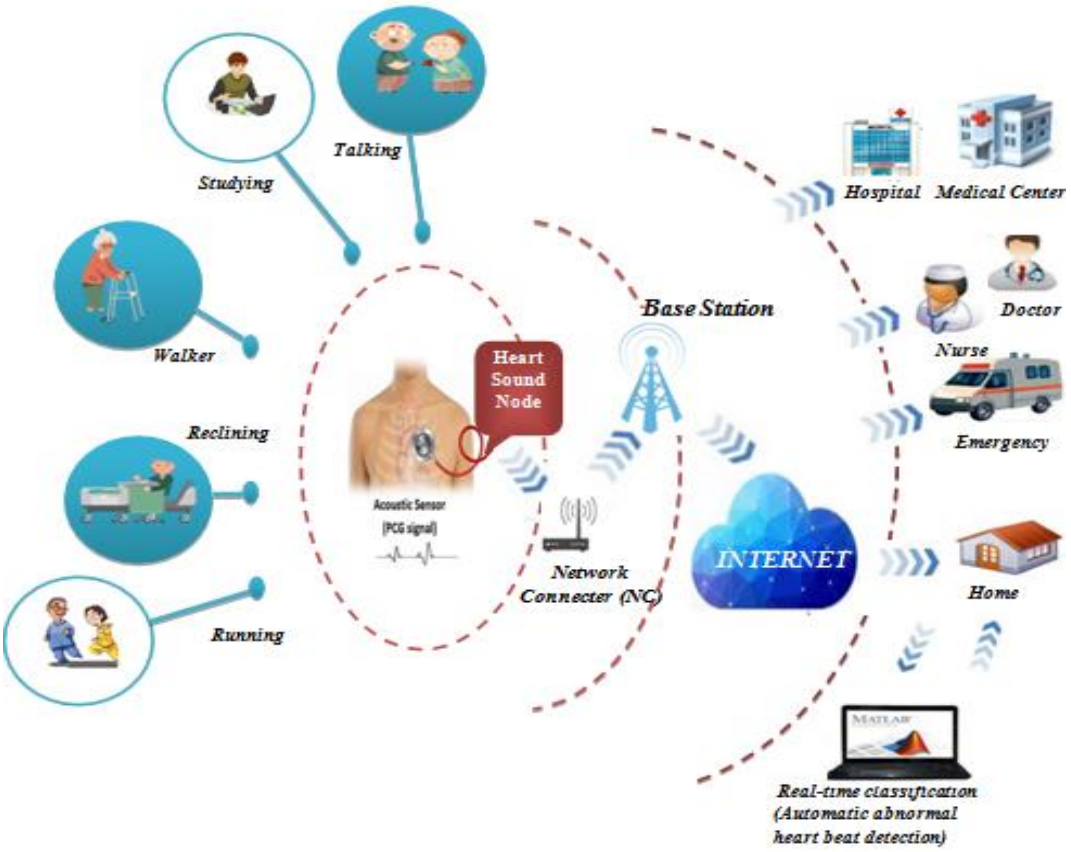


Figure II.1 Proposed synoptic

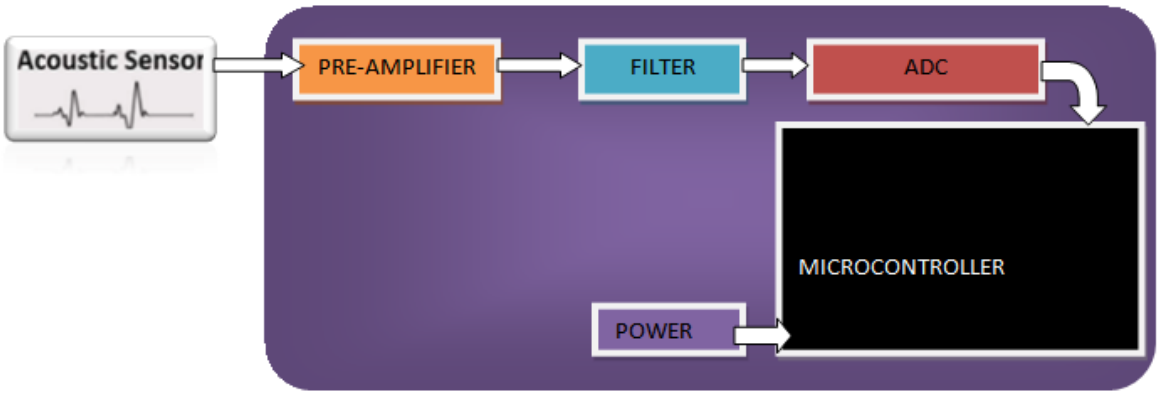


Figure II.2 Heart sound node diagram

II.3 Hardware description

II. 3.1 Senser:

The sensor is the main element of a measuring instrument which is is directly applied the physical quantity to be measured. Its role is to convert values of this quantity called measurand, in electrical signals usable by the other elements of the measurement chain. For this, the latter must be chosen with thoroughness. The sensor used in our project is a microphone. The microphone: is a sensor that converts the energy of sound (in the form of pressure and depression) in electrical energy. When recording audio, the most commonly used sensors are microphones or accelerometers (also called contact microphone or piezoelectric microphone) in most cases. For recording heart sounds, the types of shifts that can be used are:

II.3.1.1 The Contact Microphone (piezoelectric accelerometer):

This type of microphone exploits the piezoelectric property of a Crystal whichelectrically polarize when subjected to mechanical stress. The principleis to detect the variation of the vibrations of a solid (in our case, the wallchest) and not changes in air pressure. This sensor must be glued tothe rib cage near the heart to detect vibration movementsdue to heartbeatDetected vibrations deform the piezoelectric membrane changes the valueof capacity between it and another fixed metal plate, inducing a changein the capacitor voltage under constant charge conditions. [6]

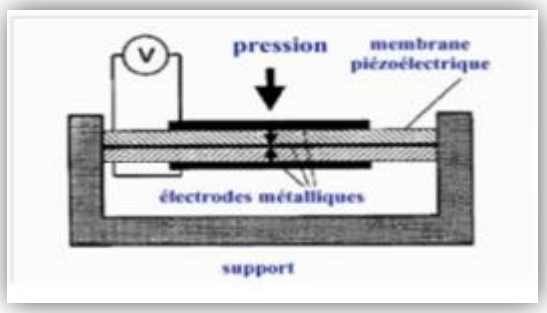


Figure II.3 Piezoelectric accelerometer

Microphones in turn fall into three most commonly used types:

II.3.1.2 Microphone Dynamique :

This type of microphone consists of a plastic or metal diaphragm, the latter is fixed to a copper coil, located in a magnetic field created by a magnet. The sound pressure waves that strike the diaphragm cause it to move, which in turn causes the coil located in the magnetic field to move. The resulting magnetic variations result in electrical variations that generally correspond to the

physical variations of the original sound wave. Dynamic microphones are extremely robust, offer a smooth and wide frequency response, do not require an external DC source for their operation. Dynamic microphones are widely used in public discourse, and in virtually all recording applications. [6]

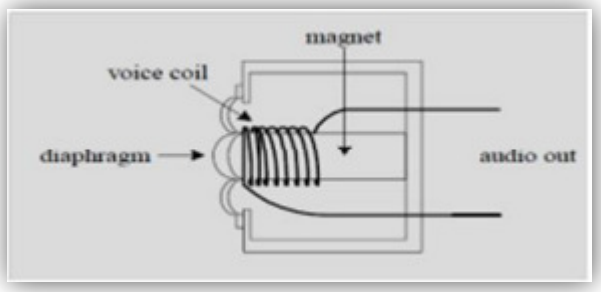


Figure II.4 Microphone Dynamique

II.3.1.3 The Condenser Microphone:

This type of microphone consists of a pair of plates which can beapproach and move away by the impact of atmospheric pressure. Indeed, theplates act as a sound-sensitive capacitor. One of the plates is inrigid metal fixed with mass. The other plate is made of metal or metallized plasticflexible which is positively charged by an external voltage source. Thecondenser microphones are characterized by low noise sound and ahigh quality recording.[6]

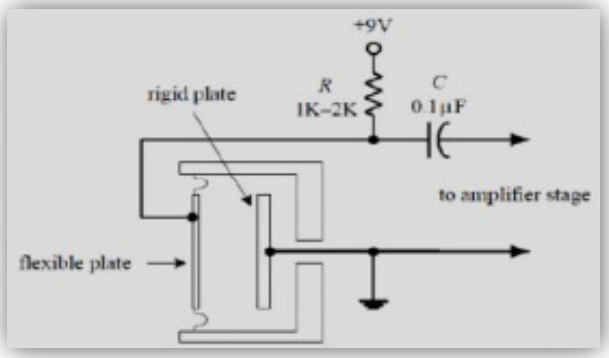


Figure II.5 The Condenser Microphone

II.3.1.4 The Electret microphone (ECM) or A microphone electret capacitor:

Is a variant of the condenser microphone. Instead of using a source ofexternal voltage to charge the diaphragm, it uses a charged plastic elementpermanently (electret) placed in parallel with a metal platedriver. Most electret microphones have a built-in FET amplifierin their little amplifier. This requires an external power supply to operate,generally a DC

voltage between +4 and +10 V. This voltage supplies themicrophone through a resistor (1-10 K) (see figure II.6). The electret microphoneresponds well to medium frequencies, however it has a poor response to themlow frequencies. For this reason, its use is limited to communicationsvocal. In addition, the performance of electret microphones has been decreasing over the years.because of the decrease in the electret charge. [6]

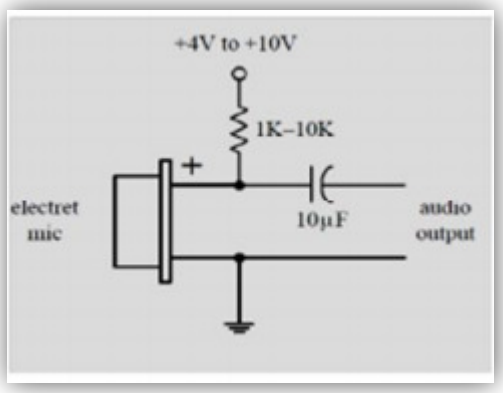


Figure II.6 Microphone electret capacitor

The following table (Table II.1) shows a comparative study of the characteristics ofthree types of microphones:

	ECM microphone	Microphone dynamic	Microphone piezoelectric	Preference
Cost	Low	Low	Important	ECM / dynamic
Sensitivity	Good	Low	Good	ECM / piezo
Dimension	Small	Big	Small	ECM / piezo
Frequency band	[20Hz-20KHz]	[50Hz-20KHz]	[10Hz-20KHz]	ECM / piezo

Table II.1 Comparative study between microphone types

Based on these characteristics presented in the previous table, we opted forusing an ECM electret microphone. This choice is also consolidated by thewide availability of this type of sensor in the component storeelectronic at the level of the Faculty of Technology of our university. [b.4]This type of Microphone has been used for the detection of heart sounds.illustrated in figure II.7 This ECM microphone has been inserted into the end of thetubing of a stethoscope to acquire heart sounds.

Figure II.7 Elements necessary for the cardiac noise sensor

II. 3.2 The amplification:

Knowing that the sound signal of the cardiac movements is very weak amplitude and cannot be detected clearly therefore, the sensor output requires appropriate amplification. The analog front end of the prototype system is mainly made up of the pre-amplifier and band pass filter. The pre-amplifier provides amplification with gain of 11 v/v. Moreover, it provides DC shifting controlled by the resistors R2 and R1 to remove the negative components of the signal (Figure 6), which is necessary for the analog-to-digital converter (ADC). The resistors R4 and R3 DC-bias are the microphone input .

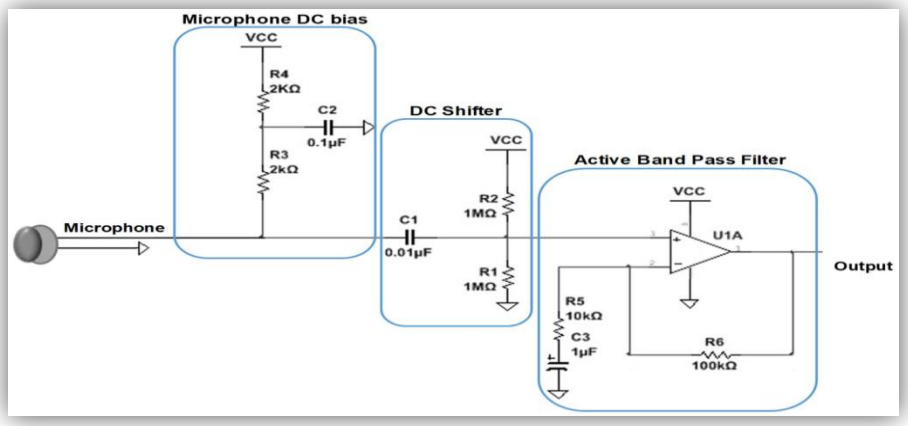


Figure II.8 Schematic of the pre-amplifier of the sensor system.

II. 3.3 The filtring

In addition, the design has a first-order band pass filter of 20–600 Hz cutoff frequencies used to provide extra filtering to the signal. The Bessel filter was selected to perform the filtering of the HS signal as it has a linear phase shift that is necessary for audio signal filtering. The design consists of a fourth order high-pass filter (HPF) followed by a fourth-order low-pass filter (LPF). The bandwidth of the filter was chosen to be between 20 Hz to 600 Hz, which is sufficient to record a clean HS signal. The filter has a total gain of 3.06 v/v. The MCP604 quad operational amplifier was used for designing the filter as it provides low-bias current, high-speed operation, and rail-to-rail output swing.

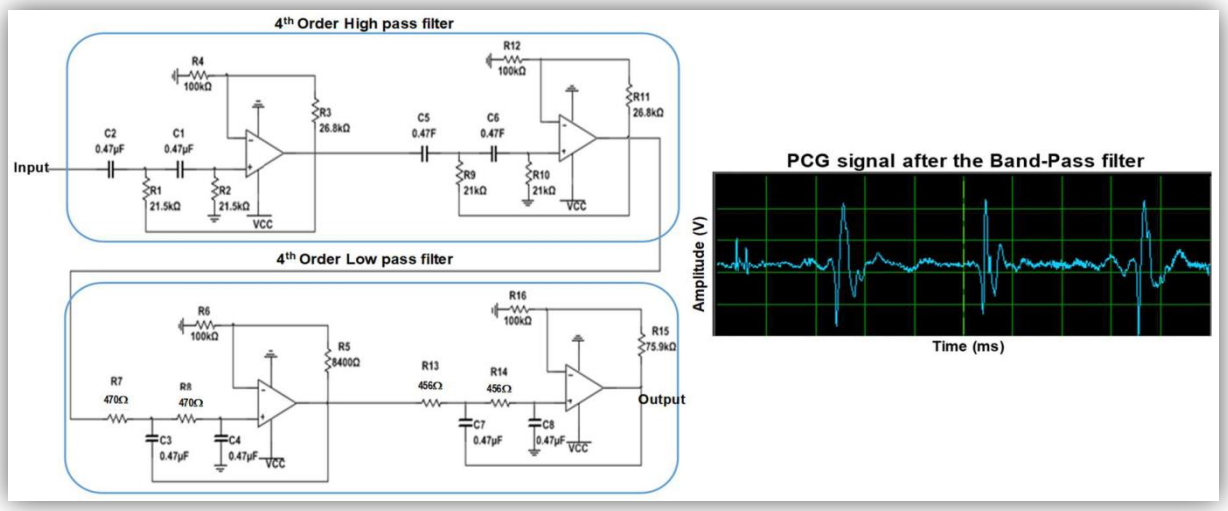


Figure II.9 Schematic of the filter of the sensor system.

II. 3.4 The MCP3008:

II.3.4.1 PIN DESCRIPTIONS

The descriptions of the pins are listed in Table II.2 . Additional descriptions of the device pins follow.

MCP3008 PDIP, SOIC	Symbol	Description
1	CH0	Analog Input
2	CH1	Analog Input
3	CH2	Analog Input
4	CH3	Analog Input
5	CH4	Analog Input
6	CH5	Analog Input
7	CH6	Analog Input
8	CH7	Analog Input
9	DGND	Digital Ground
10	$\overline{CS}$/SHDN	Chip Select/Shutdown Input
11	D_{IN}	Serial Data In
12	D_{OUT}	Serial Data Out
13	CLK	Serial Clock
14	AGND	Analog Ground
15	V_{REF}	Reference Voltage Input
16	V_{DD}	+2.7V to 5.5V Power Supply
–	NC	No Connection

Table II.2 Pin function table [7]

1. Digital Ground (DGND)

Digital ground connection to internal digital circuitry.

2. Analog Ground (AGND)

Analog ground connection to internal analog circuitry.

3. Analog inputs (CH0 - CH7)

Analog inputs for channels 0 - 7, respectively, for the multiplexed inputs. Each pair of channels can be programmed to be used as two independent channels in single-ended mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN.

4. Serial Clock (CLK)

The SPI clock pin is used to initiate a conversion and clock out each bit of the conversion as it takes place. See Section 6.2 "Maintaining Minimum Clock Speed", "Maintaining Minimum Clock Speed", for constraints on clock speed.

