

iscte

INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Digital platform for psychological assessment supported by sensors and efficiency algorithms

Francisco Mateus Valente de Matos Silva

Master's in Computer Engineering

Supervisor:

PhD. Pedro Joaquim Amaro Sebastião, Professor,
ISCTE-IUL

November, 2020

Resumo

A tecnologia está a evoluir, criando um impacto nas nossas vidas e na telessaúde. A telessaúde engloba a prestação de serviços e informações de saúde usando uma abordagem tecnológica. Vários estudos documentam os benefícios de métodos baseados na web para fornecer ajuda médica. No entanto, poucos estudos de saúde e ajuda psicológica combinam esse método com sensores vestíveis. Este estudo tem como objetivo criar uma plataforma online para utilizadores receberem ajuda e informações sobre autocuidado, usando sensores vestíveis. Além disso, investigadores a desenvolver um projeto semelhante têm uma base sólida como referência. Esta dissertação fornece descrições e análises da arquitetura do software e hardware. Exibe e explica um algoritmo eficiente e dinâmico de frequência cardíaca que calcula continuamente os valores desejados dos sensores. Apresenta diagramas que ilustram o processo de implementação online do site e os meios que o servidor da web utiliza para lidar com os dados dos sensores. O objetivo é criar um projeto funcional que usa hardware compatível com Arduino. Os sensores de frequência cardíaca enviam os seus valores para uma plataforma online. Uma placa de microcontrolador usa um algoritmo para calcular os valores de frequência cardíaca do sensor e envia-os para um servidor web. A plataforma visualiza os dados do sensor, resume-os num relatório e cria alertas para o utilizador. Os resultados mostraram uma estrutura sólida do projeto e da comunicação do hardware e software. O servidor da web exibe os dados do sensor de frequência cardíaca na plataforma online, apresentando observações e avaliações.

Palavras-chave: Telessaúde, Saúde baseada na Web, Sensores Vestíveis, Arduino, Placa de Microcontrolador, Frequência Cardíaca BPM.

Abstract

Technology is evolving, creating an impact on our everyday lives and Telehealth. Telehealth encapsulates the provision of healthcare services and information via a technological approach. Several studies document the benefits of using web-based methods to provide healthcare help. Nonetheless, few health and psychological help studies combine this method with wearable sensors. This study aims to create an online platform for users to receive self-care help and information using wearable sensors. In addition, researchers developing a similar project obtain a solid foundation as a reference. This dissertation provides descriptions and analyses of the software and hardware architecture. Exhibits and explains a heart rate dynamic and efficient algorithm that continuously calculates the desired sensors' values. Presents diagrams that illustrate the website deployment process and the web server means of handling the sensors' data. The goal is to create a working project using Arduino compatible hardware. Heart rate sensors send their data values to an online platform. A microcontroller board uses an algorithm to calculate the sensor heart rate values and outputs it to a web server. The platform visualizes the sensor's data, summarizes it in a report, and creates alerts for the user. Results showed a solid project structure and communication from the hardware and software. The web server displays the conveyed heart rate sensor's data on the online platform, presenting observations and evaluations.

Keywords: Telehealth, Web-Based Healthcare, Wearable Sensors, Arduino, Microcontroller Board, Heart Rate BPM.

Acknowledgments

First, I want to thank my mother, Maria da Conceição Mateus, and my father, Francisco José Silva. Thank you for all the support and help you gave me, both for my education and personal life. Because of you, I have all the opportunities and motivation to follow my goals and achieve them.

I also want to thank my brother, Nuno Mateus Silva. For always answering my questions and doubts about everything. You always provide advice and direction towards my goals.

I would also like to thank my professor and supervisor, Ph.D. Pedro Sebastião. Thank you for all the orientation and encouragement given during the dissertation development.

To all my close friends, Rui Matos, José Simões, Francisco Santos, Maria Coelho, Miguel Almeida, and Mariana Carvalho. Thank you for putting up with me and always keeping me company. To Jorge Santos (Mankey), I hope this made you proud.

Finally, I want to thank ISCTE-IUL for providing me this education and all the support and help I needed.

Contents

Resumo	iii
Abstract	v
Acknowledgments	vii
List of Figures	xiii
List of Tables	xvii
List of Acronyms	xix
Chapter 1 : Introduction	1
1.1: Motivation and Scope	1
1.2: Relevance and Objectives	1
1.3: Contributions	2
1.4: Thesis Structure Overview	3
Chapter 2 : State of the Art	5
2.1: Technology Impact on the Healthcare Industry	5
2.2: Telehealth	6
2.3: Hardware Used and Alternatives	9
2.3.1: Arduino	9
2.3.2: Heart Rate Sensors	10
2.3.3: Microcontroller Board and I/O Extension Shield	12
2.4: Software Used and Alternatives	14
2.4.1 Arduino IDE	14
2.4.2: Men Stack Type	15
2.4.3: Database	17
2.4.4: Domain Name	18
2.4.5: Heroku	18
2.5: Related Work	19

Chapter 3 : Hardware Architecture	23
3.1: Hardware Architecture	23
3.2: Heart Rate Sensors	24
3.2.1: PPG Heart Rate Technique	24
3.2.2: Gravity Heart Rate Sensor	26
3.2.3: VMA340 Heart Rate Sensor	27
3.3: Arduino Compatible Hardware	28
3.3.1: DFRduino UNO R3 Microcontroller Board	28
3.3.2: I/O Expansion Shield	30
3.3.3: Connecting to the Heart Rate Sensors	30
Chapter 4 : Software	33
4.1: Software Architecture	33
4.2: Arduino IDE	34
4.2.1: Serial Port (COM Port)	34
4.2.2: Heart Rate BPM Algorithm	35
4.3: Website Server	35
4.3.1: Web Server Connection to the Sensor	36
4.3.2: Visualizing the Sensor’s Data	38
4.3.3: Programming Languages, Essential Classes and Files	40
4.4: Database	42
4.4.1: Mongoose	42
4.4.2: MongoDB Atlas	43
4.5: Website Deployment	44
4.5.1: Online Server Hosting “Heroku”	44
4.5.2: DNS Connection to the Remote Server	46
Chapter 5 : Results	49
5.1: Tests	49

5.1.1: Connection Tests	49
5.1.2: Sensors Data Transmission Tests	51
5.1.3: Heart Rate Sensors Tests	52
5.1.4: Local and Remote Server Tests	53
5.2: Results Analysis	55
Chapter 6 : Conclusion and Future Work	57
6.1: Main Conclusion	57
6.2: Future Work	59
Appendix	61
Bibliography	67

List of Figures

Figure 1.1: Hardware and Software Overall Architecture	2
Figure 2.1: Telemedicine and Telehealth Hierarchy.....	6
Figure 2.2: PPG and ECG Raw Data Comparison	8
Figure 2.3: Arduino Microcontroller Boards (Left to Right: Lilypad, Arduino Mega, Arduino Pro Mini)	9
Figure 2.4: Raspberry Pi Single-Board Computer (Left) and Logo (Right)	10
Figure 2.5: Gravity ECG Sensor (Left) and Arduino PPG Pulse Sensor (Right).....	11
Figure 2.6: MEN Stack Type.....	15
Figure 2.7: Smartwatches Fitbit Versa 2 (Left), Apple Watch (Center) and Garmin Forerunner (Right)	21
Figure 2.8: VitalPatch Biosensor	21
Figure 3.1: Hardware Architecture	23
Figure 3.2: Hardware Components	24
Figure 3.3: Photoplethysmography (PPG) Technique	25
Figure 3.4: PPG Data Example.....	25
Figure 3.5: Gravity Heart Rate Sensor.....	26
Figure 3.6: Gravity Sensor Architecture	26
Figure 3.7: VMA340 Heart Rate Sensor.....	27
Figure 3.8: VMA340 Sensor Architecture	28
Figure 3.9: DFRduino UNO (v3.0) R3	29
Figure 3.10: Gravity I/O Expansion Shield V7.1 (Left) and Inserted on the DFRduino Board (Right)	30
Figure 3.11: Gravity Sensor Connection.....	31
Figure 3.12: Gravity Sensor Connection Schematic	31
Figure 3.13: VMA340 Sensor Connection.....	32
Figure 3.14: VMA340 Sensor Connection Schematic	32
Figure 4.1: Software Architecture.....	33
Figure 4.2: Serial Port Recognition and Arduino IDE Interface	35

Figure 4.3: Flowchart of the Web Server Handling the Sensor Data	36
Figure 4.4: Serialport Object Creation and Events	37
Figure 4.5: Sensor’s Data Visualization.....	38
Figure 4.6: Client-Side Socket.....	39
Figure 4.7: Sensor’s Heart Rate Data Visualization	40
Figure 4.8: EJS Example	41
Figure 4.9: “User” Schema.....	42
Figure 4.10: Mongoose Connection.....	43
Figure 4.11: Collection Containing a Document for a “Francisco” User	44
Figure 4.12: Flowchart of the Node.js Web Server Deployment	45
Figure 4.13: Heroku CLI Commands Used for the Deployment.....	46
Figure 4.14: Project Domain Names and their DNS Target.....	47
Figure 4.15: DATABASEURL Config Var	47
Figure 5.1: Microcontroller Board Recognition Test.....	50
Figure 5.2: Sketch Upload Test	50
Figure 5.3: VMA340 (Left) and Gravity (Right) LED Lights Test.....	50
Figure 5.4: Sensor's Raw Data Serial Plotter (Left) and Serial Monitor (Right) Tests.....	51
Figure 5.5: Sensor’s Calculated Data on the Serial Plotter (Left) and Serial Monitor (Right) Tests	51
Figure 5.6: Heart Rate BPM Person A Test	52
Figure 5.7: Heart Rate BPM Person B Test	52
Figure 5.8: Heart Rate BPM Person C Test	53
Figure 5.9: Sensor’s Data Transmission to Web Server Test.....	53
Figure 5.10: Sensor’s Data Visualization Test on the Local Server	54
Figure 5.11: User Register Form (Left) and its Designated MongoDB Document (Right)....	54
Figure 5.12: Psytechy.com and Psytechy.pt Domain Name Validation Tests	55
Figure 5.13: Remote Server “thesis test” Login Test.....	55
Figure A.1: Setup Function	61
Figure A.2: Example of Raw Data from a PPG Sensor	62
Figure A.3: Project sensor's raw data gathered from Arduino IDE	63
Figure A.4: Flowchart of the Loop Function Algorithm	63