5. Serial Data Input (DIN)

The SPI port serial data input pin is used to load channel configuration data into the device.

6. Serial Data Output (DOUT)

The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

7. Chip Select/Shutdown (CS/SHDN)

The CS/SHDN pin is used to initiate communication with the device when pulled low. When pulled high, it will end a conversion and put the device in low-power standby. The CS/SHDN pin must be pulled high between conversions. [7]

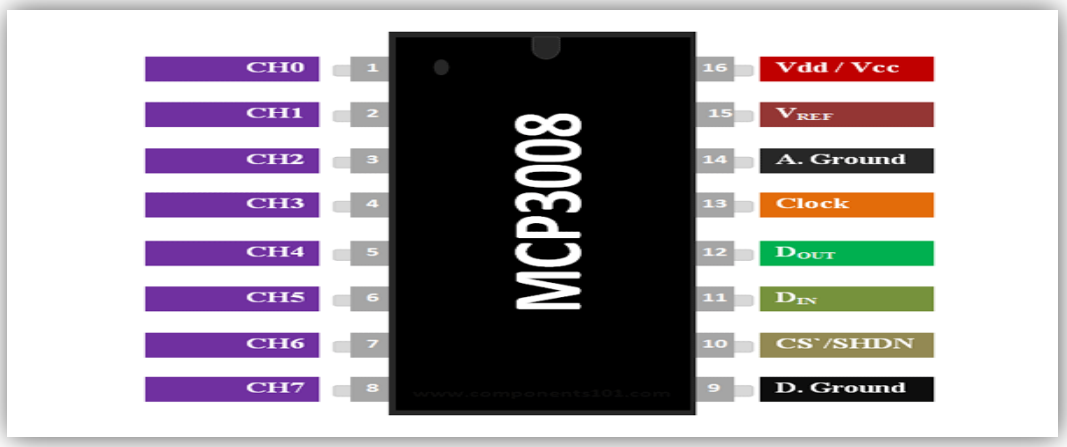


Figure II.10 MCP3008

II. 3.5 The microcontroller board

II. 3.6.1 Arduino board

it is an integrated circuit based on a programmable microcontroller which can analyze and produce electrical signals in order to perform specific tasks [5] [6]

Figure II.11 Arduino UNO board

STM32 card

It is an integrated circuit based on a microcontroller composed of an ARM processor, RAM memory and a debugging interface [5]

Figure II.12 *STM32 board*

II. 3.6.2 Raspberry Pi

It is a minicomputer that runs the Linux operating system [5]

Figure II.13 Raspberry Pi board

II. 3.6.3 Beaglebone

It's a mini electronic card that has the functionality of a computer basic. [5]

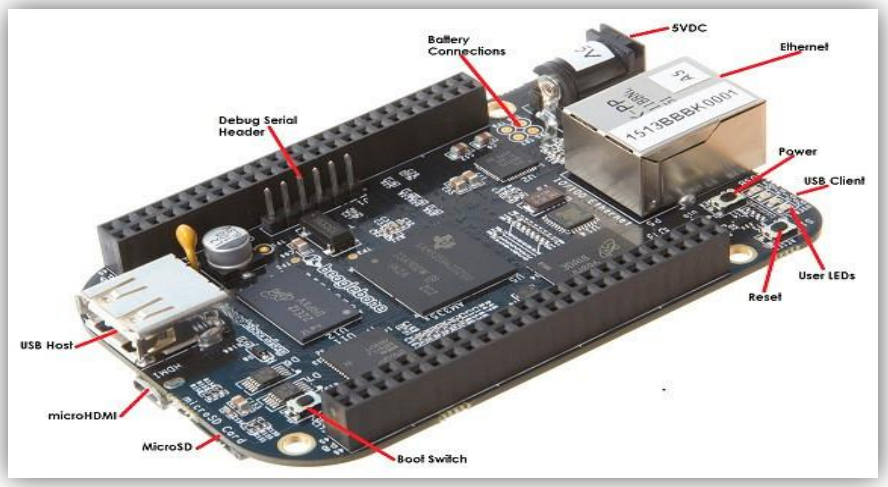


Figure II.14 Beaglebone board

The following table gives us a technical comparison between these boards: [5]

Boards' name	Arduino uno	beaglebone	Raspberry pi b+	STM 32
Born	2005	2008	2012	2007
Cost (Dollars) $	30	90	30	10
Processor	ATmeg328 8 bits	Sitara335x on a cotex	BCM2835 in An ARM11-cpu integred	cotex
Memorry flash	32 Ko	MicroSD-4G	MicroSD-32G	Max 2048 Ko
Consumption (W)	0.5	2.5	3.5	base
RAM	2 ko	256 Mo	512 Mo	Max 128 Ko
Ethernet	NO	YES	YES	NO
USB	NO	1	4	Micro-USB
Video output	NO	NO	HDMI,RCA	NO
Operating system	RTOS	LINUX	LINUX	RTOS

Table II.3 Technical comparison: Arduino Uno, Beaglebone, Raspberry Pi and STM32

Analyzing the comparative table

- We note that in terms of RAM or external memory, the capacity Raspberry is more important than other cards.

- The Raspberry has connectors (HDMI, Ethernet port, USB port ...) more than Beaglebone, STM32 and Arduino UNO.

- The Raspberry and the Beaglebone differ in their execution of the Linux system which facilitates communications with the outside world.

- The Arduino has a weak power in front of the other cards. Taking into account Table III.1, we deduce that the Raspberry represents a more efficient and better suited to our needs. The Raspberry has connectors suitable for the functioning of our system. In addition, its price performance ratio is more than affordable [Aissia & Chaieb 2015].

II. 3.6.4 Raspberry pi presentation

A Raspberry Pi is a thirty five dollar, credit card sized computer board which when plugged into an LCD and attachment of a keyboard and a mouse, it is able to complete the functions of any regular PC can. Like a PC, it has RAM, Hard Drive (SD Card), Audio and Video ports, USB port, HDMI port, and Ethernet port. With the Pi, users can create spread sheets, word-processing, browse the internet, play high definition video and much more. It was designed to be a cost friendly computer for users who needed one. There are three models, Model zero, A and B. Model B 3 is the faster containing 1GB of RAM as well as the ability to over clock. The **Raspberry Pi 3 Model B+** is the latest product in the Raspberry Pi 3 range, boasting an updated 64-bit quad core processor running at **1.4GHz with built-in metal heat sink**, dual-band 2.4GHz and **5GHz wireless LAN**, **faster (300 mbps) Ethernet**, and **PoE capability via a separate PoE HAT**. [1]

Figure II.15.A. Raspberry pi 3 B+

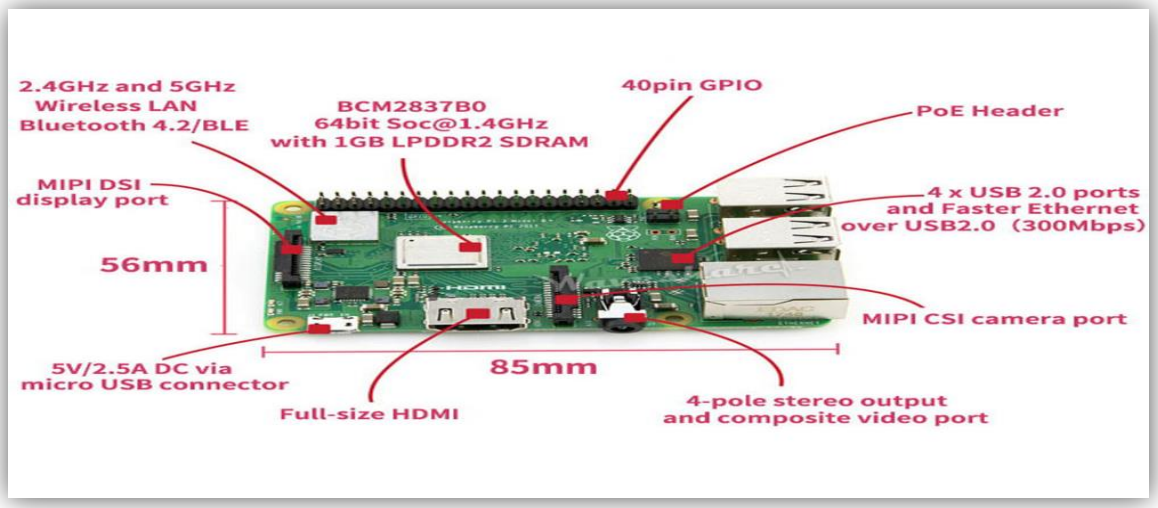


Figure II.15.B. Raspberry pi 3 B+ (with explication)

II. 3.6.5 The components of Raspberry Pi

The following figure represents a standard Raspberry:

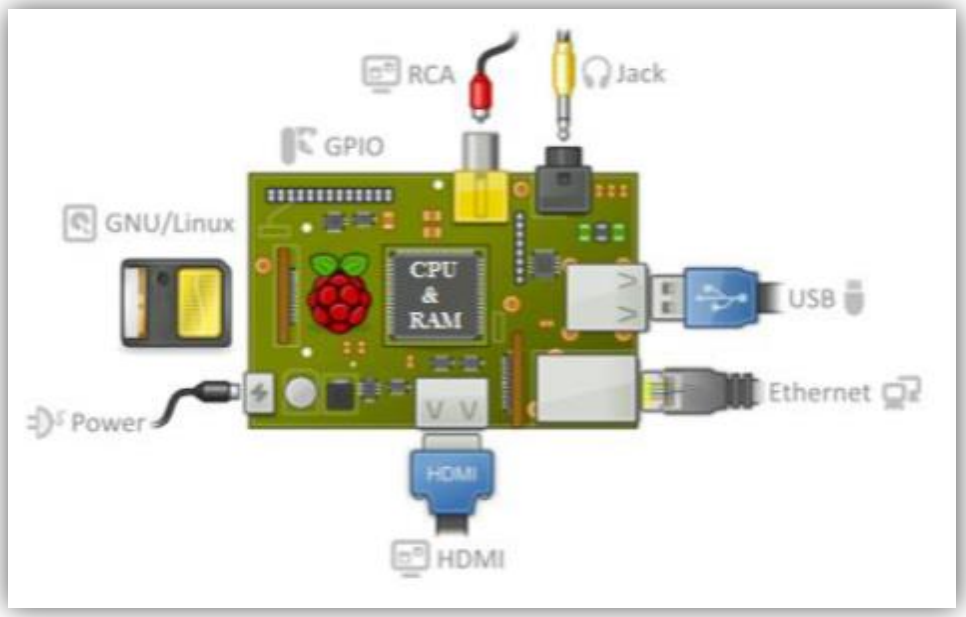


Figure II.16 the standard components of a Raspberry pi

A. ARM processor

ARM architectures are low-power processor architectures, introduced from 1983 by "Acorn Computers" and developed since 1990 by "ARM Ltd". Raspberry Pi B + has an integrated Quad-core ARM Cortex-A7 900 MHz processor (Broadcom BCM2836). [5]

B. RAM memory

It is the memory in which the Raspberry places the data during its processing, it has a capacity of 512MB .

C. A variety of connectors

C.1. HDMI: "High Definition Multimedia Interface" allows the Raspberry Pi to be connected to a compatible device: LCD screen or a video projector ...

C.2. USB 2.0 port: The "Universal Serial Bus" port is a serial port which is used to connect the Raspberry to other devices, and there are 4 USB ports. The Raspberry Pi works with virtually any USB keyboard and mouse. You can also use most wireless keyboards and mice (those that work with a dongle that plugs into a USB port).

C.3. Ethernet Port: This is a port that corresponds to the international Ethernet protocol for packet-switched local area networks.

C.4. RCA socket: "Radio Corporation of America" is an electrical connector used in the audio / video field via jack.

C.5. A SD card slot: The Raspberry needs additional external memory to operate. This slot is used to connect the external memory. You can use your own SD card in the Raspberry Pi, but it will need to be prepared with a disk image of an operating system.

C.6. A jack: It is an audio-video connection its dimension is equal to 3.5 mm.

C.7. Power supply: the conventional USB power supply with a USB-A to micro-USB cable. The power supply should be able to deliver at least 700mA, but 1A gives the Raspberry Pi some headroom that will be used by devices connected to its USB ports. If you look closely at the specifications of the power supply, you should be able to determine its capacity. Sometimes the capacity is expressed in watts (W); if this is the case, it must be at least 3 W. If it indicates 5 W, this corresponds to 1 A. [5]

C.8. GPIO: The GPIO pins "General Purpose Input / Output" of the Raspberry Pi allow you to control other electronic components as well as interfaces such as LEDs, motors and relays. These various interfaces are generally grouped under the term of "outputs". As for "inputs", your Raspberry Pi can read and interpret the status of buttons, switches, temperature, light, motion or proximity sensors, etc., the list goes on. Using the asynchronous (serial port seen in the previous chapter) and synchronous (SPI and I2C)

digital interfaces, the potential applications are even more numerous. The Raspberry B + has 40 pins. [5]

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figure II.17 the GPIO port

II.4 Electronic schema:

According to the hardware choices, we implemented the following electronic schema (Figure II.16)

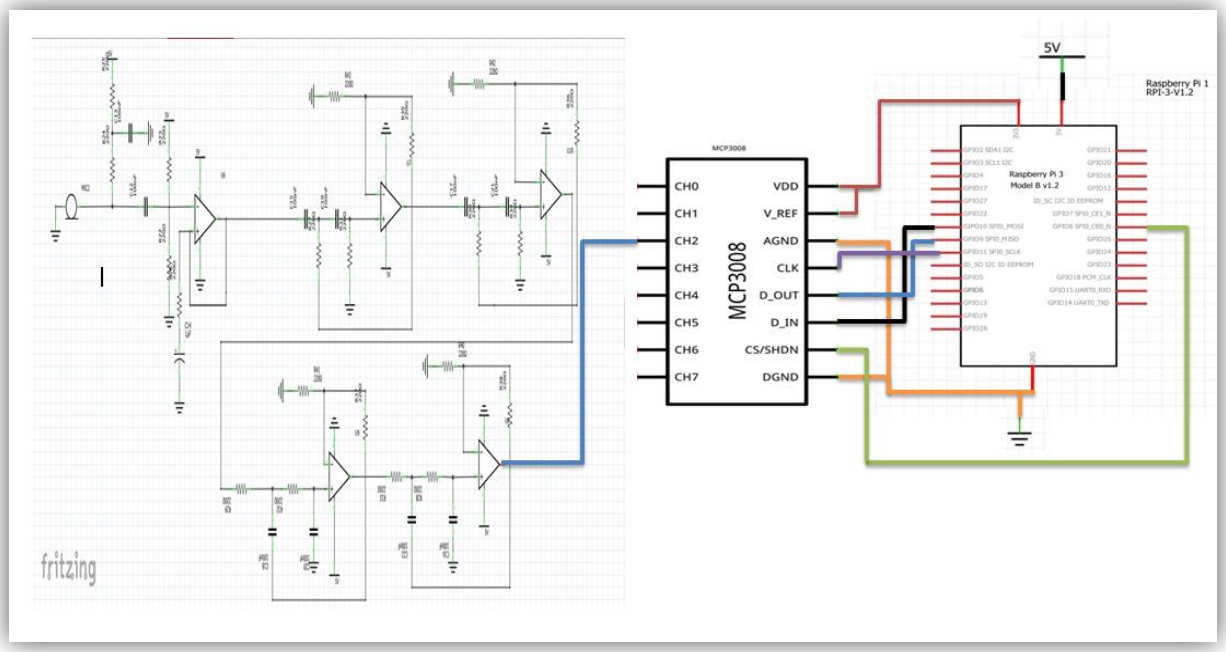


Figure II.18 Electronic schema

Than to connect the circuit above, we followed the order bellow:

- The output of filtering and amplifying circuit to CH2 MCP3008
- MCP3008 VDD to Raspberry Pi 3.3V
- MCP3008 VREF to Raspberry Pi 3.3V
- MCP3008 AGND to Raspberry Pi GND
- MCP3008 DGND to Raspberry Pi GND
- MCP3008 CLK to Raspberry Pi pin 18
- MCP3008 DOUT to Raspberry Pi pin 23
- MCP3008 DIN to Raspberry Pi pin 24
- MCP3008 CS/SHDN to Raspberry Pi pin 25

ANNABA UNIVERSITY

CHAPTER 3

III.1 Raspberry pi preparing:

III.1.1 Installation of the "Raspbian" operating system

The Raspberry Pi does not have an operating system. First, we downloaded Raspbian from the official site: raspberrypi.org/downloads. Then we installed it on the memory card using image writing software: "Win32Disk Imager".[8]

Figure III.1 installation of Raspbian with Win32Disk Imager

After installing the operating system, we executed the following commands in a terminal session to update it:

- sudo apt-get update
– sudo apt-get upgrade

System Startup

Follow these steps to start your Raspberry Pi for the first time.

1. Insert the SD card into its holder.
2. Connect a USB keyboard and mouse to the Raspberry Pi.
3. Connect the HDMI output to your TV or monitor.
4. Plug in the power to the Pi. If all goes well, you will see a list of messages scrolling on the screen.

These log messages show all of the operating system processes that launch when you boot your Pi. You will see the network interface initialize and the various devices will be detected. These messages can be examined later, entering the mesg command on the dubash command line. The very first time you start your Pi, the raspi-configse tool will automatically launch to personalize your card (figure III.20). The default configuration will need to be adjusted for your Raspberry Pi to work as expected. If necessary, you can restart this configuration tool later by entering the following command: [5]

sudo raspi-config

Figure III.2 the main menu of the raspi-config tool

III.1.2 Configuring the Pi

We will now go over the various basic and distinctive parameters which are essential to those which can be adjusted later. In the raspi-config tool, use the up and down arrow keys on the keyboard to navigate through the list. The space bar allows you to select an option and the Tab key to move from one field to another or to move the cursor at the bottom of the screen to exit [5]. The available options are:

–**expand_rootfs:** You should always choose this option, which expands Linux system files, in order to use an entire SD card. For example, if you have a 4GB SD card, Linux will be able to access an additional 2GB.

– **Overscan:** to start, leave this option disabled. If you have a high definition monitor, text may stick out of your screen. To resolve this issue, turn on the overscan option and change the values for which image fits your screen size. Enter positive values if the screen image is displayed, negative values if black margins appear on the sides.

– **keyboard:** by default, the keyboard settings are set for a UK keyboard (QWERTY). In order for the correct letters to be entered when typing on the displayed keys, you must choose the correct keyboard type. Fortunately, Linux supports a lot of languages and key layouts.

Note: your localization settings (change_locale option below) may affect keyboard settings.

–**password:** we recommend that you change the default password (rasp-berry) to make it more reliable

– **change_locale:** if you are not in England, you will need to change the value of this option to enable your language and preferences character encoding. The default setting is British (UK) English (en) with UTF-8 character encoding (resulting in en_GB.UTF-8). It is advisable to select fr_FR.UTF-8 if you are in France.

– **change_timezone:** modify this option to select the time zone corresponding to the place where you are.

– **Memory_split:** this option allows us to change the amount of memory al- rented for display. The default split is fine for now.

– **overclock:** we have the option to run the processor at speeds above 700MHz. For our first start, keep the default settings. If later you need to modify this parameter, trying the Medium or Modest values. Turbo mode allows you to go up to 1GHz with a heatsink.

– **ssh (Secure Shell Server):** this option enables access to the secure shell, which allows us to connect to our Raspberry Pi through the network. is really handy.

–**boot_behavior:** This option is selected by default and enables automatic booting of the graphical desktop environment. If we select No, the Pi will stay in Text mode. We can still launch the graphical interface manually, as follows: **[8]** raspberrypi login:

piPassword: raspberry (or the password you just set)

pi @ raspberrypi~ $ startx

Once the configuration is complete, selecting Finish (access with the Tab key). Depending on the options we have changed, the raspi-config tool may restart the system. Otherwise, we will be redirected to the command line on which you will enter:

pi @ raspberrypi~ $ sudo reboot

so that our Pi restarts and takes into account our new parameters [8] [5]

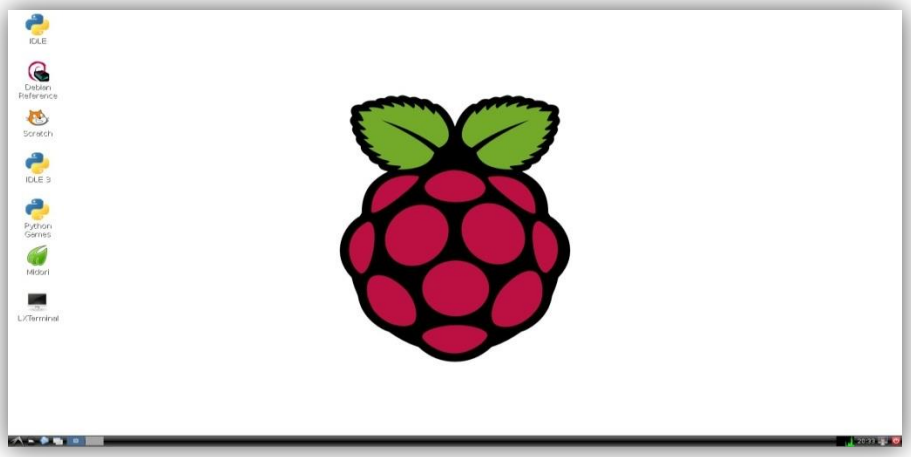


Figure III.3 Raspberry start screen

III.1.3 Turning off the Raspberry Pi

There is no power button on the Raspberry Pi, although the newer boards have a pin to accommodate a reset button. To properly shut down the operating system, go to the Logout menu in the GUI and select shutdown. You can also shut down the Pi from the command line by typing: **pi @ raspberrypi~ $ sudo shutdown –h nowousudo Ralt**

Note: Be sure to shut down the operating system as shown here and do not wildly unplug your Pi. If you cut the power without shutting down the system properly, you may damage the SD card .

III.1.4 Remote connection

To be able to connect remotely to our Raspberry, you must connect it to a local network and use the SSH protocol. 1. SSH connection: The SSH server is activated by default on the Raspberry Pi. Under Linux, the client is integrated into most distributions. On the other hand, under Windows, we must install "Putty" which is an SSH and Telnet client. Then, we just have to enter the IP address of the Raspberry in Putty to be able to connect. In order to access the graphical interface, we used the tool pre-installed on the Windows PC "Remote Desktop Connection". 2. Windows Remote Desktop Connection: Before being able to connect to the Raspberry desktop, it you have to install a small program on it. To do this, typing on our Raspberry (connected to the Internet) the following command: **sudo apt-get install xrdp** Continue by pressing **"Y".** Once the installation is complete, we have the following row that should appear. [8]

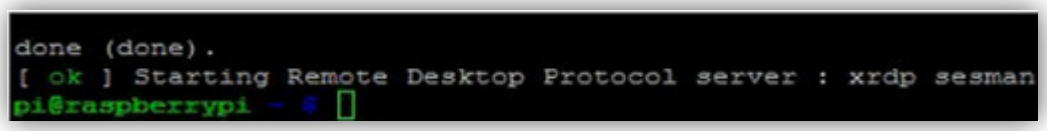


Figure III.4 xrdp installation successful **[5]**

It means the protocol is installed and started. We can now go to the PC configuration to connect to the Raspberry desktop. Now we open the Windows "Remote Desktop Connection" tool as shown in the following image: [8] [5]

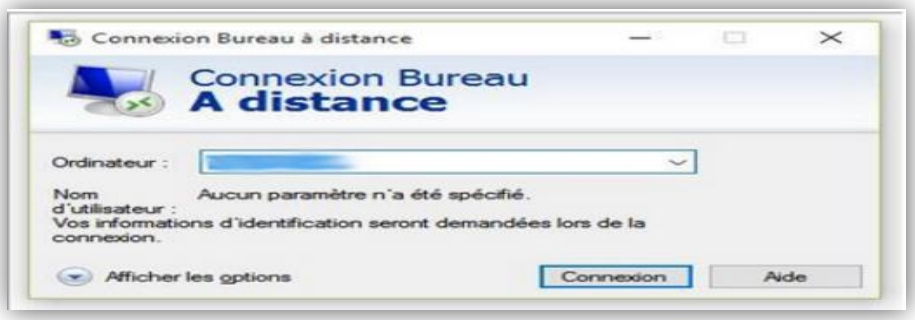


Figure III.5 Remote connection tool

We type the IP address of the Raspberry pi , we click on connection to get the following window:

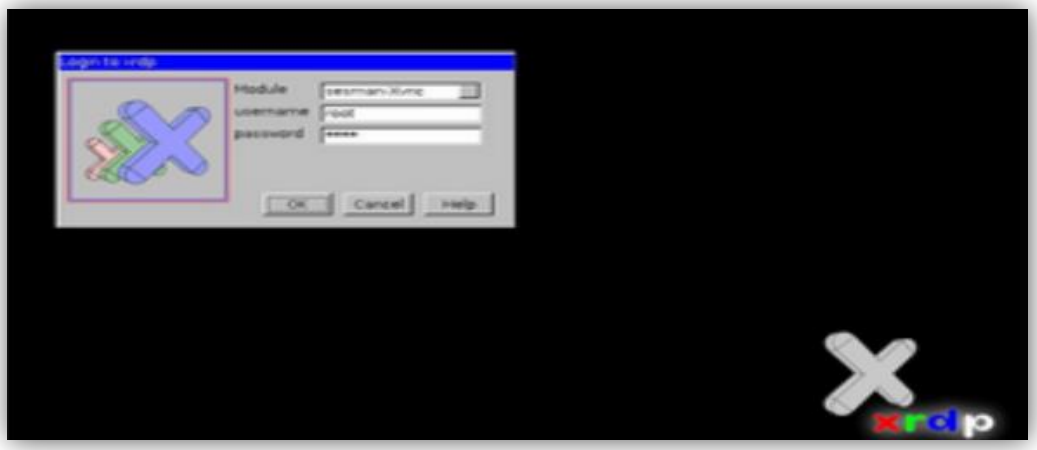


Figure III.6 Authentication window

Once logged in, we can use our Raspberry desktop on the computer [5].

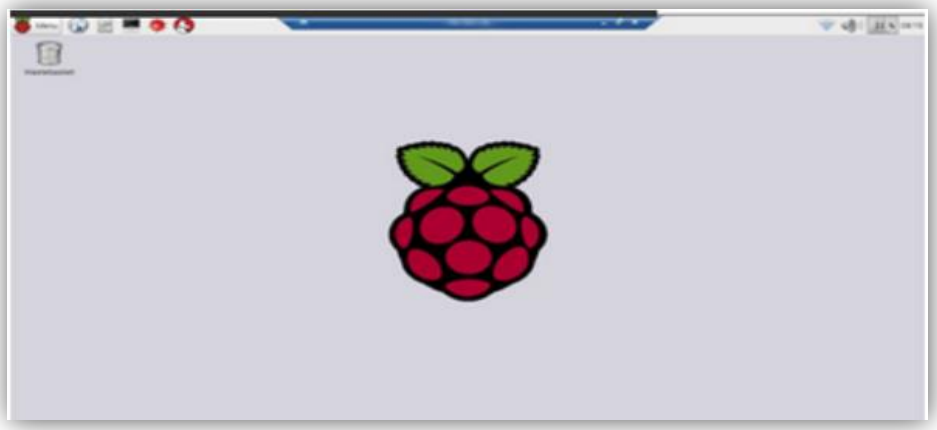


Figure III.7 Raspberry Desktop

III.1.5 IP address reservation

A DHCP server distributes an IP address to clients according to the availability of addresses in the determined range. However, it is possible to make a host always host the same assigned IP address, this is called address reservation. We must therefore indicate in the DHCP server that such IP address is reserved for such MAC address **[8]**

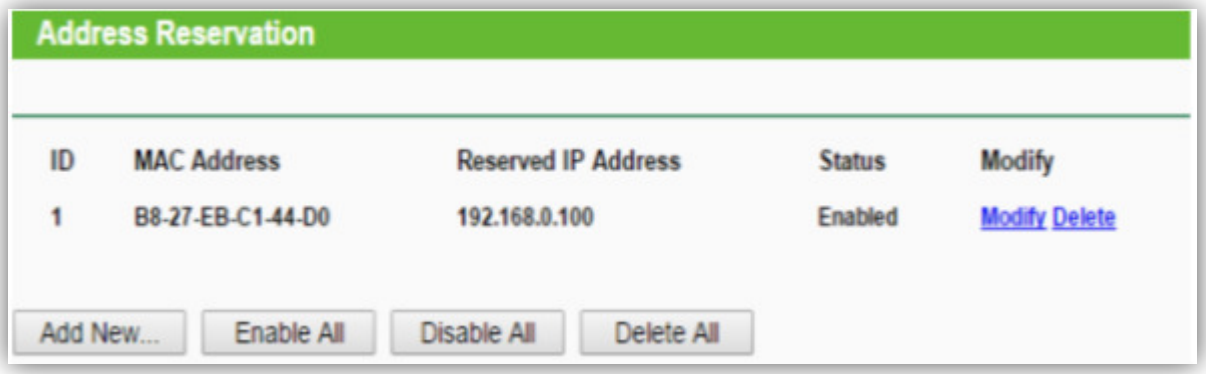


Figure III.9 Reservation of the address 192.168.0.100 to Raspberry

Or of course allocate a static IP address to our Raspberry for a wired connection via following configuration:

- We open the configuration file of the network cards:

sudo nano / etc / network / interfaces

- Then let us indicate to our network card that we want to change to static:

auto eth0
iface eth0 inet static
address192.168.1.20
netmask255.255.255.0
gateway192.168.1 .1

III.1.6 Raspberry pi programming test

Programming language

Now is the time to start creating our own programs for the Raspberry Pi. The language we are going to use is called Python. It has the great advantage of being easy to learn while being powerful enough to create interesting programs [Raspberry2017 official site].

Presentation of python

Python is a programming language, the first version of which was released in 1991. Created by Guido van Rossum, it traveled from the Macintosh of its creator, who at that time worked at Centrum voor Wiskunde in Informatica in the Netherlands, until it was associated with a particularly dedicated non-profit organization, the Python Software Foundation. , created in 2001 **[12] [8] [5] [12]**

Figure III.10 Python Language logo

Programming inputs / outputs with python

Pins marked GPIO can be used as input / output pins. In other words, any pin can be programmed as an input or an output. In this sense we will use several programming languages capable of controlling these pins such as C, Java, Bash ... but in our project we have opted for python to control these pins. The GPIO module is installed by default on the most common versions. Recent Raspbian Linux distribution.] But for older versions, you probably need to install and update. For that we execute the following command:

"**sudo apt-get install python-rpi.gpio**"

Then we run the following command for the update:

"**sudo apt-get update**"

Note: Before using the pins, you must tell the GPIO module how your code will access them. The Raspberry Pi allows two numbering: that of the screen printing of the card connector (GPIO.BOARD), or the electronic numbering of the chip (GPIO.BCM). It's up to us to choose the one we want. Here is now a small programmable led.py to control an LED:

```
import RPi.GPIO as GPIO
importtime
GPIO.setmode(GPIO.BCM)
GPIO.setup(18,GPIO.OUT)#broche18comme sortie
print"LED on"
GPIO.output(18,GPIO.HIGH)   #led allumee
time.sleep(1)#attendre pour une seconde
print"LED off"
GPIO.output(18,GPIO.LOW)#led etteinte
```

Pour tester le programme on exécute la commande suivante sur la console :

"**sudo python led.py**"

III.2 The Raspberry Pi as a web server, why?

To be able to access our Raspberry remotely via Wi-Fi and control the GPIOs/USB, we need to make our card a web server capable of hosting one or more Internet sites and therefore ensure communication with a client and respond to its requests thanks to the network protocol http. Most often, a Web server uses other software which works in collaboration with the HTTP server (Apache, IIS, Lighthttp), such as the database server (MySQL, Oracle), and to communicate with a web server, we need a server programming language like Node.js. In our project, we will opt to install the combination (Apache, Node.js, MySQL)

III.2.1 Node.js Web Server

Node.js definition

Node.js is a server-side platform wrapped around the JavaScript language for building scalable, event-driven applications. [9]

Figure III.11 Node.js logo

Installing Node.js

To download and install newest version of Node.js, use the following command:

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
```

Now install it by running:

```
sudo apt-get install -y nodejs
```

Check that the installation was successful with:

- the Node.js version with: **node –v**
- the npm version with : **npm –v**

III.2.2 Installation of Apache server

Apache definition

First, we are going to install Apache, which is the web server as such. When we talk about a web server, we often think of the machine, but this term also refers to the software that allows the machine to analyze requests from a user (in the form of http), and return the file corresponding to the request (or an error if the file is not found, or the request is badly formulated). In the context of Apache, it is therefore software that we are talking about. **[10]**

Figure III.12 APACHE logo

Installing Apache:

Before installing the server, make sure we have a fully up-to-date machine. To do this, we run the following commands:

sudo aptitude update then **sudo aptitude upgrade**

Once the Raspberry Pi is updated, we will install the Apache server with the following command:

sudo aptitude install apache2

Once the installation is complete, we can test that Apache works correctly by going to the address of the Raspberry Pi. To do this, we must try to access the Raspberry from port 80 (this port is not yet open from the outside, it will have to be done from the Raspberry itself). All we have to do is open

the Raspberry web browser (by default Midori on Raspbian), and go to the address "http: //127.0.0.1". We then get a page with a message of the type "It works!" and lots of other text.

You can now use your Raspberry to build a site in pure HTML, CSS, and JavaScript internally. **[10]**

III.2.3 A MySQL database for our server

What is a DBMS? Why MySQL?

Now that we've implemented NodeJs, we are going to set up a DBMS (Database Management System), namely MySQL. MySQL is a free, powerful, massively used DBMS (around 56% of free DBMS market share). **[11]**

Figure III.13 MySQL logo

Installing MySQL

To do this, we are going to install mysql-server and by the command:

sudo aptitude install mysql-server

When installing mysql-server, we are asked a password for the MySQL administrator account (root), be careful to remember it, because it will be used later. To check the functioning of MySQL, this time we will only use the command line. For that, we will simply connect via the command:

mysql –user = root –password = our password

We should then have a message like "Welcome to MySQL! ".

III.2.4 Easily manage databases with PHPMyAdmin

However, we might want a slightly simpler interface to administer our databases than a simple MySQL console. For that, we can install PHPMyAdmin.

PHPMyAdmin, what is it?

PHPMyAdmin is an application developed in PHP, and which aims to provide a simplified interface for MySQL. [13]

Installing PHPMyAdmin

Installing PHPMyAdmin is not at all compulsory. We will do an installation here without any particular security parameters. The installation of PHPMyAdmin is done very simply, via the package manager, using the following command:

sudo aptitude install phpmyadmin

PHPMyAdmin will ask us several questions concerning its settings, so choose it an Apache server, and then use the default choice for the base **(dbconfig-common).** For the root password, this is the one you used for MySQL. We will also need to modify the Apache configuration file to include our PHPMyAdmin installation. We open it with the nano program entering the following command:

sudo nano /etc/apache2/apache2.conf

Now at the bottom of this file we enter the following line:

Include /etc/phpmyadmin/apache.conf

When done we save and quit by pressing **CTRL + X** then **Y**. Now just restart the Apache service by entering the following command:

sudo /etc/init.d/apache2restart.

To check correct functioning of PHPMyAdmin, we will simply try to access it, using the address of our Raspberry followed by / phpmyadmin. For example, locally it will be "http: //127.0.0.1/phpmyadmin".

III.3 Configure microphone with Raspberry Pi (python)

III.3.1 *Streaming code*

This code is based on reading the microphone via a USB port. Then it will be traced like any other voice signal. This plot will be captured every 2 seconds respectively, and sent to the web page. It will be a live broadcast heartbeat signal, according to its short period. [23]

III.3.2 Recording code

This code is based also on reading the microphone via GPIO port, but it will record our sound and save it as a "file.wav" (.mp3). So our patient can send it or read it in real time. [23]

III.4 Web application design

Our site is composed of 5 folders and 4 other files as shown in the figure below.

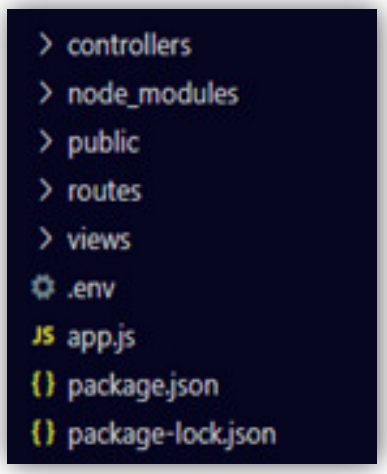


Figure III.14 Project files
And folder

III.4.1 The node_modules folder:

Before doing any other think; all modules that we need must been installed with this command:

sudo npm i express mysql dotenv jsonnwebtoken bcryptjs nodemon

III.4.2 Application main script (app.js)

This script is the basic of our web application, because it execute all the pages in the port and ip address chosen, connect it with database, define the routes.

III.4.3 Configuration script file (.env)

It's actually a simple configuration text file that is used to define some variables we want to pass into our application's environment. In our work we use it to keep the information of database secure.

III.4.4 Views folder:

The views folder contains the html pages which are saved as hbs pages ,

III.4.5 Controllers folder:

This folder contains one script "auth.js" which controls:

- The information of new user in the registering page. Which mean when this user enters an existed email address or the password and the confirmed password are not the same; it will render the same register.hbs page with an error.
- The existed users in the users' table in database. So when someone tries to hack your account it will be protected, because the error message doesn't specify which one of them (email or password) is wrong.

III.4.6 routes folder

This folder contains two scripts:

- "auth.js": this script post the information in the login and register pages to the "controllers/auth.js " , in order to verify them .

- "pages.js": This script organizes the project very well. It renders the pages which are in hbs and we have 5 ones. It works with the method of request/response. These routes are between the server and the client.

III.4.7 Public folder

This folder contains all concerning the design of our pages, which mean the CSS file and some pictures used in the web application.

III.4.8 Package.json and package-lock.json

- The "package.json" is used for more than dependencies like defining project properties, description, author & license information, scripts, etc.
- The "package-lock.json" is used to lock dependencies to a specific version number.

III.5 Performance Evaluation of Machine Learning Abnormality Detection Algorithms

Performance Evaluation of Machine Learning Abnormality Detection Algorithms Normal and abnormal heart sound data from a public PhysioNet-2016 challenge database we reused for training and testing of the machine learning algorithms in the Matlab environment to identify the best-performing algorithm and optimize the parameters of the best-performing algorithms to obtain the highest accuracy.

III.5.1 Database Description

PhysioNet challenge 2016 dataset consists of five databases (A through E) containing a total of 3126 heart sound recordings, lasting from 5 s to just over 120 s. These HS data were recorded from clinical and nonclinical environment from both healthy and pathological patients (e.g., children and adults) from four different locations—aortic, pulmonic, tricuspid, and mitral areas. In both training and test sets, heart sound recordings were divided into two types: Normal and abnormal heart sound recordings. Both the training and test sets are unbalanced, i.e., the number of normal recordings does not equal that of abnormal recordings. The number of normal recordings is higher than abnormal recordings. The recordings last from several seconds to up to more than 100 s. All recordings have been resampled to 2000 Hz and have been provided as.wav format. Each recording contains only one PCG lead.

III.5.2 Optimized Classification Model Selection

The classification model selection is shown with work flow diagram of the signal pre-processing and application of machine learning algorithm.

The HS data from the database were segmented to training and testing datasets. Signal pre-processing and automatic segmentation were accomplished using the signal processing toolbox, and training and classification of HS were accomplished by the statistics and machine learning toolbox in the Matlab 2018a.

The pre-processing steps of the HS are summarized in the analysis section. Several time (t)-domain, frequency (f)-domain, and Mel frequency cepstral coefficients (MFCC) features were extracted from the segmented HS data. The training dataset underwent pre-processing steps before it was fed into the machine learning algorithms for training. The details of the pre-processing steps and detection algorithm will be discussed in the analysis section.

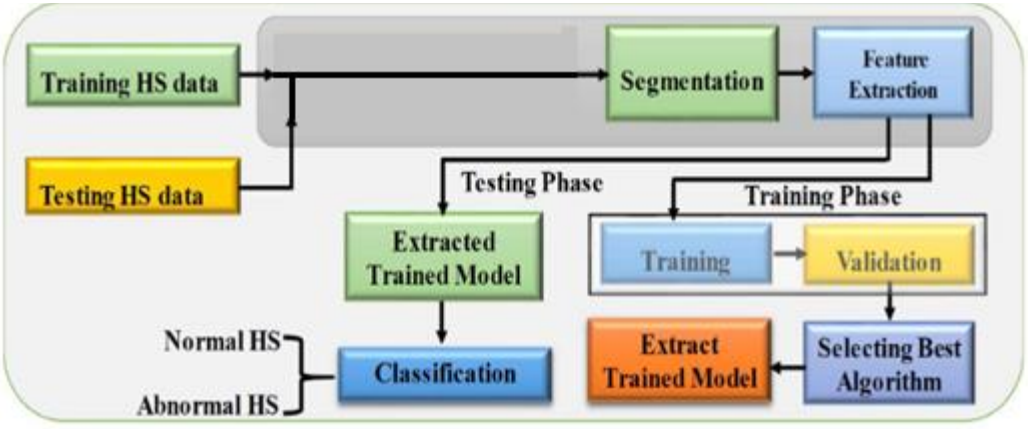
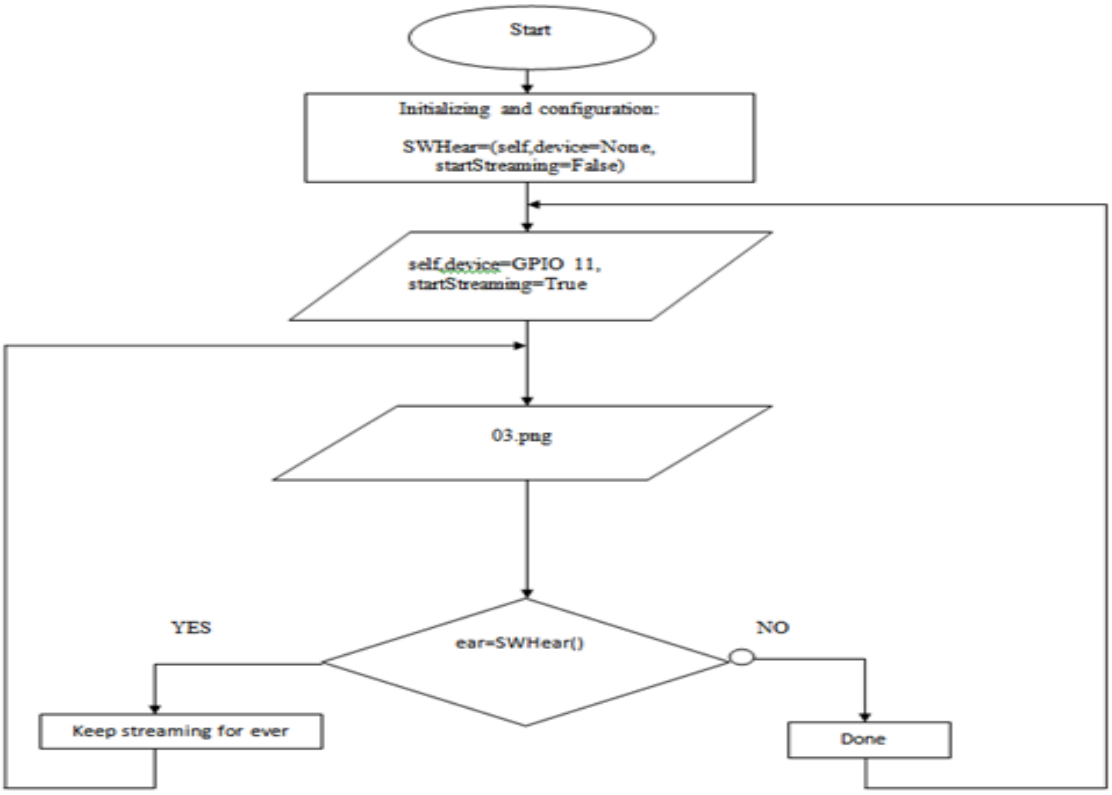


Figure III.15 Blocks of the machine learning-based abnormality detection algorithm.

Twenty-two different ML algorithms (three decision tree, two discriminant analysis, six support vector machines (SVM), six k-nearest neighbor (KNN), and five ensembles classifiers) were trained with five-fold cross-validation using 27 features of the training dataset and the best-performing

algorithm was identified. Feature reduction and hyper-parameter optimization was used to optimize the best-performing algorithm. Their validation accuracy, sensitivity, and specificity along with other performance metrics were evaluated. The trained model was used for calculating the performance evaluation matrix for the testing data in identifying the normal and abnormal HS.

III.6 Hardware Organization chart (first python code)



CHAPTER 4

VI.1 Web page results

VI.1.1 Home page (index.hbs)

This page present the first page that any client can see it when he/she enter to the ip address with the coerent port .

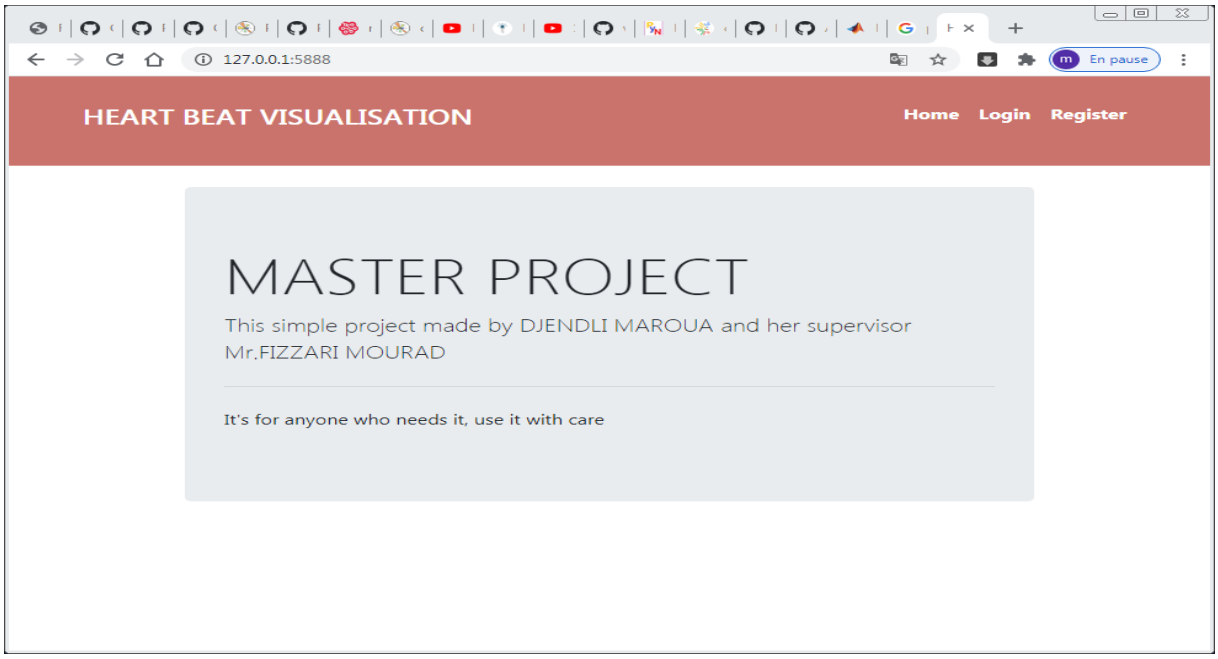


Figure VI.1 home page

VI.1.2 Registering page (register.hbs)

This page present the second page that any client can see it when he/she press the button "Regiter".Any one can creat an acount if he/she put their correct information

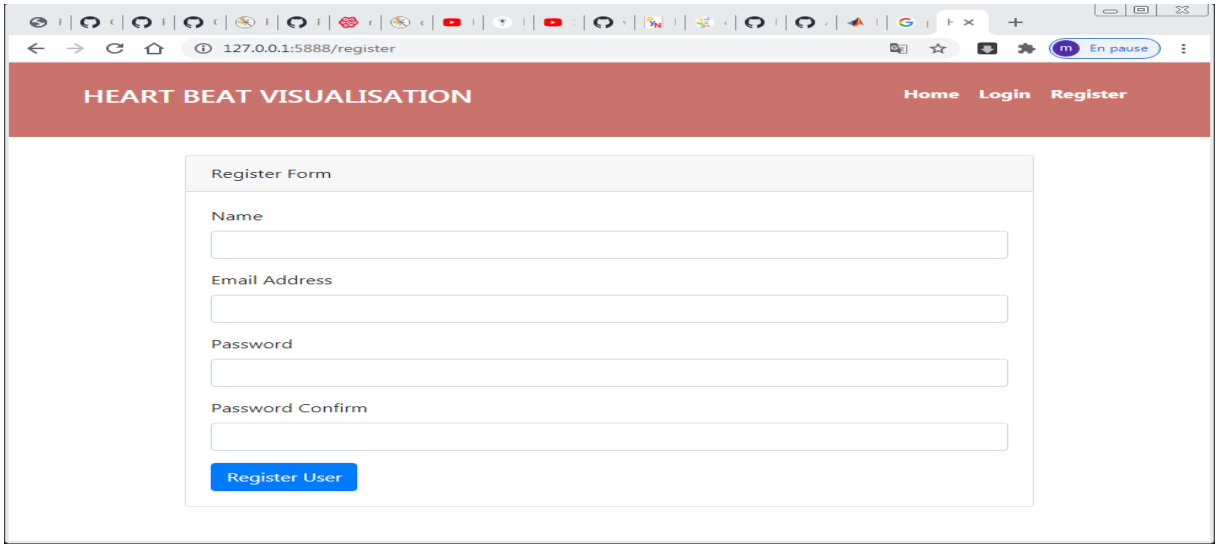


Figure VI.2 Registering page

VI.1.3 Login page (login.hbs)

After creating an account, press the button login and enter your address email and password then press login (the blue one)

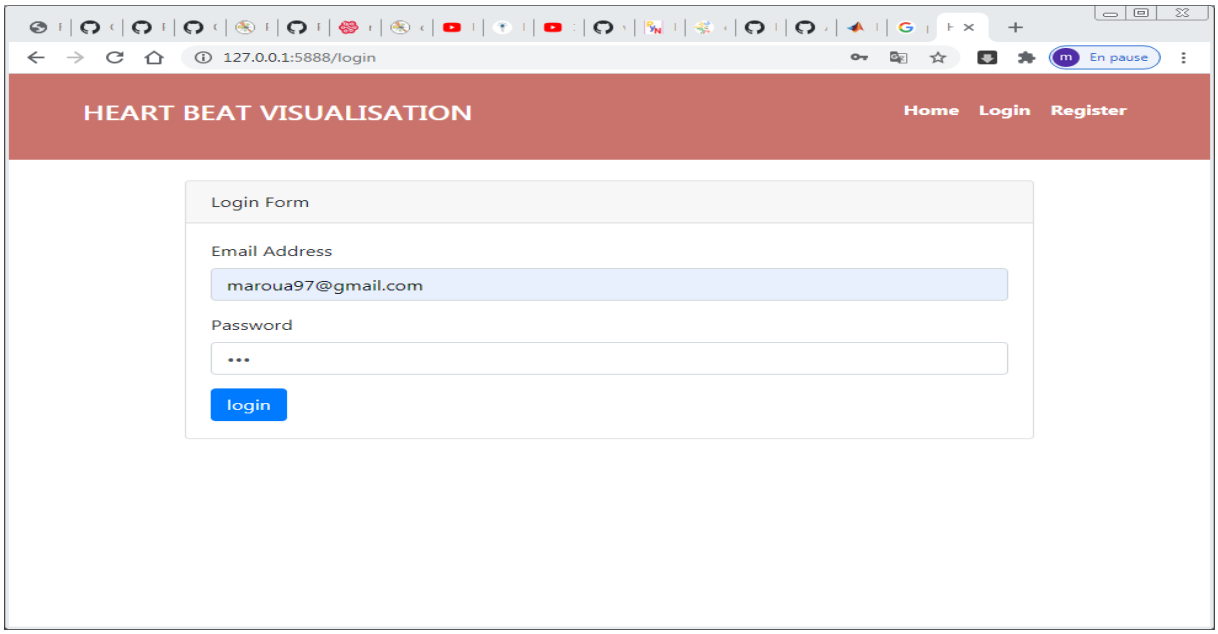


Figure VI.3 login page

VI.1.4 My profile (profile.hbs)

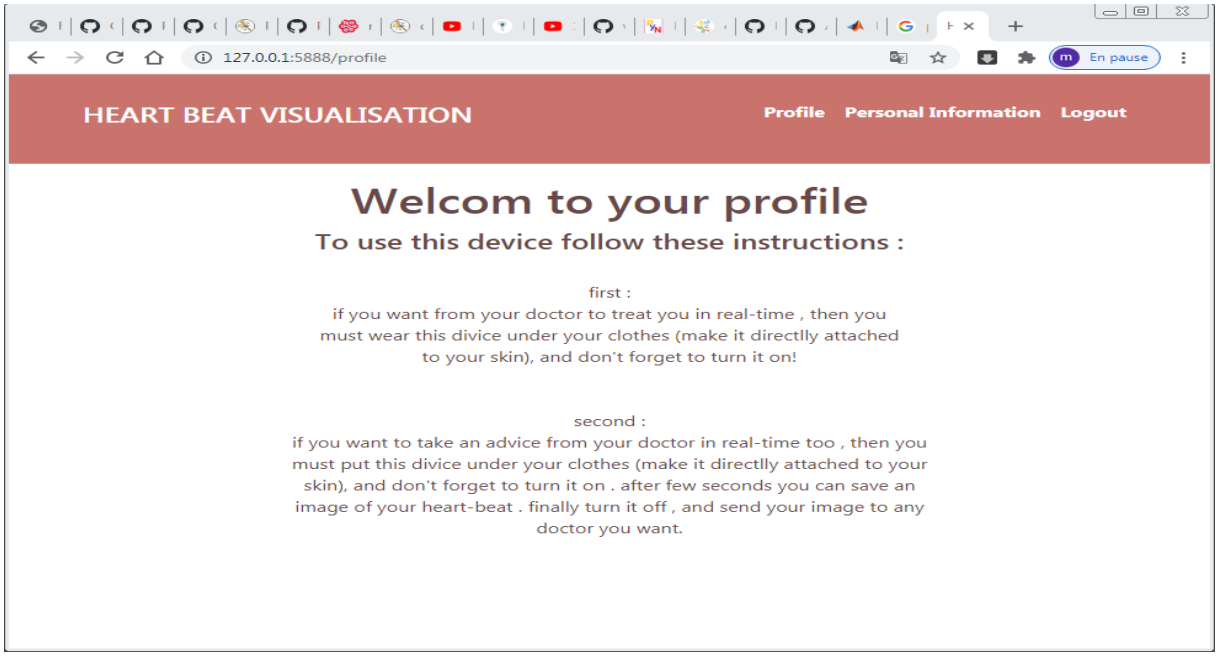


Figure VI.4 profile page

VI.1.5 Personal information (inform.hbs)

Press the button personnel information and you will see the figure contain your heart signal in real time (after running the first python code). You can save it as image directly, or as a file .wav by running the second python code. This file .wav give us the ability of classify it also in real time by using the Matlab .

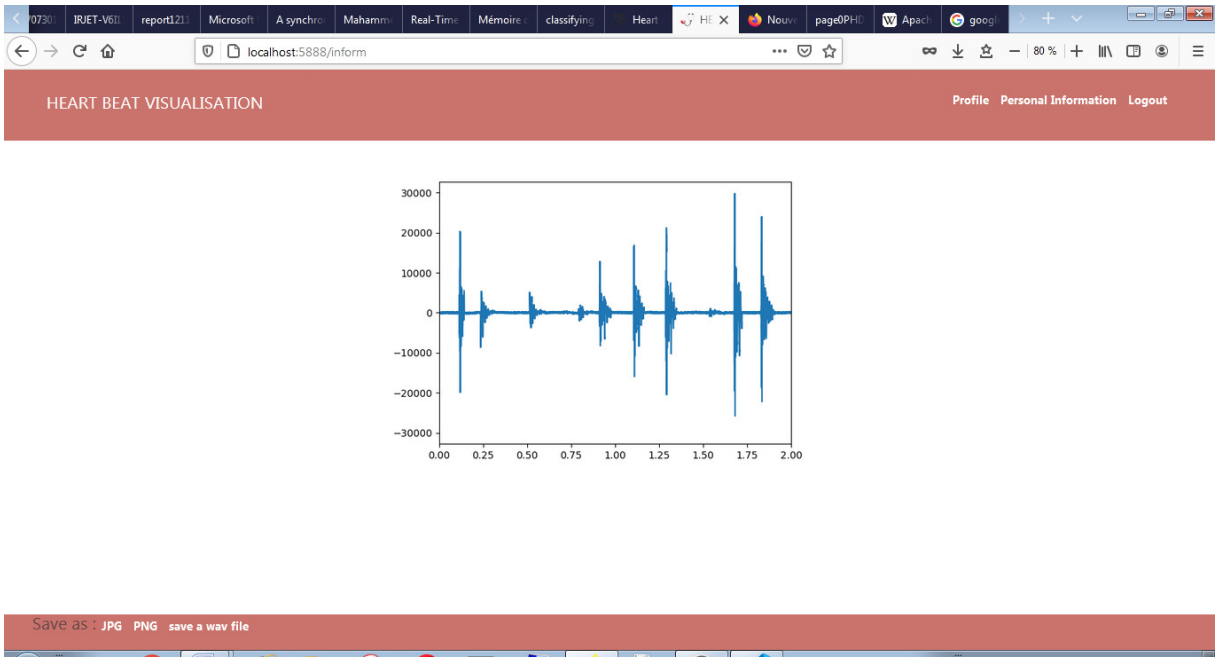


Figure VI.5 Personnel information page

VI.2 MATLAB Results:

VI.2.1 Analysis

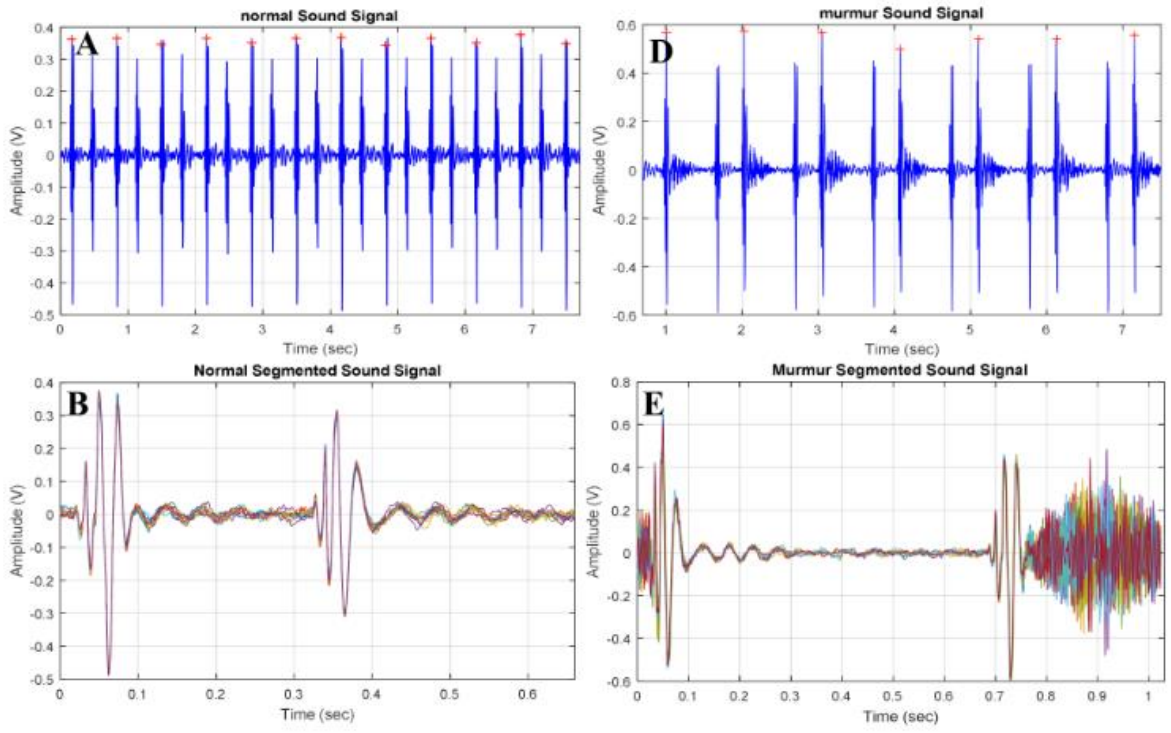
There are several pre-processing steps that were applied to the heart sound data before it can be used by the machine learning algorithm. During the training and testing period, these steps were done manually, and these were made automatic for real-time classification.

VI.2.1.1 Pre-Processing Steps

The following pre-processing steps were carried out to filter the noises and spikes, and segment the HS data:

VI.2.1.2 Segmentation

Segmentation of the PCG signals into heart cycles or marking of cycle starting instances are very important to generate the epoch of interest for training, and testing of the machine learning algorithm. There is much literature and state-of-the-art tools publicly available for segmenting the HS data. Since the location of the HS acquisition place has significant influence on the noise contamination to the PCG signal; therefore, extraction of the heart cycle period reliably is a challenging task. However, when the PCG signal was recorded with electrocardiogram (ECG) signal; this segmentation process become comparatively easier as ECG R-peaks are more distinct than the PCG signal's S1 and S2 peaks. In this work, we have automatically identified the S1 peaks, which are the most dominant peak of the PCG signal. PCG signal between one S1 peak to another S1 peak was used mainly as the heart cycle along with offset to capture the beginning of S1 signal and ending of S2 signal. Figure 11 show the normal and abnormal PCG signal of several seconds were segmented to heart cycle.



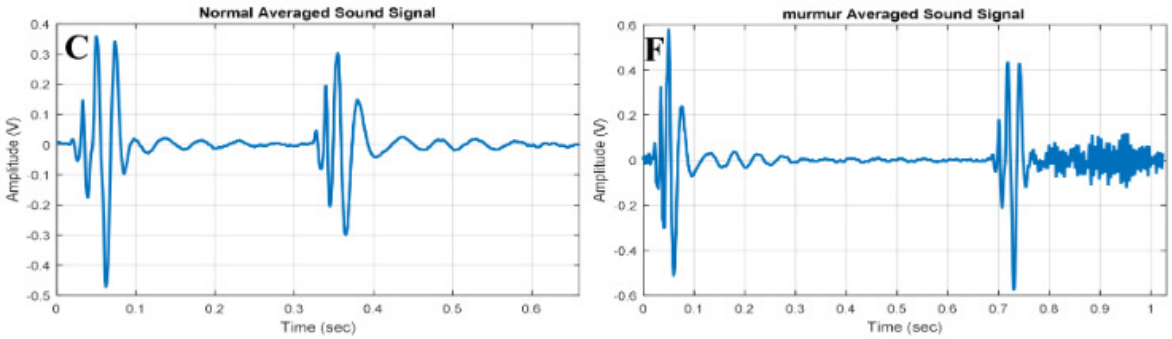


Figure VI.6 Normal and abnormal heart sounds (HS): (A,D) detection of peaks; (B,E) overlaid segments;(C,F) average of the segments.

VI.2.1.3 Feature Extraction

The power spectral of the signal in Figure 12 shows that the power spectral density peaks appear at different frequencies for normal and abnormal PCG signals. Moreover, the power spectral density at higher frequency has no peak for normal PCG signal, where as this is not the case for abnormal PCG signals and there are several peaks in between 300 Hz to 600 Hz. This reflects that the simple frequency domain feature can help significantly in classifying the PCG signals. However, more t-domain, f-domain, and Mel frequency cepstral coefficients (MFCC) provide insight on the signal while compensating for the noise or motion arte facts. Moreover, it has been shown in the literature that the MFCC features can contribute significantly in classifying the sound waves. Therefore, 27 features encompassing t-domain, f-domain, and MFCC features were extracted for each heart sound cycle (Table 1). The t-domain, f-domain and MFCC features used in this study are taken based on the previous works. The t-domain features were: Mean value, median value, standard deviation, mean absolute deviation, signal 25th percentile, signal 75th percentile, signal inter quartile range, skewness, kurtosis, and Shannon's entropy; whereas the f-domain features were: Spectral entropy, maximum frequency in the power spectrum, signal magnitude at maximum frequency, and ratio of signal energy between maximum frequency range and overall signal. The rest of the features were MFCC features.

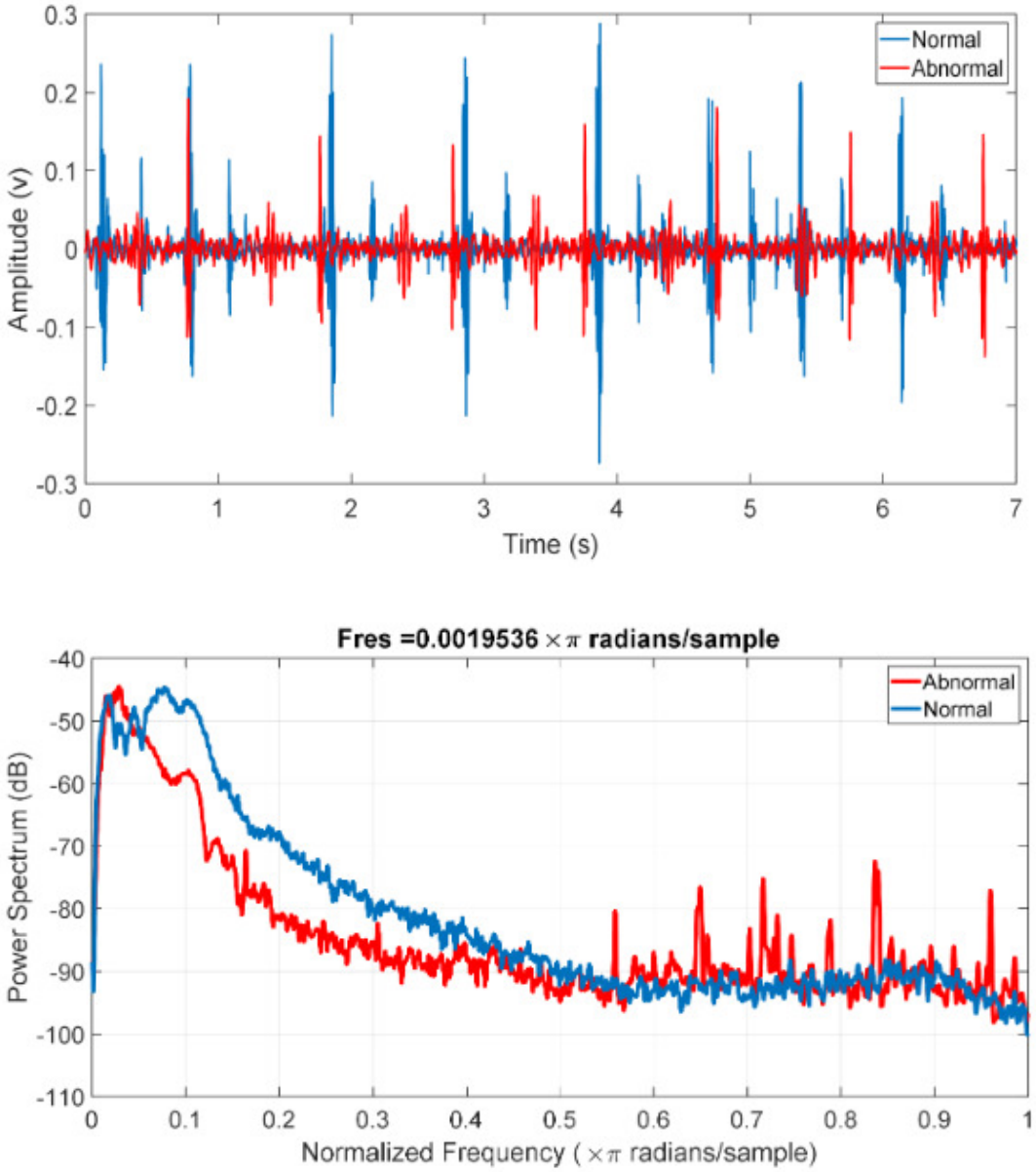


Figure VI.7 Time domain PCG trace and its power spectral density for normal and abnormal subjects.

Feature	Definition	Equation
Mean	Sum of all data divided by the number of entries.	$\bar{x}$ $\frac{\sum x}{n}$
Median	Value that is in the middle of ordered set of data.	Odd numbers of entries: Median – middle data entry. Even numbers of entries: Median – adding the two numbers in the middle and dividing the result by two.
Standard Deviation	Measure variability and consistency of the sample.	$s - \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$
Percentile	The data value at which the percent of the value in the data set are less than or equal to this value.	25^{th} $\left(\frac{25}{100}\right)n$ 75^{th} $\left(\frac{75}{100}\right)n$
Mean Absolute Deviation	Average distance between the mean and each data value.	MAD – $\frac{\sum_{i=1}^{n}\lvert x_i-\bar{x}\rvert}{n}$
Inter Quartile Range	The measure of the middle 50% of a data set.	IQR – $Q_3 - Q_1$ Q_3: third quartile, Q_1: first quartile, Quartile: dividing the data set into four equal portions.
Skewness	The measure of the lack of symmetry from the mean of the dataset.	g1 $\frac{\sum_{i=1}^{N}(Y_i-Y)^3/N}{s^3}$ Y: mean, s: the standard deviation, N: number of the data.
Kurtosis	The pointedness of a peak in distribution curve, in other words it's the measure of sharpness of the peak of distribution curve.	k – $\frac{\sum_{i=1}^{N}(Y_i-Y)^4/N}{s^4} - 3$ Y: mean, s: the standard deviation, N: the number of data.
Shannon's Entropy	Entropy measures the degree of randomness in a set of data, higher entropy indicates a greater randomness, and lower entropy indicates a lower randomness.	$H(x) - -\sum_{i=0}^{N-1} p_i \log_2 p_i$
Spectral Entropy	The normalized Shannon's entropy that is applied to the power spectrum density of the signal.	SEN – $\frac{-\sum_{i=0}^{N-1} p_k \log_2 p_k}{\log N}$ p_k: the spectral power of the normalized frequency, N: the number of frequencies in binary
Maximum Frequency	The value of highest frequency in the signal spectrum	f_{max}
Magnitude at Fmax	Signal magnitude at highest Frequency	$X(f_{max})$
Ratio of signal energy	Ratio of signal energy between $f_{max} \pm \Delta f$ and the whole spectrum	$X(f_{max} \pm \Delta f) / \sum_{i=0}^{N-1} X_i(f)$
MFCC (13 features)	Mel-Frequency Cepstral Coefficients (MFCC): coefficients that collectively make up a Mel-Frequency Cepstral (MFC).	x – x – 0.95*[0; x (1: N-1)]; X – fft(x);

Table VI.1. Extracted features.

VI.2.1.4 Classification

A large variety of machine learning (ML) algorithms can be used for classifying the HS signals into normal and abnormal. In the subsequent section, we will discuss the training and validation of different ML model, and testing of the pre-trained best performing model for classification of HS signals to detect abnormality in a real-time manner.

VI.2.1.5 Performance Evaluation Matrix

The benchmark dataset was randomly partitioned into two subsets: (i) Training and validation set (80% of data), and (ii) testing set (20% of data). The ML model was trained for two classes: "normal"

and "abnormal". To compare the performance of several ML algorithms in classifying PCG signals, confusion matrices for each algorithm after 5-fold cross-validation were created and several standard statistical evaluation parameters were calculated to evaluate the performance of the algorithms: True Positive Rate (TPR)/Recall/Sensitivity:

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

Specificity:

$$Specificity = \frac{TN}{TN + FP}$$

False Positive Rate (FPR):

$$FPR = 1 - Specificity = \frac{FP}{TN + FP}$$

Precision:

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

F-measure or score:

$$F\ score = \frac{2 * Recall * Precision}{Recall\ +\ Precision}$$

Accuracy:

$$Accuracy\ (ACC) = \frac{TP + TN}{P + N}$$

Error:

$$Error = 1 - \text{Accuracy}$$

$$\text{Matthews correlation coefficient, } MCC = \frac{(TP * TN - FP * FN)}{((TP + FP) * P * N * (TN + FN))^{0.5}}$$

Where:

TP: True Positive Rate **TN**: True Negative Rate

FP: False Positive Rate **FN**: False Negative Rate

P=TP+FN **N=FP+TN**

The above-mentioned parameters were estimated using 5-fold cross validation such that the training database was divided into five equal sets. Out of five sets, four sets were used for training while one set was used for testing. This process is repeated five times such that each set is tested once. The final results are obtained by averaging the results of all the iterations. The average of all the above-mentioned parameters was calculated. Performance evaluation of three different best-performing ML algorithms was calculated to identify best one in the testing phase.

VI.2.1.6 Feature Reduction

Neighborhood component analysis (NCA) is a non-parametric and embedded method for selecting features with the goal of maximizing prediction accuracy of classification algorithms. The Statistics and Machine Learning Toolbox™ built-in functions can be used to perform NCA feature selection with regularization to learn feature weights for minimization of an objective function that measures the average leave-one-out classification loss over the training data. It was found that the most

contributory features are 15 features out of the 27 features. These are kurtosis, maximum frequency value, and all of the MFCC features. The ML algorithms were trained again with the same training data subset with the reduced feature matrix to see whether this feature reduction can improve the classification accuracy by reducing over-fitting or not. Performance measures were calculated for the three best-performing algorithms to identify the best one for the testing data subset.

VI.2.1.7 Hyperparameter Optimization of the Best-Performing Algorithm

Each of the trained machine learning algorithms were trained with some default parameters, which produce particular validation accuracy; however, these algorithms can be tuned to optimize their hyperparameters. It would be a very tedious task to tune all the algorithms trained to check the validation accuracy. Therefore, the best-performing algorithms were optimized to calculate the performance measures. The accuracy and other performance measures were then calculated for the testing dataset (20% of the whole database).

VI.2.1.8 Unequal Misclassification Costs

It is apparent from Table VI.2 that the number of abnormal and normal observations for both the training and testing datasets are unequal or the dataset is imbalanced. Moreover, misclassifying observations of abnormal class has more severe consequences than misclassifying observations of normal class. Since the classes are adequately represented in the training data but we have to treat them asymmetrically, the cost of classes was made different. Since we want to classify patients with normal and abnormal heart sounds, failure to identify an abnormal class (false negative) has far more severe consequences than misidentifying normal class as abnormal (false positive). We have assigned 10 times more cost to misidentifying abnormal HS as normal HS and low cost to misidentifying normal HS as abnormal HS. We have trained best-performing classification algorithm with unequal classification costs to increase accuracy for abnormal HS with higher accuracy by partially scarifying the accuracy of the normal HS classification. This partial reduction in the accuracy in normal HS classification should be acceptable, as this is only a screening test.

	Categories	No. of Observation
Training and Validation	Abnormal Normal	2505 7907
Testing	Abnormal Normal	653 1950

Table VI.2 Dataset observation

VI.2.2 Results

VI.2.2.1 Performance Evaluation of Machine Learning Abnormality Detection Algorithm

Twenty-two different algorithms (three decision tree, two discriminant analysis, six support vector machines (SVM), six k-nearest neighbor (KNN), and five ensembles classifiers) were trained using27 features of the training dataset (80% of the whole dataset). The validation accuracy and their corresponding performance measures are listed in Table VI.3.b

K-Nearest Neighbor classification:

it is categorizing query points based on their distance to points (or neighbors) in a training dataset can be a simple yet effective way of classifying new points. You can use various metrics to determine the distance. Given a set X of n points and a distance function, k-nearest neighbor (kNN) search lets us find the k closest points in X to a query point or set of points. [21]

Deference between Fine KNN and Weighted KNN:

Classifier Type	Prediction Speed	Memory Usage	Interpretability	Model Flexibility
Fine KNN	Medium	Medium	Hard	Finely detailed distinctions between classes. The number of neighbors is set to 1.
Weighted KNN	Medium	Medium	Hard	Medium distinctions between classes, using a distance weight. The number of neighbors is set to 10.

Table VI.3.a fine KNN and weighted KNN [21]

Items	Fine KNN	Weighted KNN	Ensemble Subspace Discriminant
Accuracy	94.63%	93.72	93.17%
Accuracy: Abnormal	88%,12%	85%,15%	87%, 13%
Accuracy: Normal	96.6%, 3.4%	97%,3%	95%, 5%
Error	5.37%	6.28%	6.83%
Sensitivity	96.32%	95.24%	95.67%
Specificity	89.34%	88.72%	85.49%
Precision	96.62%	96.54%	95.29%
FPR	10.66%	11.28%	14.51%
F_Score	96.46%	95.88%	95.48%
MCC	85.34%	82.7%	81.5%

Table VI.3.b Performance measures of three best performing algorithms for full-feature set.

It is obvious from the above table that the best validation accuracy was observed for "Fine Tree" classifier. Moreover, the accuracy of classifying normal is higher than abnormal and this is because of the imbalanced dataset as shown in Table VI.2. Therefore, we needed to check the potential over-fitting of the features. This could be dealt with by reducing the number of features used in the

training process. Therefore, the training dataset was retrained with the reduced number of features (15) and the confusion matrix and evaluation measures were calculated. Table VI.4 summarizes the accuracy and other evaluation measures for identifying the best algorithm after feature reduction.

Items	Fine KNN	Weighted KNN	Ensemble Subspace Discriminant
Accuracy	92.36%	92.02%	92.89%
Accuracy: Abnormal	84%,16%	82%,18%	83%, 17%
Accuracy: Normal	95%, 5%	95%,5%	96%, 4%
Error	7.64%	7.98%	7.11%
Sensitivity	94.85%	94.30%	94.77%
Specificity	84.52%	84.62%	86.71%
Precision	95.08%	95.22%	95.90%
FPR	15.48%	15.38%	13.29%
F_Score	94.96%	94.76%	95.33%
MCC	79.17%	78.09%	80.42%

Table VI.4 Performance measures of three best performing algorithms for reduced-feature set.

However, it is apparent from Table 4 that the overall accuracy was reduced, and classifying normal and abnormal were also both reduced even though the same algorithms were performing best in the classification after feature reduction. Therefore, it can be said that the features used for classification are optimized and cannot be reduced. To improve the performance of the best-performing algorithms by optimizing the hyperparameters of the algorithms, it was observed that the performance of the ensemble algorithm can be improved. Two important parameters were optimized for the ensemble algorithms: "Distance" and "Number of neighbors". Figure VI.8 shows the optimization of two parameters and Figure VI.9 shows the number of required iterations to reach the objective.

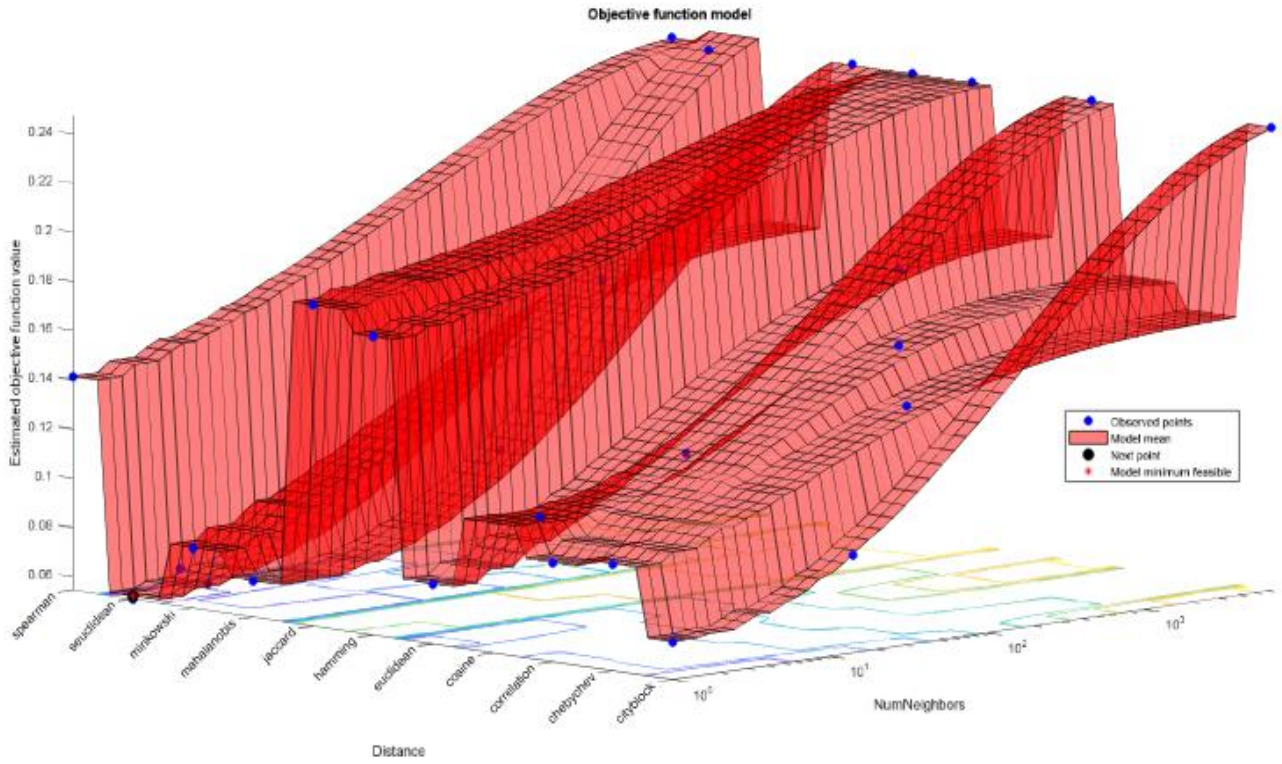


Figure VI.8 Optimization of hyperparameter for ensemble algorithm

Iter	Eval result	Objective	Objective runtime	BestSoFar (observed)	BestSoFar (estim.)	Method	NumLearningCycles	LearnRate
1	Best	0.14186	115.34	0.14186	0.14186	GentleBoost	87	0.44096
2	Accept	0.17403	79.574	0.14186	0.14313	RUSBoost	22	0.0039408
3	Best	0.1262	228.49	0.1262	0.12803	GentleBoost	232	0.0067354
4	Accept	0.16548	89.179	0.1262	0.12626	AdaBoostM1	18	0.35649
5	Best	0.11861	384.71	0.11861	0.11862	GentleBoost	472	0.015861
6	Best	0.091433	1352.1	0.091433	0.091437	Bag	499	-
7	Accept	0.093258	157.1	0.091433	0.091495	Bag	53	-
8	Accept	0.2426	432.97	0.091433	0.091578	LogitBoost	456	0.0010057
9	Accept	0.25941	989.61	0.091433	0.091638	AdaBoostM1	248	0.0010009
10	Accept	0.11977	122.68	0.091433	0.091623	RUSBoost	34	0.97844
11	Accept	0.11035	35.088	0.091433	0.091613	LogitBoost	33	0.99605
12	Accept	0.10142	450.59	0.091433	0.091608	LogitBoost	494	0.15808
13	Accept	0.11362	1127.2	0.091433	0.091599	RUSBoost	465	0.1263
14	Accept	0.1506	9.021	0.091433	0.091437	LogitBoost	11	0.33469
15	Accept	0.093354	407.7	0.091433	0.091438	LogitBoost	495	0.93427

Figure VI.9 Number of evaluations to reach minimum objective

The best-performing and hyperparameter-optimized algorithm was retrained with cost adjustment to make sure that the "abnormal HS" will be classified with more confidence than the "normal HS". The result of this asymmetric cost adjustment it is evident that the classification accuracy of abnormal HS became approximately 97% while that is for normal HS is approximately 88%. The team with the highest score of the PhysioNet-2016 challenge competition reported sensitivity, specificity, and overall scores of 94.24%, 77.81%, and 86.02%, respectively; however, in this work, an overall sensitivity, specificity, and accuracy of 96.32%, 89.34%, and 94.63%, respectively were attained. It therefore can be observed that the trained model can much more reliably classify abnormal HS sound with 96.68% accuracy and normal HS with 87.87% for the testing dataset, as shown in Figure VI.10.

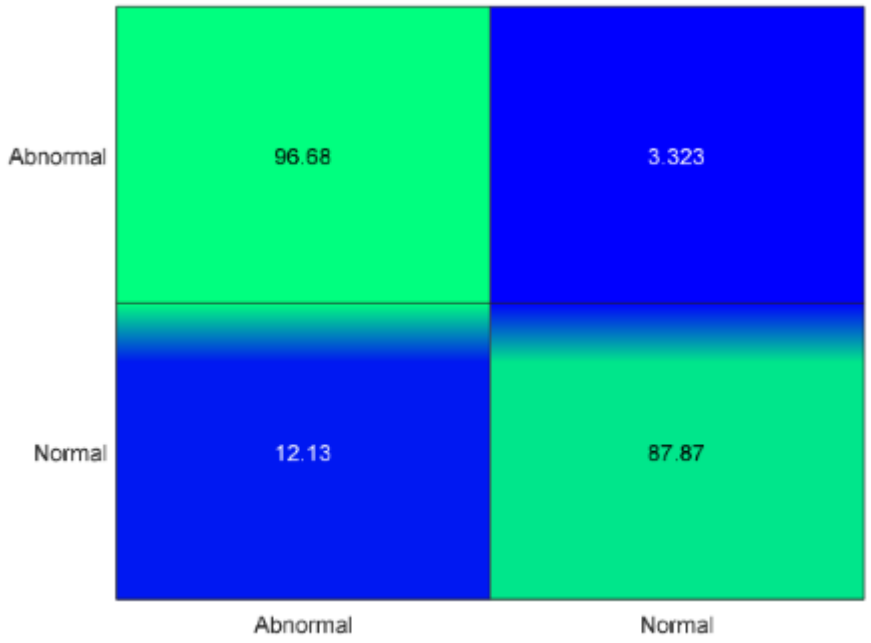


Figure VI.10 Confusion matrix for hyperparameter optimized ensemble algorithm for test dataset.

VI.2.2.2 Real-Time Classification of Heart Sound Signals

The second python code was used to save heart signals , which were used in this Matlab code in real time classification .

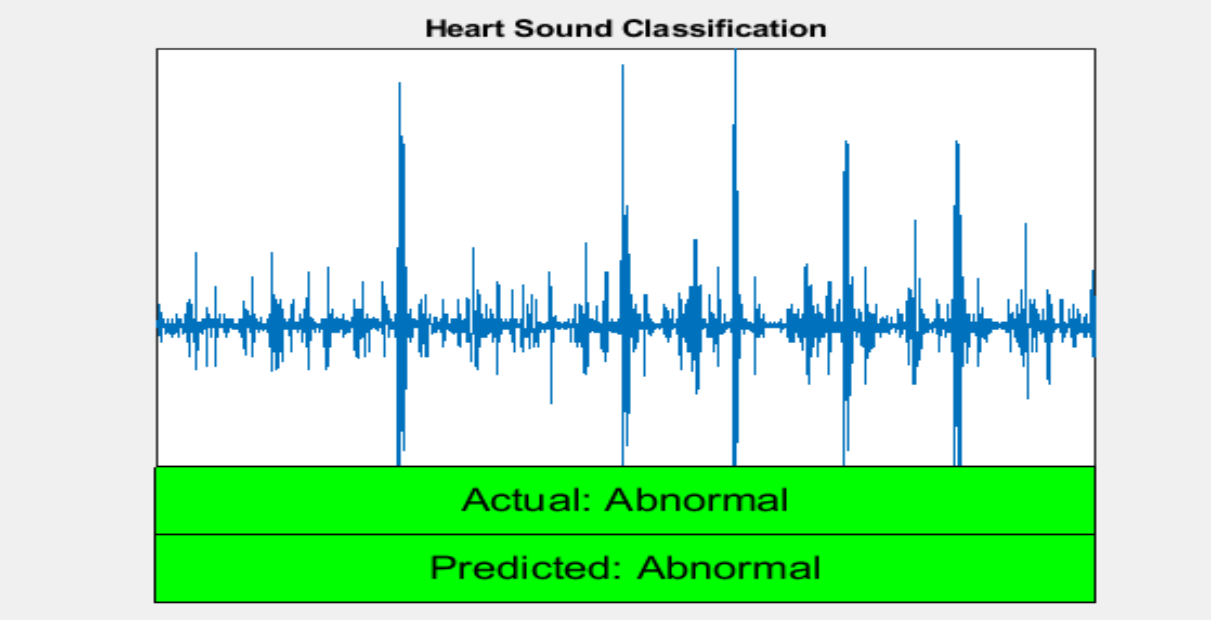


Figure VI.11 Graphical user interface for real-time HS classification using MATLAB

General conclusion

We have proposed and implemented a node for heart sound capturing system for real-time heart sound anomaly detection. This project was designed by modifying an analog stethoscope and adding an analog front end and miniaturized microcontroller with wifi transmission. By using this device, the user can keep track of his/her heart condition on a daily basis, at low cost. A public large imbalanced dataset was used to train and test the algorithm with 27 t-domain, f-domain, and MFCC features. The best-performing algorithms in terms of classification accuracy were reported with several other statistical performance measures. Feature reduction, hyperparameter optimization, along with asymmetrical cost assignment in the training of algorithm were evaluated to obtain best performance from the algorithms. It was observed that the optimized Ensemble algorithm can outperform all the trained algorithm in classifying the test data subset. The highest score of the PhysioNet-2016 challenge competition reported over all accuracy of 86.02%, where as the work reported has achieved a higher accuracy of 94.63% (97% abnormal and 88% normal). The classification accuracies with the cost adjustment were found to be 97% and 88%for detecting abnormal HS and normal HS, respectively. In summary, the device can contribute to excellent health monitoring and improve personal care of cardiac patients at home in a completely non invasive manner.

In the future, we would like to make this project more compact in size and more professional looking with an embedded decision-making unit to classify the HS on-board. The system might be modified to classify the HS signals real-time and display results on-screen interactively, which can be a new life-saving gadget.

Bibliographical references:

[1]:ELECTRONIC STETHOSCOPE AND HEART RATE MONITOR NICOSIA-2015
[2]:https://en.wikipedia.org/wiki/Heart
[3]:https://en.wikipedia.org/wiki/Heart_sounds
[4]:https://opentextbc.ca/anatomyandphysiology/chapter/19-3-cardiac cycle/
[5]: Yasser Ben Aissia et Ghassene Chaieb. Gateway d'un système demonitoring d'un malade à un serveur d'urgence. Master's thesis, Ecole Nationaled'Ingénieurs de Tunis,2015.
[6]:CHIOUKH ABDELAZIZ CHOUGAR RACHID Thème Conception et réalisation d'un stéthoscope Numérique avec Arduino Mémoire soutenu publiquement le 11 juillet 2017
[7]: MCP3008-I/P Fiche technique(HTML)-Microchip Technology.pdf
[8]: BOUHARAOUA AbderrahimMr,BOUKLI HACENE Mohammed Imad - Automatisation d'une maison intelligente via une application Android
[9]: https://fr.wikipedia.org/wiki/Node.js
[10]: https://fr.wikipedia.org/wiki/Apache_HTTP_Server
[11]: https://en.wikipedia.org/wiki/MySQL
[12]: https://fr.wikipedia.org/wiki/Python_(langage)
[13]: https://www.phpmyadmin.net/
[14]: Transmission of Audio Over Wireless Networks,Using Raspberry Pi in Real-Time,Nalini Bagal, Prof. Shivani Pandita
[15]:MCP3008 datasheet.pdf
[16]: nternational Journal of Electronics, Electrical and Computational System,IJEECS,ISN 2348-17X,Volume 6, Isue 7,July 2017,Electronic Stethoscope with Pulse Monitoring On Online
Server
[17]: International Journal of Computer Applications (0975 – 8887),Volume 137 – No.10, March 2016,33
Design of Electronic Stethoscope and Heart Rate
Monitor for Remote Area Application
[18]: Ahmad Nabil bin Md Nasir, "Portable Medical Electronic WorkbenchMeasuring Sp02 And Heart Beat". Bachelor of Electrical Engineering(Electronics). Faculty of Electrical Engineering, Universiti TeknologiMalaysia, Skudai.. 2009.
[19]: SMART WIRELESS ELECTRONIC STETHOSCOPE,K M Manoj Kumar, K Lokesh Naidu,K Pavan Kumar,Bhimesh,T S Deepthi Murthy5
[20]: Mohamed Lamine MEKHALFI Thème Analyse Des Signaux PCG Par L'usage De la CWT et De la D
[21]: https://www.mathworks.com/help/stats/choose-a-classifier.html
[22] : https://makersportal.com/blog/2018/8/23/recording-audio-on-the-raspberry-pi-with-python-and-a-usb-microphone
[23] : https://swharden.com/blog/2016-07-19-realtime-audio-visualization-in-python

ANNEXES

Annex 1 . Python streaming code [23]