Figure A.5: Heart Rate Sensor's BPM Algorithm Code65

List of Tables

Table 2.1: Telemedicine Tools and Services for Clinicians and Patients	7
Table 2.2: ECG Advantages and Disadvantages	8
Table 2.3: Raspberry Pi Advantages and Disadvantages	10
Table 2.4: Gravity and VMA340 Sensors Specifications	11
Table 2.5: Microcontroller Board Alternatives and their Specifications	13
Table 2.6: I/O Expansion Shield Alternatives and their Specifications	14
Table 2.7: MongoDB NoSQL Database and SQL Database Main Differences.....	17

List of Acronyms

AWS	Amazon Web Services
BPM	Beats Per Minute
BPS	Bits Per Second
BT	Bluetooth
CAD	Coronary Artery Disease
CHD	Coronary Heart Disease
CLI	Command Line Interface
CM	Centimeters
CSS	Cascading Style Sheets
CVD	Cardiovascular Disease(s)
DaaS	Database as a Service
DB	Database
DNR	Domain Name Registrar
DNS	Domain Name System
ECG	Electrocardiography
EJS	Embedded JavaScript
EKG	Electrocardiography
GIT	GitHub
GND	Ground
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input / Output
HR	Heart Rate

HRV	Heart Rate Variability
HTML	Hypertext Markup Language
I/O	Input / Output
I2C	Inter-Integrated Circuit
IaaS	Infrastructure as a Service
IAPMEI	Instituto de Apoio Às Pequenas e Médias Empresas e ao Investimento
ICAST	International Conference on Applied Science and Technology
ICSP	In Circuit Serial Programming
ICU	Intensive Care Unit
IDE	Integrated Development Environment
IP	Internet Protocol
IT	Information Technology
JS	JavaScript
JSON	JavaScript Object Notation
LED	Light-Emitting Diode
MDPI	Multidisciplinary Digital Publishing Institute
MM	Millimeters
NPM	Node Package Manager
ODM	Object Data Modeling
ORM	Object-Relational Mapping
OS	Operating System
PaaS	Platform as a Service
PC	Personal Computer
PCB	Printed Circuit Board

PHC	Primary Health Care
PHP	Hypertext Preprocessor
PPG	Photoplethysmography
PWM	Pulse Width Modulation
RAM	Random Access Memory
SaaS	Software as a Service
SD (card)	Secure Digital (card)
SPI	Serial Peripheral Interface
SQL	Structured Query Language
TLD	Top Level Domain
URL	Uniform Resource Locator
USB	Universal Serial Bus
VCC	Voltage Common Collector
WASET	World Academy of Science Engineering and Technology
Wi-Fi	Wireless Fidelity

Chapter 1 : Introduction

1.1: Motivation and Scope

Technology is growing fast, influencing the Telehealth service industry this dissertation covers. This new healthcare concept can provide monitoring and self-care help to users worldwide.

Many studies analyze and create prototype projects using technological approaches to Telehealth research. Results show a positive impact of these services in the healthcare industry. Although they provide many benefits and advantages, they do not replace the more traditional methods. Instead, both of them can help and support each other.

Research shows the benefits and advantages of Telehealth. However, there is a small gap in developing projects using wearable sensors, such as heart rate sensors, in a web-based context. Few studies on Telehealth combine wearable sensors' applicability to web-based methods' usefulness. The development of this dissertation's project aims to fill that gap.

1.2: Relevance and Objectives

The proposed research is essential and relevant to the future development of similar projects. Findings and results gathered through the hardware and software implementations act as guides and examples to help other works take conclusions from it.

Apart from helping similar projects, this dissertation's primary goal is to deliver a platform for users to receive health and psychological help using heart rate and heart rate variability data.

This dissertation's project architecture comprises two major phases, the hardware phase, and the software phase, demonstrated in Figure 1.1. The hardware phase establishes the microcontroller board and sensors' connection and communication. The software phase uploads the heart rate BPM algorithm sketch to the board, and it outputs the sensors' data to the web server, which then handles it.

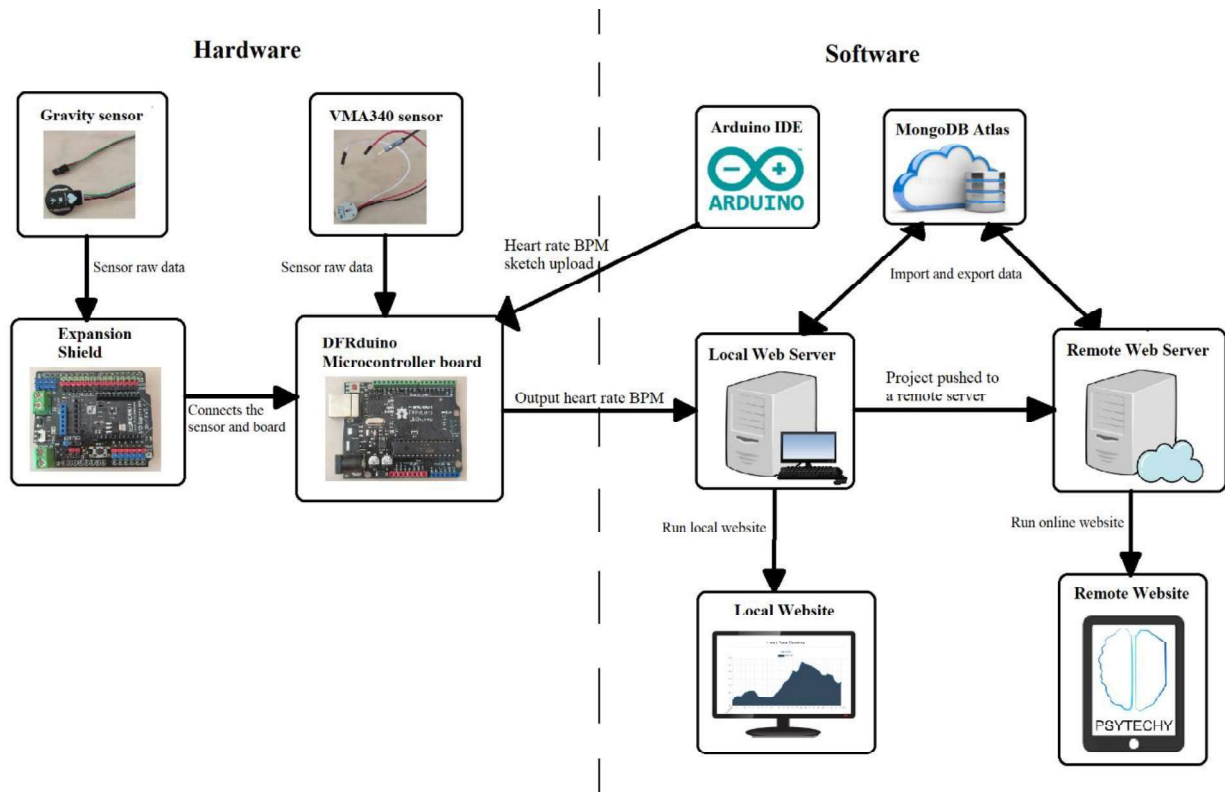


Figure 1.1: Hardware and Software Overall Architecture

This project aims to use wearable sensors to process and send data to a web server, which then visualizes that same data and delivers informational reports and alerts based on it. The wearable sensors used are heart rate sensors based on the PPG technique. The sensors send their raw data to a microcontroller board, whose duty is to calculate the heart rate BPM data and output it to the web server. The web server hosts the online website and updates the sensor calculated data on a webpage. The website then handles that sensor's data to provide informational reports and alerts for the user.

The final goal is to synchronize all the hardware and software components and have working communications without issues and errors. Accomplishing this will result in a solid foundation and development of the project. This foundation will serve as a basis for experimenting with new sensors and other hardware, extra website features, and an excellent guide to create a similar project and improve upon it.

1.3: Contributions

This dissertation served as a basis of integration for two financed projects, “*Psytechy*” and “*WAYLA*”. Both projects apply and transfer the technologies and methodologies used for this

dissertation. IAPMEI funded the projects' creation and development. The dissertation was submitted to a journal of applied natural sciences and an international conference on applied science and technology (ICAST).

- *Psytechy* – This project is the online platform on which this dissertation bases on. I developed the IT infrastructure and was responsible for implementing the necessary technology. R. Sebastião coordinated and led the project and contributed with psychology knowledge to apply on the platform. B. Ribeiro marketed and managed the project.
- *WAYLA* – This project implements wearable sensors and analysis technologies to create reports that help marketing companies understand user experience on their products and services. R. Matos and I developed and built the project.
- *Multidisciplinary Digital Publishing Institute (MDPI) Applied Sciences* journal on all aspects of applied natural sciences published semi-monthly online [1].
- *International Conference on Applied Science and Technology (ICAST)*. Hosted by the World Academy of Science Engineering and Technology (WASET) [2].

1.4: Thesis Structure Overview

Following this introduction, chapter 2 addresses the literature review. This chapter has descriptions of essential themes and topics in the dissertation, such as Telehealth, wearable sensors, hardware components, and software frameworks. The chapter explores possible alternatives and defines most of the project's requirements.

Chapter 3 handles the hardware architecture and development phase of the project. The chapter then addresses the hardware used and the connections and communications between them. Finishing the hardware phase without errors and issues enables the next development phase.

Chapter 4 covers the software phase developed after the hardware implementations. This chapter describes the sketch program's algorithm responsible for calculating the heart rate BPM data. Receiving that data on the web server and handling it on the website. Most of the dissertation objectives result from this chapter.

Chapter 5 shares the testing done for the most critical aspects of the project and that contribute to the dissertation's goals. The chapter analyzes the testing results, improvements and interprets the best results from the tests.

Finally, chapter 6 handles the dissertation conclusion, whether the project delivered on all the main objectives. The chapter also describes potential future work on some of the project's aspects that need progression.

Chapter 2 : State of the Art

This review first focuses on Telehealth and the various telehealth methods used, in combination with some examples. The chapter then explores heart rate sensors and how heart rate can determine and contribute to someone's health assessment. Finally, the chapter describes the Hardware and Software components used for this dissertation's project, their alternatives, and related work observations.

2.1: Technology Impact on the Healthcare Industry

Since the initial growth of information technology (IT), multiple studies have analyzed its impact in various industries, such as healthcare. Studies, such as [3]–[5], determined a positive outcome in IT investment in the healthcare industry, based on performance, efficiency, quality, reducing response times, and other aspects.