```
import pyaudio
import time
import matplotlib.pyplot as pylab
import numpy as np
class SWHear(object):
    """
    The SWHear class is made to provide access to continuously recorded
    (and mathematically processed) microphone data. ""
    def __init__(self,device=None,startStreaming=True):
        """fire up the SWHear class."""
        print(" -- initializing SWHear")
        self.chunk = 4096 # number of data points to read at a time
        self.rate = 44100 # time resolution of the recording device (Hz)
        # for tape recording (continuous "tape" of recent audio)
        self.tapeLength=2 #seconds
        self.tape=np.empty(self.rate*self.tapeLength)*np.nan
        self.p=pyaudio.PyAudio() # start the PyAudio class
        if startStreaming:
            self.stream_start()
    ### LOWEST LEVEL AUDIO ACCESS
    # pure access to microphone and stream operations
    # keep math, plotting, FFT, etc out of here.
    def stream_read(self):
        """return values for a single chunk"""
        data = np.fromstring(self.stream.read(self.chunk),dtype=np.int16)
        #print(data)
        return data
    def stream_start(self):
        """connect to the audio device and start a stream"""
        print(" -- stream started")
        self.stream=self.p.open(format=pyaudio.paInt16,channels=1,
                                rate=self.rate,input=True,
                                frames_per_buffer=self.chunk)
    def stream_stop(self):
        """close the stream but keep the PyAudio instance alive."""
        if 'stream' in locals():
            self.stream.stop_stream()
            self.stream.close()
        print(" -- stream CLOSED")
    def close(self):
        """gently detach from things."""
        self.stream_stop()
        self.p.terminate()
    ### TAPE METHODS
    # tape is like a circular magnetic ribbon of tape that's continously
    # recorded and recorded over in a loop. self.tape contains this data.
```