The article [6] investigated five medical conditions to determine if medical advances' technological benefits exceed the costs. The study estimates four conditions to have much more effective use of technological change than the price; those conditions were heart attacks, low-birthweight infants, depression, and cataracts. The fifth condition studied was breast cancer, which resulted in an equal cost of benefits and costs.

This dissertation focuses on IT applications and methods for psychological illnesses (depression) and cardiovascular diseases (CVD) (coronary artery disease (CAD)), which are a safe investment and an area that will improve from using IT.

Heart rate [7] has always been a popular topic, and with the rise of technology, new methods and applications are finding ways of handling it. Heart rate is the number of times a heart beats per minute (BPM). Usually, a regular resting heart rate is between 60 and 100 BPM, although this reading can depend on many variables.

Heart rate variability (HRV) [8] is the variation of beat-to-beat intervals. A normal heart has a high HRV, while diminished HRV may pronounce cardiac disease. HRV also decreases with exercise-induced tachycardia. HRV is a well-investigated area, and there have been many studies done exploring its usability, methods, and results [9]–[11]. For example, one aspect of heart rate variability is to measure fitness, the speed at which one's heart rate drops upon the termination of vigorous exercise [12]. The pace at which a person's heart rate normalizes is

faster for a more athletic person. Descending twenty beats in a minute is common for a healthy individual.

Several studies have found psychological illnesses and CVDs that correlate with heart rate and HRV. During psychological stress, there is a significant reduction in the timing and frequency of HRV and a massive increase in heart rate [13]. The article [14] brings up the links between stress, anxiety, depression, and coronary heart disease (CHD) with heart rate and HRV. The standing HRV is lower in panic disorder patients [15]. Further linking the relationship between depression and CVD, depressed patients tend to have higher heart rates and lower HRV, and also both higher heart rates and lower HRV increase CVD risk [16], [17].

2.2: Telehealth

This dissertation focuses on Telehealth applications and methods.

Telehealth's definition is the use of medical information exchanged from one site to another through electronic communication to improve a patient's health. Telehealth administers health care remotely through several telecommunication tools, including telephones, smartphones, web services, and mobile wireless devices. Telehealth applications are increasing, and it can transform the delivery of health care for millions of persons.

Another area that is usually mentioned alongside Telehealth is Telemedicine [18]. Telehealth encapsulates all Telemedicine services and tools (Figure 2.1), while the other does not. In other words, Telemedicine is a subdivision of Telehealth. Telemedicine is concerned with remote clinical services, while Telehealth is involved with remote and non-remote clinical services.

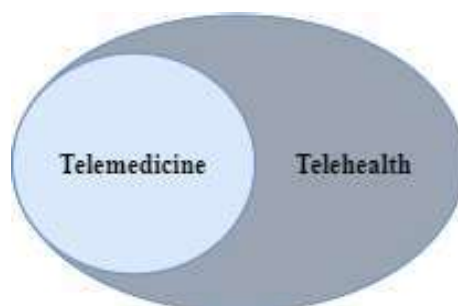


Figure 2.1: Telemedicine and Telehealth Hierarchy

Table 2.1 describes some of the tools and services provided by Telemedicine.

Table 2.1: Telemedicine Tools and Services for Clinicians and Patients

	Telemedicine Tools	Telemedicine Services
Clinician to Clinician	Communication through video and email	Radiology, Dermatology, ICU care, Emergency Trauma, and Surgical peer mentoring
Clinician to Patient	Video, Phone, Email, Internet, Remote wireless monitoring	Wound care, Counseling, Care for chronic conditions, Medication management, Mental health, Post-discharge follow-up
Patient to Mobile Health Technology	Video, Email, Smartphones, Wearable monitors, Mobile apps, Web portals, Games	Monitoring of diet, Monitoring of physical activity, Health education, Cognitive fitness, Medication adherence

According to [19], three linked trends are currently shaping Telehealth. The first is presenting convenience and ultimately reducing cost. The second is the augmentation of Telehealth to also addressing episodic and chronic conditions. The third is the migration of Telehealth to the home and mobile devices.

In the healthcare domain, Telehealth enables the use of sensors within or on the human body, known as wearable sensors [20], [21]. Wearable sensors such as ECG, PPG, and activity monitors are specific types of medical sensors placed on the human body allowing non-invasive, unobtrusive, 24/7 data collection for health monitoring. The increasing number of ambient devices installed in homes surrounding the human body provide telehealth interventions to citizens seeking affordable healthcare while remaining in touch with medical practitioners remotely. Some wearable sensors are clothing embedded sensors, headbands, skin patches, wrist-worn, cuffs, and finger-worn. Article [22] describes these wearable sensors.

This project uses two heart rate sensors. Both use Photoplethysmography (PPG) [23] in order to obtain the heart rate data desired. Section 3.1.2 explains the PPG technique/method used by the sensors to gather heart rate data.

Another alternative popular to PPG is Electrocardiography (ECG or EKG) [24]. The technology involving ECG has continuously advanced throughout the years. Nowadays, experts generally record an ECG by placing electrodes on a patient's chest or another suitable

area that checks the heart's rhythm and electrical activity. Wires connect the electrodes to a device that receives and handles that data, which in this project's case would be an ECG sensor connected to the microcontroller board. Each time the heart beats, the device records an electrical impulse as a "wave", which it then uses to calculate the patient's appropriate heart rate.

Table 2.2 lists the main advantages and disadvantages [25] of using an ECG compared to a PPG.

Table 2.2: ECG Advantages and Disadvantages

<i>ECG Advantages</i>	<i>ECG Disadvantages</i>
<ul style="list-style-type: none"> - More accurate HR - More accurate HRV - Better power consumption - A better understanding of overall heart performance 	<ul style="list-style-type: none"> - Less comfortable - Not practical in most situations - More complex to use

A PPG sensor is used instead of an ECG one because a more comfortable and easier to apply sensor makes more sense in the project's context. Figure 2.2 shows the difference between the PPG and ECG heart rates, where the "PP" and "RR" variables represent the time intervals between heart beats.

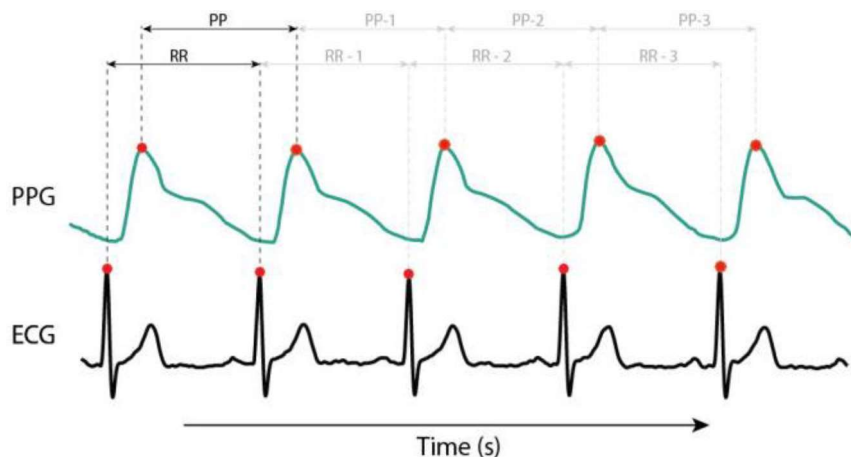


Figure 2.2: PPG and ECG Raw Data Comparison

2.3: Hardware Used and Alternatives

The hardware used for this dissertation's project consists of Gravity and VMA340 Arduino compatible heart rate sensors, Arduino compatible DFRduino UNO (v3.0) R3 microcontroller board, I/O expansion shield compatible with the DFRduino board, a USB type B cable, and finally a personal computer. Chapter 3 further explains all the hardware.

2.3.1: Arduino

Arduino is a company that distributes open-source hardware and software. Arduino uses development boards, or microcontroller boards (Figure 2.3), as prototyping platforms that process inputs and outputs with other boards and other components that interact with it, such as sensors, LEDs, and displays.



Figure 2.3: Arduino Microcontroller Boards (Left to Right: Lilypad, Arduino Mega, Arduino Pro Mini)

When working with Arduino hardware, the most common approach is to use the Arduino IDE software to upload sketches (programs) to operate on the Arduino microcontroller board. The open-source concept helps to share community knowledge and contributes to its overall growth and expansion. It also benefits people beginning to experiment with the Arduino platform and developing new projects. Arduino is also the most popular open-source electronics platform, contributing to its community growth and availability of online help and tutorial projects. One of the most popular alternatives to Arduino is the Raspberry Pi (Figure 2.4).

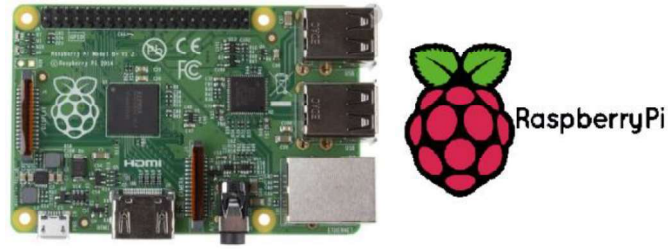


Figure 2.4: Raspberry Pi Single-Board Computer (Left) and Logo (Right)

Raspberry Pi are affordable single-board computers that run the Raspberry Pi operating system (OS) and provide a set of general-purpose input/output (GPIO) pins that allow controlling electronic components for physical computing [26]. Table 2.3 describes the main advantages and disadvantages [27] of using a Raspberry Pi when compared to an Arduino.

Table 2.3: Raspberry Pi Advantages and Disadvantages



<i>Raspberry Pi Advantages</i>	<i>Raspberry Pi Disadvantages</i>
<ul style="list-style-type: none"> - Has its own Operating System - Can run multiple programs - Appropriate for bigger and complicated projects - Suitable for experts on the subject 	<ul style="list-style-type: none"> - More difficult to install libraries - Harder to interface with other components - More complex and problematic to use - Challenging for beginners to pick up

This dissertation's project uses the Arduino platform instead of the Raspberry Pi. While Raspberry Pi is considered better for handling multiple tasks and more complex projects, the Arduino is considered a better alternative for repetitive tasks such as reading heart rate sensor data. Its open-source community and immense popularity are a great benefit, enabling a more painless experience for new projects.

2.3.2: Heart Rate Sensors

The heart rate sensors used for this project, Gravity [28] and VMA340 [29], are based on the PPG technique and are fully compatible with the rest of the Arduino hardware. Both sensors need to be operational with all the remaining hardware and software used; otherwise, the sensors' data would not forward to the computer. Table 2.4 lists the two sensors specifications.

Table 2.4: Gravity and VMA340 Sensors Specifications

Sensor Name	Operating Voltage	Operating Current	HR Method	Dimension and Cable Length	Connections	Output type
	3.3 to 6V	<10mA	PPG	28x24mm, 29.5cm	GND, VCC, Signal (compacted)	Analog/ Digital
	3 to 5V	4mA at 5V	PPG	16x18mm, 18cm	GND, VCC, Signal (separated)	Analog

The heart rate sensor's alternatives usually do not vary much, and most output similar results. The main differences are in the sensor's compatibilities and methods to retrieve the heart rate. The Arduino pulse sensor [30], shown in Figure 2.5, is a possible alternative that would work in this project and is developed by Arduino. The project does not implement this sensor because the Gravity one is more comfortable. The same company (DFRobot) developed the microcontroller board, expansion shield, and sensor, and all of them are compatible. The project implements the VMA340 sensor because it was easily obtained, and its results are compared with the Gravity sensor. Another possible alternative that uses a different method to receive the heart rate readings is the Gravity ECG sensor [31] (Figure 2.5), which uses the ECG technique to read heart rate readings instead of the PPG technique used in this project.



Figure 2.5: Gravity ECG Sensor (Left) and Arduino PPG Pulse Sensor (Right)


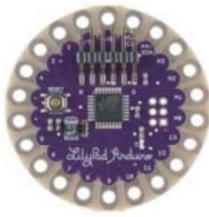


2.3.3: Microcontroller Board and I/O Extension Shield

This project uses the microcontroller board DFRduino UNO (v3.0) R3 [32]. This board is fully compatible with all the hardware and software that concern the project. The board is more comfortable to implement and less complicated than some of its alternatives.

A single-board microcontroller [33], [34] is fundamentally a microcontroller built onto a single printed circuit board. This microcontroller board comes with all the necessities needed to process information and output a specific programmed task. The board has a microprocessor, RAM, integrated circuits, and more. The board has digital and analog inputs and outputs (I/O) pins used to connect to sensors, batteries, modules, and other hardware to establish connections. These I/O pins combine with breadboards and extension shields to extend the number of possible outcomes and connections to other hardware. The board also implements a USB interface. It uses the USB interface to upload programs from a personal computer and send the desired heart rate data output to the personal computer.

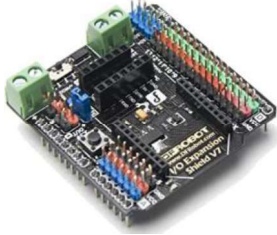
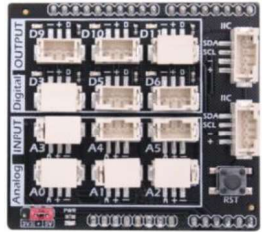

Table 2.5 shows some of the alternatives that could be used and would make a good option for this project. Arduino Lilypad [35], which is a wearable microcontroller board. Arduino Nano [36], which is a smaller but complete microcontroller board. Finally, Arduino UNO Wifi Rev2 [37], which enables Bluetooth (BT) and Wi-fi communications. The project implements the DFRduino because there are many projects and overall knowledge available online, and the alternatives can be implemented in the future.

Table 2.5: Microcontroller Board Alternatives and their Specifications

Board Name	Microcontroller	Operating Voltage	Input Voltage	Clock Speed	Analog ports	Digital ports	Dimension
 DFRduino Uno R3	ATmega328	5V	7 – 12V	16MHz	6	14 (6 provide PWM)	75x54x15mm
 Lilypad Arduino	ATmega168 or ATmega328V	2.7 – 5.5V	2.7 – 5.5V	8MHz	6	14 (6 provide PWM)	50mm outer diameter, Thin 0.8mm PCB
 Arduino Nano	ATmega328	5V	7 – 12V	16MHz	8	22 (6 are PWM)	18x45mm
 Arduino Uno Wifi Rev2	ATmega4809	5V	7 – 12V	16MHz	6	14 (5 provide PWM)	68.6x53.4mm

Like DFRduino UNO R3, microcontroller boards can use printed circuit expansion boards called I/O expansion shields, which plug into the usually supplied microcontroller board pin headers. Table 2.6 shows three expansion shield examples and their specifications. Expansion shields can provide many new applications and possibilities to the board, such as a Global Navigation Satellite System (GNSS), Ethernet connection, breadboarding and more. In this dissertation’s project case, the Gravity I/O extension shield [38] establishes the connection between the DFRduino microcontroller board and the Gravity heart rate sensor.

Table 2.6: I/O Expansion Shield Alternatives and their Specifications

<i>Expansion Shield Name</i>	<i>Operating Voltage</i>	<i>Dimensions</i>	<i>Features</i>
 <p>Gravity</p>	3.3 or 5V	55x53mm	Supports I2C, SPI, Xbee (pro), BT, APC220, and SD card read/write.
 <p>Cookie</p>	3.3 or 5V	58x52mm	Implements a fail-safe system designed for beginners and classrooms.
 <p>Arduino Nano</p>	3.3V	56x54mm	Designed explicitly for Arduino Nano.

2.4: Software Used and Alternatives

This dissertation's project implements various software. The most relevant ones are Arduino IDE, Node.js, Express framework, MongoDB (database), Mongoose, MongoDB Atlas, the domain name registrars GoDaddy and Dominios.pt, and the PaaS Heroku used to deploy the website online. This section describes the software used and presents some possible alternatives for them. Later, chapter 4 explains in more detail the software implemented.

2.4.1 Arduino IDE

The Arduino Integrated Development Environment (IDE) [39] is the main text editing program used for Arduino programming. Essentially, it translates and compiles sketches into code that Arduino compatible hardware can understand. Arduino IDE enables the development of code

in C/C++, compiles it, and uploads a sketch program to the board's memory. The Arduino IDE also allows access to an enormous Arduino library continuously growing thanks to an open-source community.

When writing code in the Arduino IDE software, the program that runs that code is called a "sketch". This sketch is the code that is uploaded to and runs on a microcontroller board compatible with Arduino hardware. Many sketch examples [40] provided by Arduino or the open-source community are ready to be uploaded to a board for all the different types of sensors and possibilities available.

Some alternatives to apply instead of Arduino IDE are PlatformIO [41], Visual Studio Code [42] and Eclipse Arduino IDE [43]. All of these would work with the Arduino hardware and obtain similar results. The only difference would be the interface where the code is written and the setup implementing all the hardware connections to the platform. The project implements the Arduino IDE because the setup is much easier to implement, and there is an abundance of online projects and examples to serve as guides.

2.4.2: Men Stack Type

A "Stack" type [44] is known as tools on which a web server or a web project build on. Other stack type examples are LAMP, MEAN, MEN, MERN, Django, which all are acronyms for the tools that the specific stack uses. All these stacks include in them a database, backend framework, server, and a frontend framework.

The web server developed for the dissertation is of the stack type "MEN" (Figure 2.6), which stands for MongoDB, Express, and Node.js. Instead of having a frontend framework such as Angular or React (MEAN or MERN), the web server uses vanilla JavaScript.



Figure 2.6: MEN Stack Type

- MongoDB [45] is the database service used. MongoDB is a NoSQL database that utilizes JSON-like documents with a schema. MongoDB is commonly used in connection with Node.js because of its JSON-like documents for interacting with data instead of the row/column model. Another benefit is the well-supported Node library Mongoose [46], which is an easy way of interacting with Mongo through Node, and that this Node.js project uses. Another service used and explained further ahead is the SaaS application MongoDB Atlas [47], formerly known as mLab and also known as a Database as a service (DaaS). MongoDB Atlas, simply put, offers a way to host MongoDB databases on the cloud.
- Express [48] is the backend framework used. Express is generally the standard server framework for Node.js. The Express backend framework sits on the server and provides a structure for building the app's functionality. It is also responsible for loading the pages and what they can do behind the scenes, handling the sensor data, connecting to the database, handling the users, and more.
- Node.js [49] is the software that serves the webserver. Node.js is the application runtime that the MEN stack runs on, working as a server it accepts requests from users and sends content to the browser. Node.js is mostly known because it permits JavaScript, which traditionally uses a frontend language, for backend web development. Thus, it allows for the unifying web-application development around a single programming language (JavaScript) instead of using different languages for server and client-side scripts.

Two of the most commonly used stacks [50], other than the MEN and MEAN/MERN stack types, are LAMP and Django, which would have been good alternatives to the project.

Lamp (Linux, Apache, MySQL, PHP) [51] is trendy because it is open-source, easy to modify, and has been around since 2000. Its disadvantages come from having programming languages challenging to learn and the necessity to understand Linux and Apache servers.

Django [52] is a web application framework used with python. Python is one of the most popular programming languages, so there is considerable support for it. When using Django, it comes in a complete package that can slow down a project or restrain its potential growth.

2.4.3: Database

A simple definition of a database is an organized collection of information or data easily accessed, managed, and updated electronically from a computer system.

MongoDB [45] is a cross-platform document-oriented database program, as explained above, and is the database used for this project.

Table 2.7 lists some of the main differences [53] between a MongoDB NoSQL database and a SQL database.

Table 2.7: MongoDB NoSQL Database and SQL Database Main Differences

<i>MongoDB NoSQL</i>	<i>SQL</i>
Collections containing multiple JSON documents	Tables with fixed rows and columns
Fields or Attributes	Columns
Documents (combines data in a single document)	Rows of data (can reference data in other tables)
Flexible Schemas	Rigid Schemas
Models (take a Schema and create an instance of a document)	Records
Horizontal Scaling (scale-out across servers)	Vertical Scaling (scale-up with a larger server)
Does not require Object-relational mapping (documents map directly to data structures)	Requires Object-relational mapping (ORM)
Commonly requires Joins	Rarely does not require Joins
Developed in the 2000s	Developed in the 1970s

Mongoose [46] is an Object Data Modeling (ODM) library for MongoDB and Node.js. Mongoose provides a straight-forward, schema-based solution to model the application data. In terms of Node.js, MongoDB is the native driver for interacting with a MongoDB instance database, and mongoose is an Object modeling tool for MongoDB.

MongoDB Atlas [47], formerly known as mLab (changed its name due to the acquisition by MongoDB), is a fully-managed cloud MongoDB database. It handles all the complexity of deploying, managing, and monitoring a cloud service provider (IaaS) partner, such as AWS, Azure, and Google Cloud. Mongo Atlas is also described as a Database-as-a-service (Daas) for MongoDB, and it is also a partner with the Platform-as-a-service (Paas) Heroku, which deploys the Node.js project to the Internet.