```
    # the newest data is always at the end. Don't modify data on the type,
    # but rather do math on it (like FFT) as you read from it.
    def tape_add(self):
        """add a single chunk to the tape."""
        self.tape[:-self.chunk]=self.tape[self.chunk:]
        self.tape[-self.chunk:]=self.stream_read()
    def tape_flush(self):
        """completely fill tape with new data."""
        readsInTape=int(self.rate*self.tapeLength/self.chunk)
        print(" -- flushing %d s tape with %dx%.2f ms reads"%\
              (self.tapeLength,readsInTape,self.chunk/self.rate))
        for i  in range(readsInTape):
            self.tape_add()
    def tape_forever(self,plotSec=.25):
        t1=0
        try:
            while True:
                self.tape_add()
                if (time.time()-t1)>plotSec:
                    t1=time.time()
                    self.tape_plot()
        except:
            print(" ~~ exception (keyboard?)")
            return
    def tape_plot(self,saveAs="./public/03.png"):
        """plot what's in the tape."""
        pylab.plot(np.arange(len(self.tape))/self.rate,self.tape)
        pylab.axis([0,self.tapeLength,-2**16/2,2**16/2])
        if saveAs:
            t1=time.time()
            pylab.savefig(saveAs,dpi=100)
            print("plotting saving took %.02f ms"%((time.time()-t1)*1000))
        else:
            pylab.show()
            print() #good for IPython
        pylab.close('all')
if __name__=="__main__":