2.4.4: Domain Name

A domain [54] is the name used to identify a website. A domain name is an address where Internet users can access a website, rather than using IP addresses, which would be difficult for people to remember.

Translating an IP address to a domain name is accomplished through the Domain Name System (DNS) [55]. For example, it is possible to replace the IP address 207.97.195.109 with a domain name such as `www.fishingbass.com`, making it much easier to remember the domain name than the IP address. Another example of a domain name used in URLs to identify Web pages is the URL: `https://www.youtube.com/watch?v=dQw4w9WgXcQ`, where the domain name is `youtube.com`.

DNS [55] stands for Domain Name System, and its primary function is to translate domain names into IP Addresses, which computers can understand and so that browsers can load Internet resources. Each device connected to the Internet has a unique IP address, which other machines use to find the device. A common analogy used when explaining DNS is that it serves as the "internet phone-book".

Domain names use different Top-Level Domains (TLDs) [56], for example, `google.com`, `facebook.com`, `dgs.pt`. The domain name must be registered before it can be used, and every domain name is unique, which means no two websites can have the same domain name.

Some platforms provide the rights to buy a domain name, known as domain name registrars (DNR) [57]. The DNRs used in this project are GoDaddy [58] and `Dominios.pt` [59], but other DNR's provide the same services and domains at different costs, such as `domain.com`, Bluehost, HostGator, and more. As explained before, domain names need to be unique, so only the unused ones are available.

2.4.5: Heroku

Heroku [60] is a container-based cloud PaaS used as a provider of online server hosting for web projects based on Amazon Web Services (AWS), and it was the one used for this dissertation's project.

It is one of the first cloud platforms to be developed, and it now supports several programming languages: Java, Node.js, Scala, Python, PHP, Clojure, Ruby, and Go. Thus, it

has features for a developer to run, scale, build and manage applications across different languages and still perform the same actions or outputs independent of the programming language used to compile or interpret it.

There are many alternatives [61] to use instead of Heroku, and they are all great and with unique benefits. Back4app [62] is a popular parse hosting and serverless database platform. Elastic Beanstalk (AWS) [63] is a DevOps tool used to deploy, scale web applications and services and is supported by multiple languages. Google App Engine [64] is a PaaS hosted and owned by google that also scales web applications and supports multiple languages. The project uses Heroku mainly because there are many benefits when combined with Node.js.

2.5: Related Work

The following articles and projects demonstrate various examples where telehealth methods are applied and their results. Web-based methods are good examples of the various applications where the health industry can grow and improve with technology, such as internet self-report questionnaires. Some examples of wearable sensors are then described, such as heart rate and accelerometers sensors providing data that improve the user's health. Finally, smartphones and smartwatches are devices from our everyday lives that are almost guaranteed to have Telehealth applications.

Article [65] evaluated and compared traditional paper-and-pencil methods with internet data collection methods, specifically self-report questionnaires from self-selected samples. A set of 510 published traditional samples and new large internet samples (N=361,703) are compared to determine the internet samples' data quality. The internet samples show more diversity regarding gender, socioeconomic status, geographic region, and age. Results show that internet methods have their advantages and contribute to various psychology and Telehealth areas.

The study [66] tests the efficacy of a web-based approach to providing a cognitive-behavioral intervention for infertile women seeking reproductive medical technologies. Two groups compare general and infertility-related psychological stress measures. One of the groups used the web-based intervention, and the other used a wait-list control condition. Results were mixed when it came to intervention efficacy but showed significant declines in general stress for the web-based intervention group compared to the wait-list control group.

The study [67] investigated whether web-based assessment techniques were more efficient than traditional paper-based methods for Test-retest reliabilities of alcohol measures. Three conditions are randomly assigned to 255 participants, paper-based, web-based, and web-based with interruption. Results showed that web-based methods are a suitable alternative to the traditional paper-based methods with the added benefit of being cost-efficient, minimizing data collection, and increasing survey accessibility.

Article [68] develops an android application to compare HR readings from a Motorola Android phone to an ECG and Nonin 9560BT pulse oximeter [69]. The android application uses the PPG method through the smartphone's video camera. All the devices tested showed high correlation and consequentially validated the android smartphone as a valid measure for HR readings, with a 95% accuracy to the ECG.

The article [70] contributes to Telehealth by testing the usability of an ECG sensor and two Accelerometers to assist the elderly population. The ECG sensor is strapped to the user's chest and sends the HR readings through Bluetooth to a personal computer. The Accelerometers recognize the user's activities and behaviors, detecting any potential falls and sending the information through Bluetooth to the same computer.

Paper [71] demonstrates a tablet PC's development and efficiency that enables a non-invasive body sensor system for rural Telehealth applications. The system continuously collects various readings. A body sensor unit gathers all these readings and transmits the information to a Primary Health Center (PHC).

Chest straps with heart rate monitors and smartwatches with health applications are popular wearable sensors that keep growing and expanding. Some companies have been able to develop advanced ones that create a meaningful impact on our society.

Various smartwatches now provide the function of reading heart rates and other health measures. Many of them have grown popular in the healthcare and sports industry and significantly impacted both. These smartwatches usually use the PPG method of gathering HR and HRV values. Figure 2.7 shows some of the most popular smartwatches whose brands are Fitbit [72], Apple Watch [73] and Garmin (smartwatch) [74].



Figure 2.7: Smartwatches Fitbit Versa 2 (Left), Apple Watch (Center) and Garmin Forerunner (Right)

Chest straps with heart rate monitors, as the name implies, are worn just under the chest in a comfortable way that enables the appropriate gathering of the monitor's data. These wearable sensors usually employ a different method of reading HR and HRV values. This method [75] consists of transmitting a radio signal from a monitor on the chest strap whenever a heartbeat is detected. After, a receiver handles the transmitted data and displays the current HR. Some chest strap examples are Garmin (chest strap) [76], Polar [77], and Wahoo [78].

Another impressive wearable sensor used in the healthcare industry is VitalPatch (Figure 2.8) [79]. VitalPatch is a biosensor with an ECG and capable of measuring HR, HRV, respiratory rate, body or skin temperature, body posture, fall detection, and activity motion.



Figure 2.8: VitalPatch Biosensor

Chapter 3 : Hardware Architecture

3.1: Hardware Architecture

The hardware phase requires an operational and working connection between all the hardware components used. This connection is crucial to obtain the best results and the projected functionality of the developed software derived from the sensor's data. Figure 3.1 details the hardware architecture.

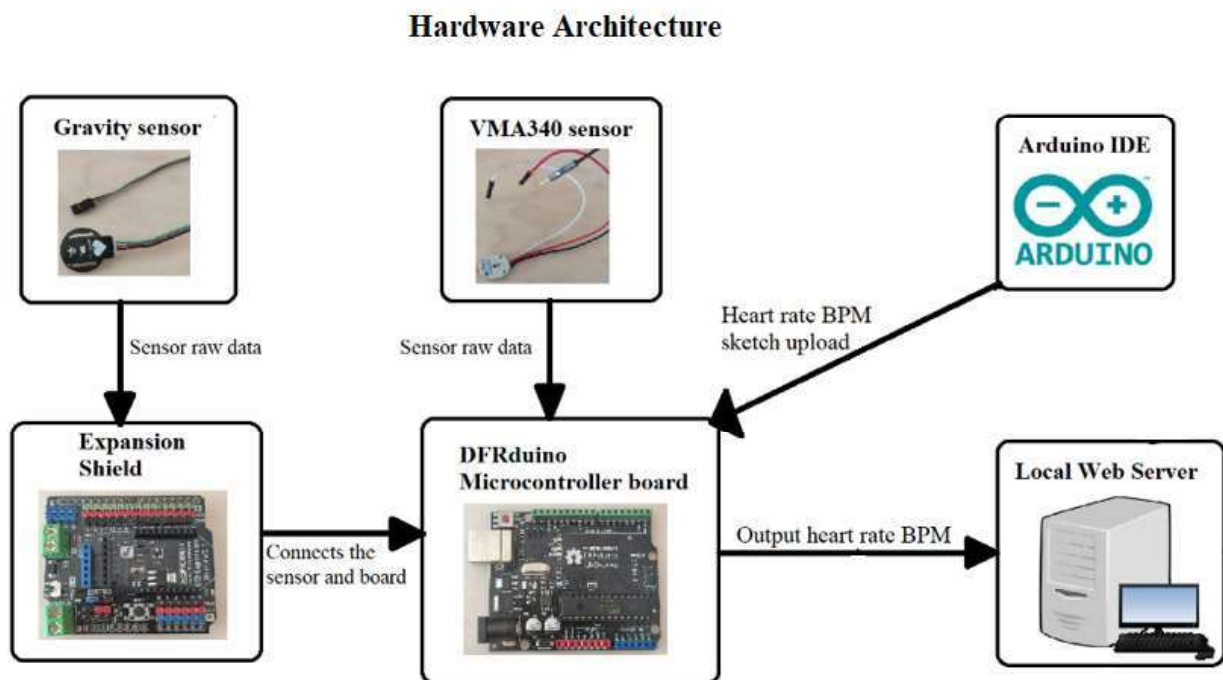


Figure 3.1: Hardware Architecture

Fundamentally, an Arduino compatible heart rate sensor will receive data input concerning the frequency of a heartbeat. That sensor will connect to an Arduino compatible microcontroller board and an expansion shield if necessary. The microcontroller board will calculate the desired heart rate BPM using an uploaded sketch (program) and output the result to the computer through a USB cable.

The hardware components used and explained to a greater extent in this chapter are shown in Figure 3.2. The Gravity (1) and VMA340 (2) Arduino compatible heart rate sensors (with the cables), the Arduino compatible DFRduino UNO (v3.0) R3 microcontroller board (3), the I/O expansion shield (4), a type B USB cable (5) and finally a personal computer (6).

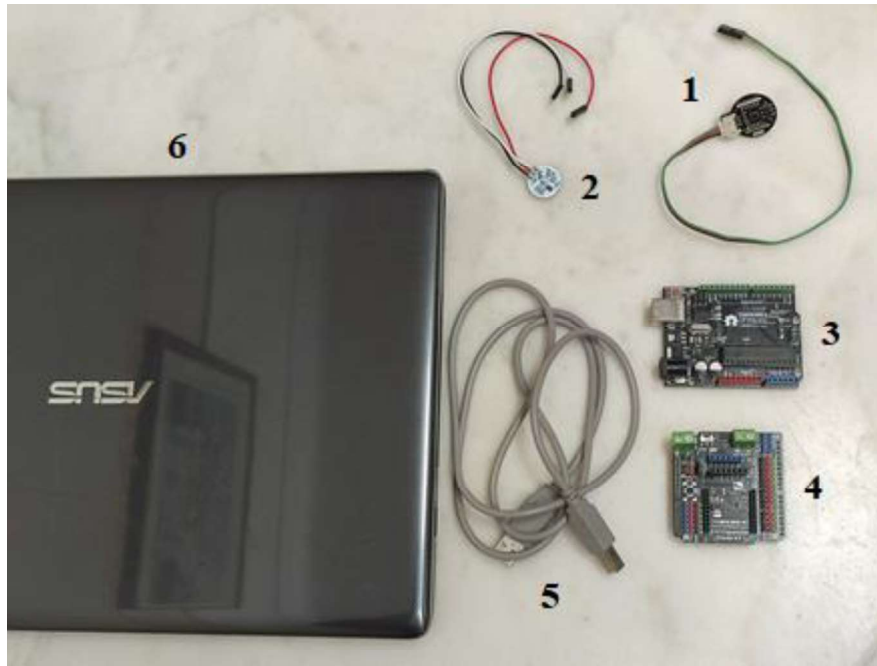


Figure 3.2: Hardware Components

The software phase works with the calculated heart rate BPM and then stores it in a database and visualizes it on the created website. This phase involves developing the sketch program to upload to the microcontroller board, which is responsible for handling the sensor's raw data and calculating the heart rate BPM to output. A website is built and deployed online using a remote server hosting provider and a domain name registrar. Finally, the project uses a cloud-based MongoDB database to handle both the remote and local server data. Chapter 4 further explores the software phase.

3.2: Heart Rate Sensors

This section explains the PPG heart rate reading method that the sensors use. After this, a brief description of the Gravity and VMA340 heart rate sensors is provided.

3.2.1: PPG Heart Rate Technique

This dissertation's project employs two heart rate sensors. Later, the study compares the data provided by both and explores the results. The heart rate sensors operate using the same technique, Photoplethysmography (PPG).

Photoplethysmography [23], mostly known as PPG, uses infrared light to measure blood pressure variations. This technique works because the infrared light emanates a light

signal in an area where arteries are visible, and whenever there is a heartbeat, the light signal reflects, and blood flow is measured. The most visible arteries to the infrared light are typically a fingertip (except the thumb), an earlobe, or wrists. Figure 3.3 demonstrates how the technique works.

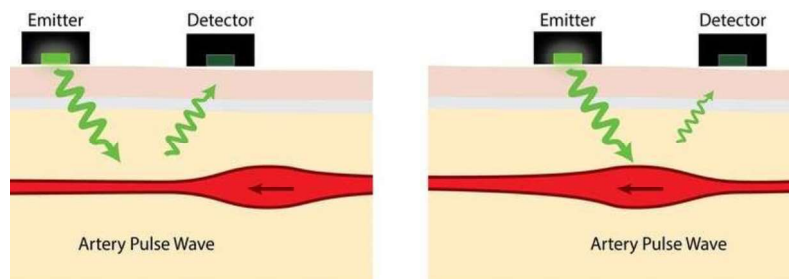


Figure 3.3: Photoplethysmography (PPG) Technique

Using this measurement technique, which corresponds to the cardiac rhythm in most cases, access to the heart rate data from the PPG signal is possible.

The variation in volume instigated by the pressure pulse is spotted by illuminating the skin with the infrared light and then determining the amount of light either conveyed or reflected to a photodiode (semiconductor that converts light into an electrical current). Each cardiac cycle appears as a peak, and through these peaks, it is possible to estimate the heart rate. Figure 3.4 demonstrates an example of data gathered from the PPG technique.

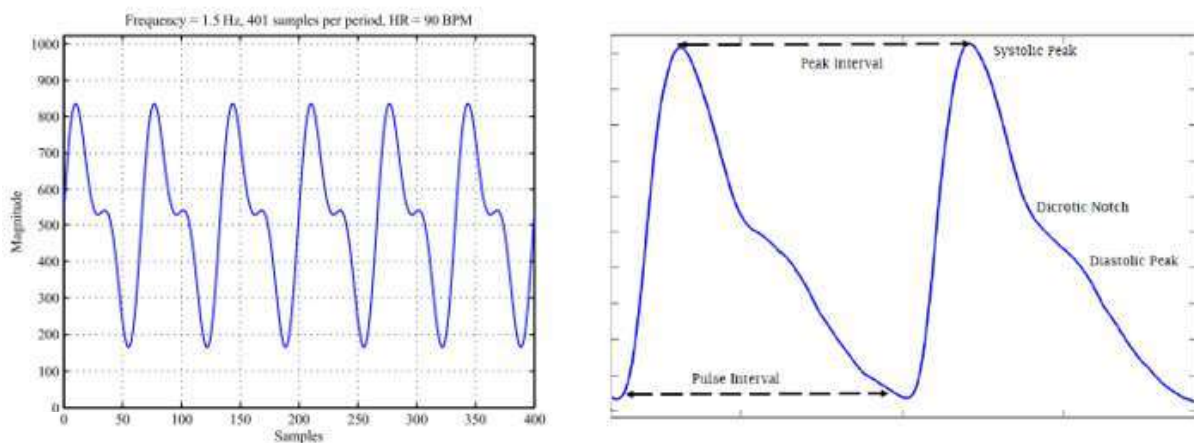


Figure 3.4: PPG Data Example

Chapter 4 (Software) details a better explanation of the heart rate calculation. To summarize, the heart rate stems from the time interval between each Systolic Peak (represented as the "Peak Interval" variable in Figure 3.4).

3.2.2: Gravity Heart Rate Sensor

The Gravity heart rate sensor (Figure 3.5) [28], also known as SEN0203, is a small heart rate monitor designed for Arduino compatible microcontroller boards and was created by DFRobot (robotics and open-source hardware provider).

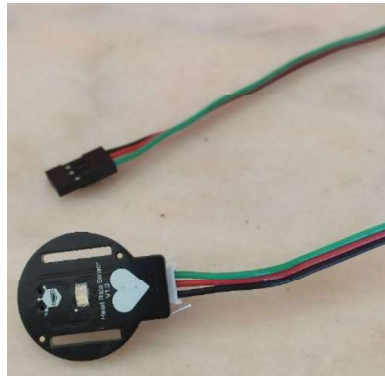


Figure 3.5: Gravity Heart Rate Sensor

The Gravity sensor is a pulse sensor that was built based on the previously mentioned PPG technique. The sensor can be used with a small belt, making its usability more comfortable and stable for readings; it can adjust and wrap on a finger or wrist, for example. The heart rate sensor has two kinds of signal output modes: analog pulse mode and digital square wave mode. These output modes can be changed using the dial switch implemented in the back of the sensor. One feature that the Gravity sensor comes with is that the connection cables (GND, VCC, and Signal) all come compacted together, which means there is a lot less room to work with unless the sensor uses some jumper cables also. This sensor is compatible with Raspberry Pi, Intel Edison, Joule, and Curie, utilizing a 3.3V Input Voltage.

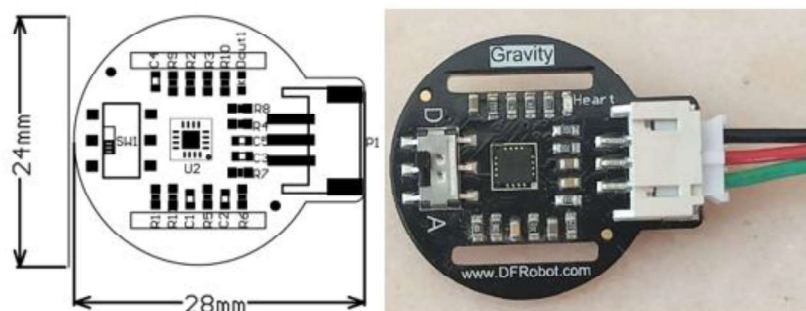


Figure 3.6: Gravity Sensor Architecture

Figure 3.6 shows that the SW1 switch corresponds to the dial switch that changes the output mode between analog pulse mode and digital square wave mode. If the dial switch mode

is not consistent with the mode used when writing the future Arduino code, the data will not show. The sensor has dimensions of 24x28mm. It has three cable connections for Ground (black), Voltage (red), and Signal (green), which all connect to an Arduino expansion shield via a three-pin format that connects to the microcontroller board.

3.2.3: VMA340 Heart Rate Sensor

Velleman made the VMA340 heart rate sensor (Figure 3.7) [29] for Arduino compatible hardware and software.

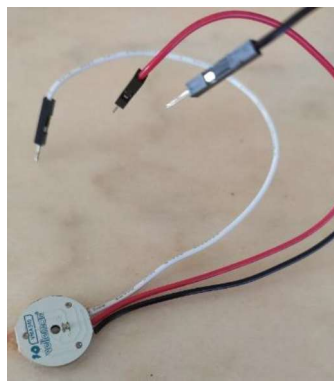


Figure 3.7: VMA340 Heart Rate Sensor

Much like the Gravity heart rate sensor, it is also small-sized and uses the same PPG technique to monitor and receive heart rate data. It differentiates from the previous Gravity sensor when it comes to the belt attached, which this sensor does not have, making it more difficult and uncomfortable to use on the wrist and other possible places. This sensor also does not have a dial switch to change from analog pulse mode and digital square wave mode, which the Gravity sensor mentioned before has. The only available signal mode for this sensor is the analog pulse mode. Some benefits the VMA340 sensor has when comparing to the Gravity sensor are:

- The simplicity of the sensor and perhaps, better results when receiving heart rate data (chapter 5 compares and analyses the results)
- The connection cables for the VMA340 sensor (GND, VCC, and Signal) are not combined, making room for more combinations and possibilities when connecting to the microcontroller board or other hardware.

Finally, the previous Gravity sensor is also much more popular than the VMA340 sensor. Therefore, it has more projects, documentation, and other support given from the open-

source community, which for the VMA340 sensor was somewhat challenging to obtain (due to its unpopularity).

The VMA340 sensor has three cable connections, just like the Gravity sensor, the Ground (black), Voltage (red), and Signal (white), shown in Figure 3.8. These cables all connect directly and individually to the microcontroller board, while the Gravity sensor needs the Arduino expansion shield's addition. Another difference is that the Signal cable is white instead of green, and as explained before, the cables are not compacted together, which leaves room for more possibilities.