    ear=SWHear()

    ear.tape_forever()

    ear.close()

    print("DONE")
```

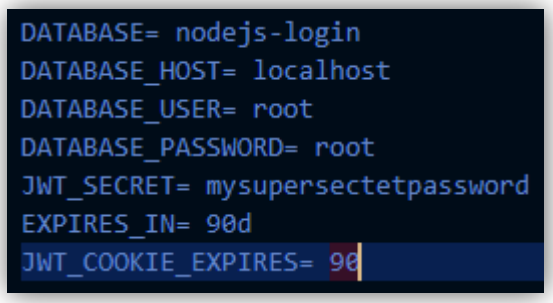
Annex 2 . Python recording code [22]

```
import pyaudio
import wave
form_1 = pyaudio.paInt16 # 16-bit resolution
chans = 1 # 1 channel
samp_rate = 44100  # 44.1kHz sampling rate
chunk = 4096 # 2^12 samples for buffer
record_secs = 3 # seconds to record
dev_index = 2 # device index found by p.get_device_info_by_index(ii)
wav_output_filename = 'public/file.wav' # name of .wav file
audio = pyaudio.PyAudio() # create pyaudio instantiation
# create pyaudio stream
stream = audio.open(format = form_1,rate = samp_rate,channels = chans, \
            input_device_index = dev_index,input = True, \
            frames_per_buffer=chunk)
print("recording")
frames = []
# loop through stream and append audio chunks to frame array
for ii in range(0,int((samp_rate/chunk)*record_secs)):
   data = stream.read(chunk)
   frames.append(data)
print("finished recording")
# stop the stream, close it, and terminate the pyaudio instantiation
stream.stop_stream()
stream.close()
audio.terminate()
# save the audio frames as .wav file
wavefile = wave.open(wav_output_filename,'wb')
wavefile.setnchannels(chans)
wavefile.setsampwidth(audio.get_sample_size(form_1))
wavefile.setframerate(samp_rate)
wavefile.writeframes(b''.join(frames))
wavefile.close()
```

Annex 3 . Application main script (app.js)

```
const express = require("express");
const path= require('path');
const mysql = require("mysql");
const dotenv = require("dotenv");
const cookieParser=require('cookie-parser');
dotenv.config({path: './.env'});
const app = express();
const db = mysql.createConnection({
   host: process.env.DATABASE_HOST,
   user: process.env.DATABASE_USER,
   password : process.env.DATABASE_PASSWORD,
   database: process.env.DATABASE
});
const publicDirectory= path.join(__dirname, './public')
app.use(express.static(publicDirectory));
app.use(express.urlencoded({extended: false}));
app.use(express.json());
app.use(cookieParser());
app.set('view engine','hbs');
db.connect((error)=>{
  if(error){
     console.log(error)
  } else{
     console.log("MYSQL connection..")
  }
});
// Define Routes
app.use('/',require('./routes/pages'));
app.use('/auth', require('./routes/auth'));
app.listen(5888,() => {
console.log("Server started on port 5888");
});
```

Annex 4. Configuration script file (.env)



Annex 5. index.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="stethoscope-icon.png" type="image/x-icon" rel="shortcut icon">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
   <link rel="stylesheet" href="/style.css">
   <title>HEART BEAT VISUALISATION</title>
</head>
<body>
  <nav>
     <h4>HEART BEAT VISUALISATION</h4>
      <ul>
         <li><a href="/"> Home  </a></li>
         <li><a href="/login"> Login </a></li>
         <li><a href="/register"> Register </a></li>
      </ul>
  </nav>
  <div class="container mt-4">
     <div class="jumbotron">
        <h1 class="display-4">MASTER PROJECT</h1>
      <p class="lead">This simple project made by DJENDLI MAROUA and her supervisor Mr.FIZZARI
MOURAD  </p>
        <hr class="my-4">
        <p> It's for anyone who needs it, use it with care  </p>
     </div>
  </div>
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
</body>
</html>
```

Annex 6. inform.hbs

```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8">
    <link href="stethoscope-icon.png" type="image/x-icon" rel="shortcut icon">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
    <title>HEART BEAT VISUALISATION</title>
   <script language="javascript">
        function RefreshImage(){
       document.pic0.src="03.png?a=" + String(Math.random()*99999999);
        setTimeout('RefreshImage()',50);
        }
   </script>
 </head>
 <body  onload="RefreshImage()" >
    <nav>
      <h4>HEART BEAT VISUALISATION</h4>
       <ul>
          <li><a href="/profile"> Profile  </a></li>
          <li><a href="/inform"> Personal Information </a></li>
          <li><a href="/login"> Logout  </a></li>
       </ul>
    </nav>
     <div  align = 'center'  >
      <img name="pic0" src="03.png"  >
     </div>
    <footer  >
       <div id="fix" class="pos" >
         <ul >
           <li> <div id="text"><h4>Save as :  </h4></div></li>
           <li><a href="/03.png" download="traitment figure.jpg">JPG </a></li>
           <li><a href="/03.png" download="traitment figure.png">PNG </a></li>
```

```
            <li>
              <a href="/file.wav" download="traitment figure.wav">save a wav file </a>
            </li>
          </ul>
        </div>
      </footer>
  </body>
</html>
```

Annex 7. login.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="stethoscope-icon.png" type="image/x-icon" rel="shortcut icon">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
   <link rel="stylesheet" href="/style.css">
   <title>HEART BEAT VISUALISATION</title>
</head>
<body>
  <nav>
     <h4> HEART BEAT VISUALISATION </h4>
      <ul>
        <li><a href="/"> Home </a></li>
        <li><a href="/login"> Login </a></li>
        <li><a href="/register"> Register </a></li>
      </ul>
  </nav>
  <div class="container mt-4">
     <div class="card">
        <div class="card-header">
          Login Form
        </div>
        <div class="card-body">
          <form action="/auth/login" method="POST">
            <div class="form-group">
              <label for="email">Email Address</label>
              <input type="email" class="form-control" id="email" name="email">
            </div>
            <div class="form-group">
              <label for="password">Password</label>
              <input type="password" class="form-control" id="password" name="password">
            </div>
```

```
            <button type="submit" class="btn btn-primary">login</button>
          </form>
        </div>
    </div
    {{#if message}}
    <h4 class= "alert alert-danger mt-4"> {{message}} </h4>
    {{/if}}
  </div>
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script></body> </html>
```

Annex 8. profile.hbs