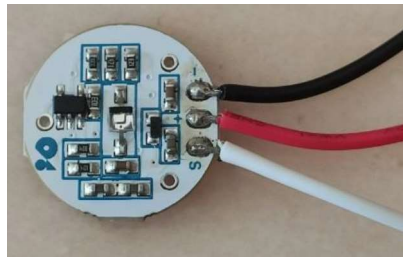


Figure 3.8: VMA340 Sensor Architecture

3.3: Arduino Compatible Hardware

This section first explains the microcontroller board and I/O expansion shield the project uses. Then it analyzes and describes their connection to the heart rate sensors.

3.3.1: DFRduino UNO R3 Microcontroller Board

The development of this dissertation's project employs the *DFRduino UNO (v3.0) R3* (Figure 3.9) [32] single-board microcontroller, compatible with Arduino software and hardware.

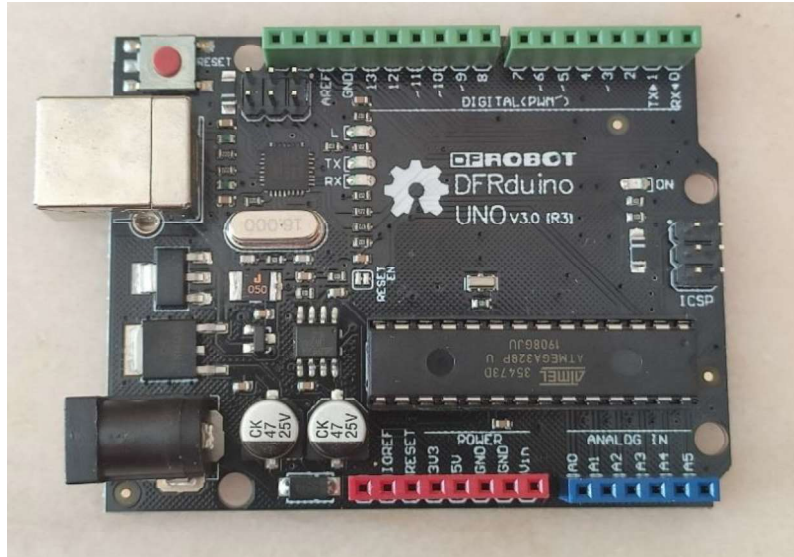


Figure 3.9: DFRduino UNO (v3.0) R3

Arduino compatible boards, like the DFRduino used, can read inputs such as light on a sensor, temperature of a room, a finger on a button; and then turn these inputs into desired outputs, for example: activating a motor, turning on an LED light, publishing something online, acquiring specific data, sending an alert message. The microcontrollers can be programmed to perform specific actions using an Integrated Development Environment (IDE) software. The software used to program the DFRduino microcontroller board was the Arduino IDE, which uses C and C++ programming languages. Through the Arduino IDE software, a program (sketch) is developed and then uploaded to the microcontroller board containing the code necessary to perform its desired output. For this dissertation's project, the microcontroller board aims to read the raw sensor data and calculate the heart rate BPM to output.

DFRobot built DFRduino UNO R3, and it is designed similarly to the more popular Arduino Uno. It is also fully compatible with it. This board features the ATmega16U2 chip, which essentially links and connects the computer's USB port and the central processor's serial port. Later, chapter 4 explains how vital this serial port and its connection are to establish a link between the microcontroller board data and the Arduino IDE and Node.js web server software.

Figure 3.9 shows the architecture of the DFRduino board. The board revolves around the single-chip microcontroller ATmega328. It has 14 digital I/O pins, 6 analog inputs, an ICSP (In Circuit Serial Programming) header, a reset button, a USB connection, and a power jack. It encompasses everything required to support the ATmega328 microcontroller. It can be connected to a computer via a USB cable or powered with an AC-to-DC adapter or a battery.

3.3.2: I/O Expansion Shield

The expansion shield used for this dissertation's project was the *Gravity: I/O Expansion shield V7.1* (Figure 3.10) [38], which was built by DFRobot to use on Arduino hardware and that is fully compatible with the DFRduino UNO R3 board used.

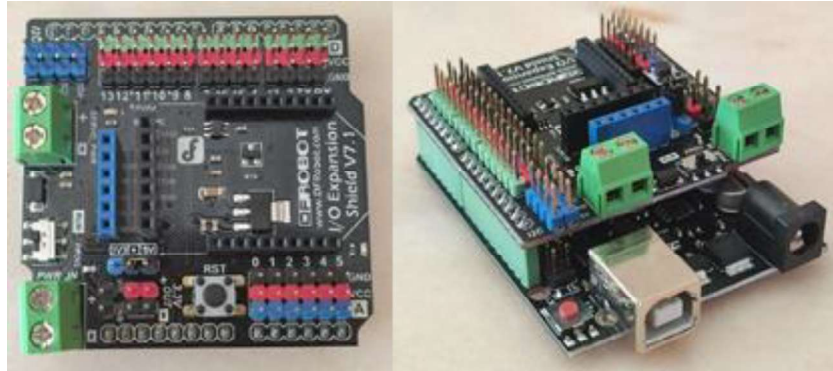


Figure 3.10: Gravity I/O Expansion Shield V7.1 (Left) and Inserted on the DFRduino Board (Right)

The expansion shield used comes with a three-pin Input format for Ground (GND), Voltage (VCC), and Signal that can be extremely useful in some applications. For example, several recent Arduino compatible hardware come with this three-pin cable format (GND, VCC, and Signal compacted together). This cable format can only be used with the same three-pin input that the Arduino expansion shield provides unless jumper cables connect each pin individually. Alternatively, the board also includes a practical and standard Xbee module that is used mainly for wireless communication purposes.

There are many advantages of using an IO Arduino expansion shield. However, the primary purpose of using one for this dissertation's project was the Gravity heart rate sensor that comes with the three connection cables compacted together (GND, VCC, and Signal), which are compatible with the expansion shield three-pins input configuration. Another reason was the future possibility of using Wi-Fi and Bluetooth communication to send the sensor data to the computer, instead of doing it through a USB cable.

3.3.3: Connecting to the Heart Rate Sensors

All the hardware has been explained, so now all that is left to do is see how everything combines itself to connect the computer to the microcontroller board with the heart rate sensor data.

Concerning the Gravity heart rate sensor connection, as mentioned before, the sensor has a three-pin cable format of Ground (GND), Voltage (VCC), and Signal that use the Arduino expansion shield in order to connect to the DFRduino UNO R3 microcontroller board. Figure 3.11 shows the overall connection.

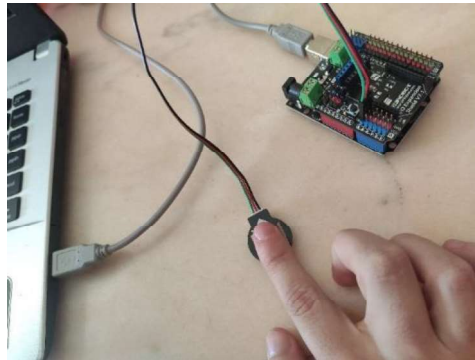


Figure 3.11: Gravity Sensor Connection

The three-pin cable (GND, VCC, Signal) of the Gravity heart rate sensor, seen in Figure 3.12, connects to the same three-pin Input format of the Arduino expansion shield at the A0 pins (for the analog mode output). This expansion shield then connects on top of the DFRduino microcontroller board that handles the sensor reading data and calculates the BPM heart rate to output to the computer via a USB type B cable.

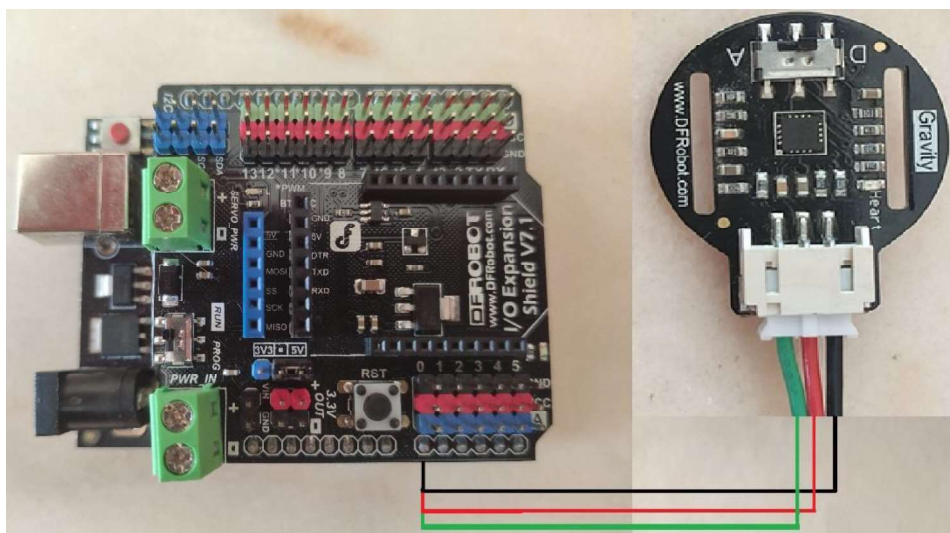


Figure 3.12: Gravity Sensor Connection Schematic

Concerning the VMA340 heart rate sensor connection, the Ground (GND), Voltage (VCC), and Signal cables come separated. The cables input individually, which means there is no immediate need to use the Arduino expansion shield. Figure 3.13 shows the overall connection.

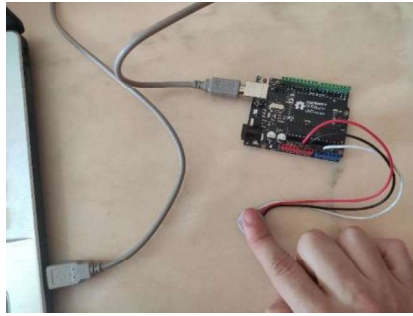


Figure 3.13: VMA340 Sensor Connection

It is possible to use just the DFRduino microcontroller board to handle the VMA340 sensor data. The Ground cable connects to one of the two GND inputs. The Voltage cable connects to the 5V input. The Signal cable connects to the A0 analog pin input (can be any analog input as long as it is referenced accordingly in the code program). After this, just like the Gravity heart rate sensor, the microcontroller handles the sensor reading data. It calculates the BPM heart rate to output through the USB type B cable connected to the computer. Figure 3.14 shows the cables connected to the board.