```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8">
    <link href="stethoscope-icon.png" type="image/x-icon" rel="shortcut icon">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
    <link rel="stylesheet" href="style.css">
     <title> HEART BEAT VISUALISATION</title>
  </head>
  <body>
    <nav>
      <h4>HEART BEAT VISUALISATION</h4>
       <ul>
          <li><a href="/profile"> Profile  </a></li>
          <li><a href="/inform"> Personal Information </a></li>
          <li><a href="/login"> Logout  </a></li>
       </ul>
    </nav>

  <div id ="text">
    <p> <h1>     Welcom to your profile </h1>
      <h4> To use this device follow these instructions : </h4>
    </br>
        first :
          </br>
          if you want from your doctor to treat you in real-time , then you </br>
          must wear this divice under your clothes (make it directlly attached
         </br>
          to your skin), and don't forget to turn it on!
         </br>
         </br>
         </br>  second :
         </br>
```

```
    if you want to take an advice from your doctor in real-time too , then you </br>
    must put this divice under your clothes (make it directlly attached to  your</br>
    skin), and don't forget to turn it on . after few seconds you can save an </br>
    image of your heart-beat . finally turn it off , and send your image to any</br>
    doctor you want.
        </br>
       </p>
   </div>
   </body>
</html>
```

Annex 9. register.hbs

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="stethoscope-icon.png" type="image/x-icon" rel="shortcut icon">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
   <link rel="stylesheet" href="/style.css">
   <title>HEART BEAT VISUALISATION</title>
</head>
<body>
  <nav>
     <h4> HEART BEAT VISUALISATION </h4>
      <ul>
        <li><a href="/"> Home  </a></li>
        <li><a href="/login"> Login </a></li>
        <li><a href="/register"> Register </a></li>
      </ul>
  </nav>
  <div class="container mt-4">
      <div class="card">
        <div class="card-header">
         Register Form
        </div>
        <div class="card-body">
          <form action="/auth/register" method="POST">
            <div class="form-group">
              <label for="name">Name</label>
              <input type="name" class="form-control" id="name" name="name" >
            </div>
            <div class="form-group">
              <label for="email">Email Address</label>
              <input type="email" class="form-control" id="email" name="email">
            </div>
```

```
            <div class="form-group">
              <label for="password">Password</label>
              <input type="password" class="form-control" id="password" name="password">
            </div>
             <div class="form-group">
              <label for="password">Password Confirm</label>
              <input type="password" class="form-control" id="passwordConfirm"
name="passwordConfirm">
             </div>
            <button type="submit" class="btn btn-primary">Register User</button>
          </form>
 </div>
    </div>
    {{#if message}}
    <h4 class= "alert alert-danger mt-4"> {{message}} </h4>
    {{/if}}
  </div>
   <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
</body>
</html>
```

Annex 10. auth.js(from controllers folder)

```
const express = require("express");
const path = require('path');
const mysql = require("mysql");
const dotenv = require("dotenv");
const jwt = require('jsonwebtoken');
const bcrypt =require('bcryptjs');
dotenv.config({path: './.env'});
const app = express();
const db = mysql.createConnection({
   host: process.env.DATABASE_HOST,
   user: process.env.DATABASE_USER,
   password : process.env.DATABASE_PASSWORD,
   database: process.env.DATABASE
});
exports.login= async (req,res) => {
 try {const{email , password} =req.body;
     if( !email || !password){
       return res.status(400).render('login',{
          message: 'Please provide an email and password'
       });
      }
db.query('SELECT * FROM users WHERE email = ?', [email] , async (error,results) => {
       console.log(results);
        if( !results || !(await bcrypt.compare(password , results[0].password)) )
        {
          res.status(401).render('login', {
          message: 'Email or password is incorrect'
```

```
        })
        } else{
          const id = results[0].id;
          const token = jwt.sign({ id }, process.env.JWT_SECRET,{
              expiresIn: process.env.EXPIRES_IN
          });
          console.log("The token is :" + token);
          const cookieOpthions={
             expires: new Date (
               Date.now()+process.env.JWT_COOKIE_EXPIRES * 24 * 60 * 60 * 1000
             ),
             httpOnly: true}
          res.cookie('JWT', token ,cookieOpthions);
          res.status(200).redirect("/profile"); }
       });
      } catch (error) {
    console.log(error); }
  }
exports.register = (req,res) => {
   console.log(req.body);
   const {name , email , password , passwordConfirm} = req.body;
db.query('SELECT email FROM users WHERE email = ?',[email], async (error,results) =>{
      if(error){
         console.log(error) ; }
      if(results.length > 0){
          return res.render('register',{
            message: 'email has been taken'
          })
       }else if (password !== passwordConfirm){
          return res.render('register',{
             message:'Passwords do not match'
          })
       }
     let hashedPassword = await bcrypt.hash(password, 8);
     console.log(hashedPassword);
  db.query('INSERT INTO users SET ?',{name:name
,email:email,password:hashedPassword},(error,results)=>{if(error){
           console.log(error);
        }else {
          console.log(results);
          return res.render('register',{
             message:'User registred'
          });
        }
     })
   }); }
```

Annex 11. auth.js(from routes folder)

```
const express = require ('express');
const authController =require('../controllers/auth');
const router = express.Router();
```

```
router.post('/register', authController.register);

router.post('/login' , authController.login);

module.exports = router;
```

Annex 12. pages.js

```
const express = require ('express');
const router = express.Router();
router.get('/',(req,res) =>{
  res.render('index')
});
router.get('/register',(req,res)=>{
   res.render('register')
});
router.get('/login',(req,res) =>{
   res.render('login')
});
router.get('/profile',(req,res) =>{
   res.render('profile')
});
router.get('/inform',(req,res) =>{
   res.render('inform')
});
module.exports = router;
```

Annex 13. style.css

```
nav{          background-color: #ca736c;
            color : #fff;
            padding: 30px 60px;
            display: flex;
            justify-content: space-between;}
nav ul{       display: flex;
            justify-content: space-around;
            align-items:center;}
nav li{       list-style:none;}
nav li a {     color:#fff;
            text-decoration:none;
            font-weight: bold;
            padding: 5px 8px;}
nav li a:hover {
           color: rgb(63, 13, 63);
           text-decoration:none;}
```

```
footer{     background-color: #ca736c;
            position:absolute;
            bottom:0;
            width:100%;
            padding-top:50px;
            height:50px;   }
div#fix {    position:fixed;
            bottom:0;       }
#fix ul{     display: flex;
            justify-content: space-around;
            align-items:center;}
#fix li{     list-style:none;}
#fix li a {   color:#fff;
            text-decoration:none;
            font-weight: bold;
            padding: 5px 8px;}
#fix li a:hover {
            color: rgba(19, 218, 141, 0.37);
            text-decoration:none;}
div#text{    color : rgb(104, 72, 72);
            text-align-last: center;}
```

Annex 14. MATLAB code

```
%%%% definitely make sure we've got the validation data set%%%%%%%%%%%%%%%%%%
if ~exist('validation.zip')%#ok
  % you need the validation data only for running the prototype app
  try
     validation_url = 'https://archive.physionet.org/pn3/challenge/2016/validation.zip';
     websave('validation.zip', validation_url);
  catch
     warning('Failed to access heart sound validation data on physionet.org - check whether path
%s needs updating', validation_url)
  end
  unzip('validation.zip', 'Data');
end
% by default, skip downloading training data (may take long time, 185 MB)
% (though you won't be able to execute the feature extraction yourself below)
getTrainingData = false;
if ~exist('training.zip') && getTrainingData%#ok
   % fetch training data from physionet site.
   % NOTE: unless you plan to execute the feature extraction, don't worry if there is an error here,
   %       we only need access to the training set to run the feature extraction
   try
      training_url = 'https://archive.physionet.org/pn3/challenge/2016/training.zip';
      websave('training.zip', training_url);
   catch
      warning('Failed to access heart sound training data on physionet.org - check your internet
connection or whether path %s needs updating',training_url)
   end
   unzip('training.zip', 'Data/training')
end
```

```
% make sure we have copies of the two example files in the main directory
if exist('Data/validation')%#ok
   copyfile 'Data/validation/a0001.wav';
   copyfile 'Data/validation/a0011.wav';
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addpath(genpath(pwd));
addpath('./HelperFunctions');  warning off;                  % suprress warning messages
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[PCG_abnormal, fs] = audioread('a0001.wav');
p_abnormal = audioplayer(PCG_abnormal, fs);
play(p_abnormal, [1 (get(p_abnormal, 'SampleRate') * 3)]);
plot(PCG_abnormal(1:fs*3)) % Plot the sound waveform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[PCG_normal, fs] = audioread('a0011.wav');
p_normal = audioplayer(PCG_normal, fs);
play(p_normal, [1 (get(p_normal, 'SampleRate') * 3)]);
plot(PCG_normal(1:fs*3)) % Plot the sound waveform
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
signalAnalyzer(PCG_normal, PCG_abnormal)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% expecting the training data in subfolders of 'Data\training\*': "training-a", etc
training_fds = fileDatastore(fullfile(pwd, 'Data', 'training'), 'ReadFcn', @importAudioFile,
'FileExtensions', '.wav','IncludeSubfolders',true);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
data_dir = fullfile(pwd, 'Data', 'training');
folder_list = dir([data_dir filesep 'training*']);
reference_table = table();
for ifolder = 1:length(folder_list)
   disp(['Processing files from folder: ' folder_list(ifolder).name])
   current_folder = [data_dir filesep folder_list(ifolder).name];
   % Import ground truth labels (1, -1) from reference. 1 = Normal, -1 = Abnormal
   reference_table = [reference_table; importReferencefile([current_folder filesep
'REFERENCE.csv'])];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
runExtraction = false; % control whether to run feature extraction (will take several minutes)
  % Note: be sure to have the training data downloaded before executing
  % this section!
if runExtraction | ~exist('FeatureTable.mat')%#ok
   % Window length for feature extraction in seconds
   win_len = 5;
   % Specify the overlap between adjacent windows for feature extraction in percentage
   win_overlap = 0;
   % Initialize feature table to accumulate observations
   feature_table = table();
```

```
    % Use Parallel Computing Toobox to speed up feature extraction by distributing computation
across available processors
    % Create partitions of the fileDatastore object based on the number of processors
    n_parts = numpartitions(training_fds, gcp);
    % Note: You could distribute computation across available processors by using
    % parfor instead of "for" below, but you'll need to omit keeping track
    % of signal lengths
    parfor ipart = 1:n_parts
        % Get partition ipart of the datastore.
        subds = partition(training_fds, n_parts, ipart);
        % Extract features for the sub datastore
        [feature_win,sampleN] = extractFeatures(subds, win_len, win_overlap, reference_table);
        % and append that to the overall feature table we're building up
        feature_table = [feature_table; feature_win];
        % Display progress
        disp(['Part ' num2str(ipart) ' done.'])
    end
    save('FeatureTable', 'feature_table');
else % simply load the precomputed features
    load('FeatureTable.mat');
end
% Take a look at the feature table
disp(feature_table(1:5,:))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
classificationLearner
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tabulate classes in training data
grpstats_all = grpstats(feature_table, 'class', 'mean');
disp(grpstats_all(:,'GroupCount'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% using split function defined at end of script to divide feature table
% into training and test set, holding out 30%
[training_set, test_set] = splitDataSets(feature_table,0.3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Assign higher cost for misclassification of abnormal heart sounds
C = [0, 10; 1, 0];
% Create a random sub sample (to speed up training) of 1/4 of the training set
%subsample = randi([1 height(training_set)], round(height(training_set)/4), 1);
% OR train on the whole training set
subsample = 1:height(training_set);rng(1);
% Create a 5-fold cross-validation set from training data
cvp = cvpartition(length(subsample),'KFold',5);
% Step 2: train the model with hyperparameter tuning (unless you simply
% load an existing pre-trained model)
% train ensemble of decision trees (random forest)
```

```
disp("Training Ensemble classifier...")
% bayesian optimization parameters (stop after 15 iterations)
opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',cvp,...
         'AcquisitionFunctionName','expected-improvement-plus','MaxObjectiveEvaluations',15);
trained_model = fitcensemble(training_set(subsample,:),'class','Cost',C,...
      'OptimizeHyperparameters',{'Method','NumLearningCycles','LearnRate'},...
      'HyperparameterOptimizationOptions',opts)
% Step 3: evaluate accuracy on held-out test set
% Predict class labels for the validation set using trained model
% NOTE: if training ensemble without optimization, need to use trained_model.Trained{idx} to
predict
predicted_class = predict(trained_model, test_set);
conf_mat = confusionmat(test_set.class, predicted_class);
conf_mat_per = conf_mat*100./sum(conf_mat, 2);
% Visualize model performance in heatmap
labels = {'Abnormal', 'Normal'};
heatmap(labels, labels, conf_mat_per, 'Colormap', winter, 'ColorbarVisible','off');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
runNCA = true;   % control whether to see NCA running or just load the selected features
if ~runNCA && exist('SelectedFeatures.mat')%#ok
   % Load saved array of selected feature indexes
   load('SelectedFeatures.mat')
else % Perform feature selection with neighborhood component analysis
   rng(1);
   % first, let's make sure we've split the data (in case we skipped the
   % programmatic model training sections above)
   if ~exist('training_set')
      [training_set, test_set] = splitDataSets(feature_table,0.3);
   end
   mdl = fscnca(table2array(training_set(:,1:27)), ...
      table2array(training_set(:,28)), 'Lambda', 0.005, 'Verbose', 0);
   % Select features with weight above 1
   selected_feature_indx = find(mdl.FeatureWeights > 0.1);
   % Plot feature weights
   stem(mdl.FeatureWeights,'bo');
   % save for future reference
   save('SelectedFeatures', 'selected_feature_indx');
end
% Display list of selected features
disp(feature_table.Properties.VariableNames(selected_feature_indx))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trainReducedModel = false; % control whether to re-train this model or load it from a previous run
if trainReducedModel | ~exist('TrainedEnsembleModel_FeatSel.mat')%#ok
   % configure key parameters: cross validation, cost, and hyperparameter  % tuning
   rng(1)
```

```
   cvp = cvpartition(length(subsample),'KFold',5);
   C = [0, 10;  1, 0];          % Assign higher cost for misclassification of abnormal heart sounds
   opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',cvp,...
         'AcquisitionFunctionName','expected-improvement-plus','MaxObjectiveEvaluations',10);
   % now we are ready to train...
   trained_model_featsel =
fitcensemble(training_set(subsample,selected_feature_indx),training_set.class(subsample),'Cost',..
       'OptimizeHyperparameters',{'Method','NumLearningCycles','LearnRate'},
'HyperparameterOptimizationOptions',opts)
   % save the model for later reference
   save('TrainedWaveletModel_FeatSel', 'trained_model_featsel');
else
   load('TrainedEnsembleModel_FeatSel.mat')
end
% Predict class labels for the validation set using trained model
predicted_class_featsel = predict(trained_model_featsel, test_set(:,selected_feature_indx));
conf_mat_featsel = confusionmat(test_set.class, predicted_class_featsel);
conf_mat_per_featsel = conf_mat_featsel*100./sum(conf_mat_featsel, 2);
labels = {'Abnormal', 'Normal'};
% Visualize model performance
% probably custom AE version: heatmap(conf_mat_per_featsel, labels, labels, 1,'Colormap',
'red','ShowAllTicks',1,'UseLogColorMap',false,'Colorbar',true);
heatmap(labels,labels,conf_mat_per_featsel, 'Colormap', winter, 'ColorbarVisible','off');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
runCodegen = false;      % this typically takes several minutes
if runCodegen || ~exist('codegen/mex/classifyHeartSounds/classifyHeartSounds_mex')
   % Save trained model as a compact model for code generation
   saveLearnerForCoder(trained_model_featsel,'HeartSoundClassificationModel');
   % Alternatively, execute the following auto-generated MATLAB script to generate C code (and
mex file, in current directory)
  classifyHeartSounds_codegen_script
   % saving the MEX file for future referene
   %copyfile codegen/mex/classifyHeartSounds/classifyHeartSounds_mex.mexw64
HelperFunctions
End
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotPredictions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% apply wavelet scattering to the whole training set (used also above)
N = 10000;        % min number of samples across whole data set (5 secs signal x 2,000 Hz)
sf = waveletScattering('SignalLength',N,'SamplingFrequency',2000);% could try InvarianceScale
training_fds = fileDatastore(fullfile(pwd, 'Data', 'training'), 'ReadFcn', @importAudioFile,
'FileExtensions', '.wav','IncludeSubfolders',true);
data_dir = fullfile(pwd, 'Data', 'training');
folder_list = dir([data_dir filesep 'training*']);
```

```
reference_table = table();
for ifolder = 1:length(folder_list)
   disp(['Processing files from folder: ' folder_list(ifolder).name])
   current_folder = [data_dir filesep folder_list(ifolder).name];
   % Import ground truth labels (1, -1) from reference. 1 = Normal, -1 = Abnormal
   reference_table = [reference_table; importReferencefile([current_folder filesep
'REFERENCE.csv'])];
end
[train_data,trainN] = extractWaveletFeatures(training_fds,sf, 10000,reference_table);
```