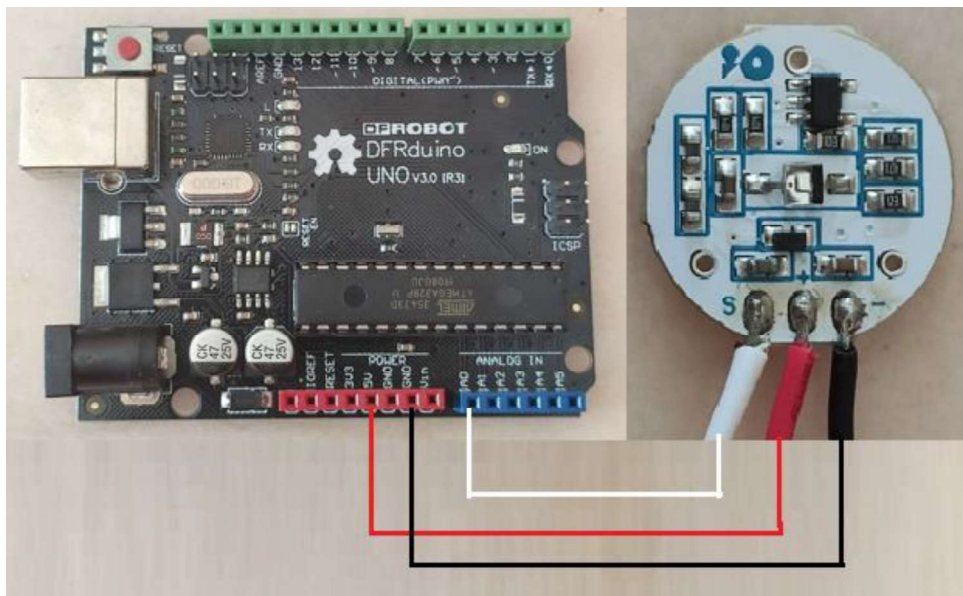


Figure 3.14: VMA340 Sensor Connection Schematic

Chapter 4 : Software

4.1: Software Architecture

Figure 4.1 displays the overall software architecture.

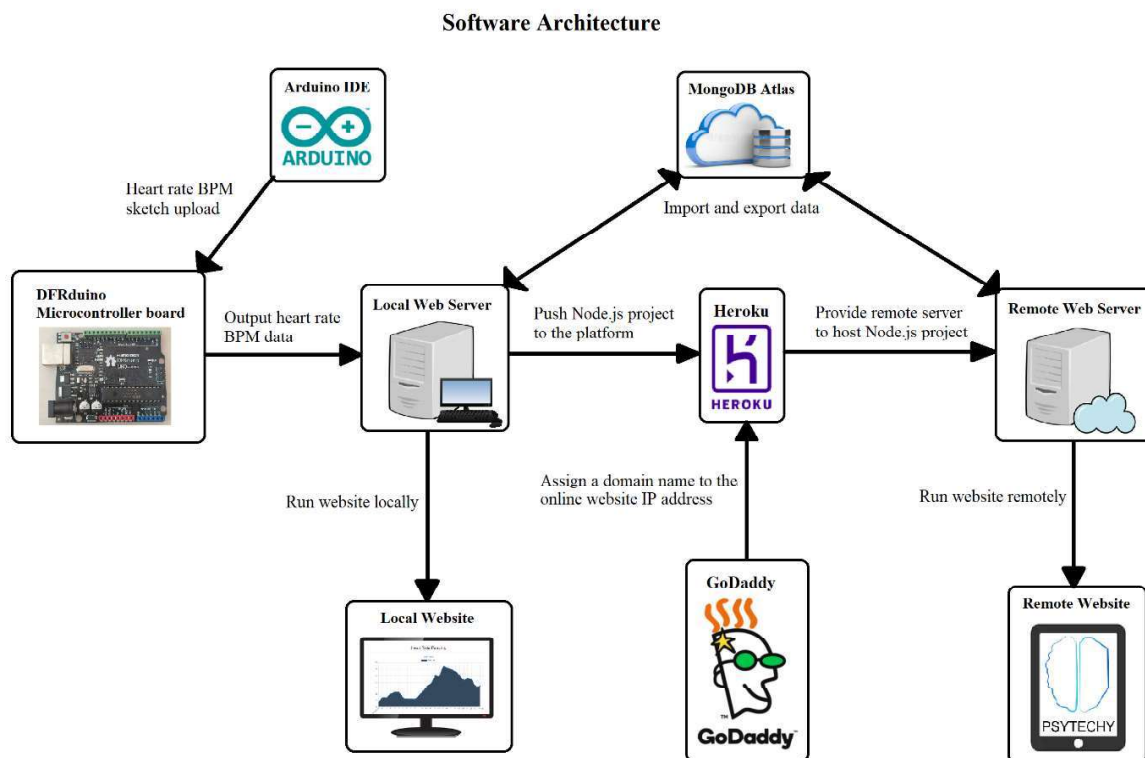


Figure 4.1: Software Architecture

This chapter describes how the sensor's data is collected and calculated into heart rate BPM using a sketch program uploaded to the microcontroller board through the Arduino IDE software. The Arduino IDE sketch, described in this chapter, is used to calculate the heart rate BPM of the two sensors used in the dissertation's project (Gravity and VMA340 sensors). In other words, the algorithm to calculate the heart rate BPM is equal for both sensors. The serial port connection established for the Arduino IDE and the Node.js web server is also equal for the two sensors.

After uploading the sketch, a "serialport" component created inside a Node.js web server captures the already calculated BPM sensor data, and using a "socket.io" component, the frontend framework (webpage) visualizes the sensor data. A cloud-based database named MongoDB Atlas stores all the essential information relevant to the project, acting as an online

hosting MongoDB database. Finally, the website project is deployed online through a cloud server hosting provider named Heroku and a domain name registrar named GoDaddy.

4.2: Arduino IDE

The software used to handle the project's sensor data is the Arduino IDE [39], which is an open-source platform compatible with the microcontroller board and heart rate sensors used in the project. The Arduino Integrated Development Environment (IDE) is a text editing program that essentially translates and compiles sketches into code that Arduino compatible hardware can understand.

When writing code in the Arduino IDE software, the program that runs that code is called a "sketch". The sketch code is in C/C++ and has the file extension ".ino". This sketch is the code uploaded to and ran on the DFRduino microcontroller board and contains the algorithm necessary to calculate the sensor's heart rate BPM to output. Section 4.2.2 further explains the sketch's algorithm and code.

4.2.1: Serial Port (COM Port)

The microcontroller board used, DFRduino UNO R3, is compatible with Arduino hardware, and therefore the Arduino IDE software can be applied to it. The DFRduino board features the "ATmega16U2" chip, which acts as a USB to serialport converter. Consequently, throughout the project, when the USB cable establishes the connection between the personal computer and the DFRduino board, the input is recognized as a serial port "COM". The computer automatically assigns a "COM" serial port as the next available one in the system. In this project's case, the system attributed the "COM3" port to the microcontroller board.

Figure 4.2 shows a sketch example containing the two primary functions (setup and loop) and the Arduino IDE software interface. The IDE recognizes the DFRduino microcontroller board as an Arduino Uno, which shares the same compatibility and features, and finally, the system attributes the COM3 port to the previous board.

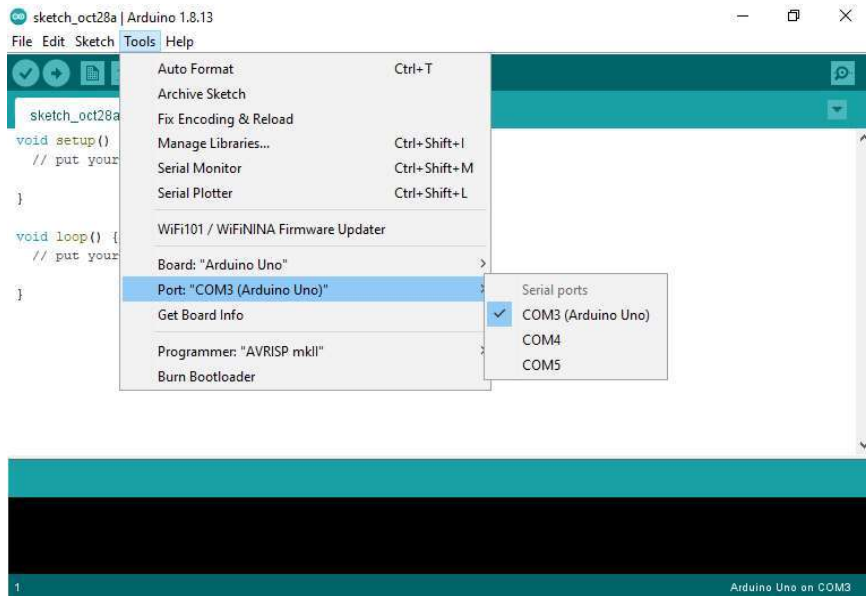


Figure 4.2: Serial Port Recognition and Arduino IDE Interface

The Arduino IDE automatically recognizes the DFRduino board and the COM serial port assigned to it (it can also be chosen manually in the IDE settings). Accordingly, all the future code developed inside the Arduino IDE is uploaded and applied to the DFRduino board through the designated COM serial port.

After the sketch program uploads to the DFRduino board, the local Node.js web server will then search and connect to the same serial port “COM3” to receive the sensor’s data, already transformed into heart rate BPM through the uploaded sketch.

4.2.2: Heart Rate BPM Algorithm

The *Appendix* chapter explains and describes the context and development behind the heart rate BPM algorithm used inside the sketch program uploaded to the microcontroller board. This algorithm is responsible for calculating and outputting the desired sensors’ heart rate BPM values.

4.3: Website Server

This section goes over the web server connection to the sensor through the “serialport” library. Details how the web server handles the sensors’ data and then visualizes it. Finally, it describes the programming languages, essential classes, and files that the project uses.

4.3.1: Web Server Connection to the Sensor

“Serialport” [80] is a library for Node.js that allows communication over a computer’s serial ports and enables the creation of a “Serialport” object which will open a specified port, allowing the reading and writing of data of that same port. This library establishes a link between the Node.js web server and the DFRduino microcontroller board transmitting the two sensors' heart rate BPM data.

Figure 4.3 contains a flowchart representing the overall flow of the sensor's data connection to the Node.js web server and how the server handles that data. This flowchart (Figure 4.3) comes to a stop on the node "A", which then continues in section 4.3.2, Figure 4.5. The flowchart's continuation explains how the sensors' data communicates to the client-side of the Node.js project, using sockets, and how the website visualizes that data.

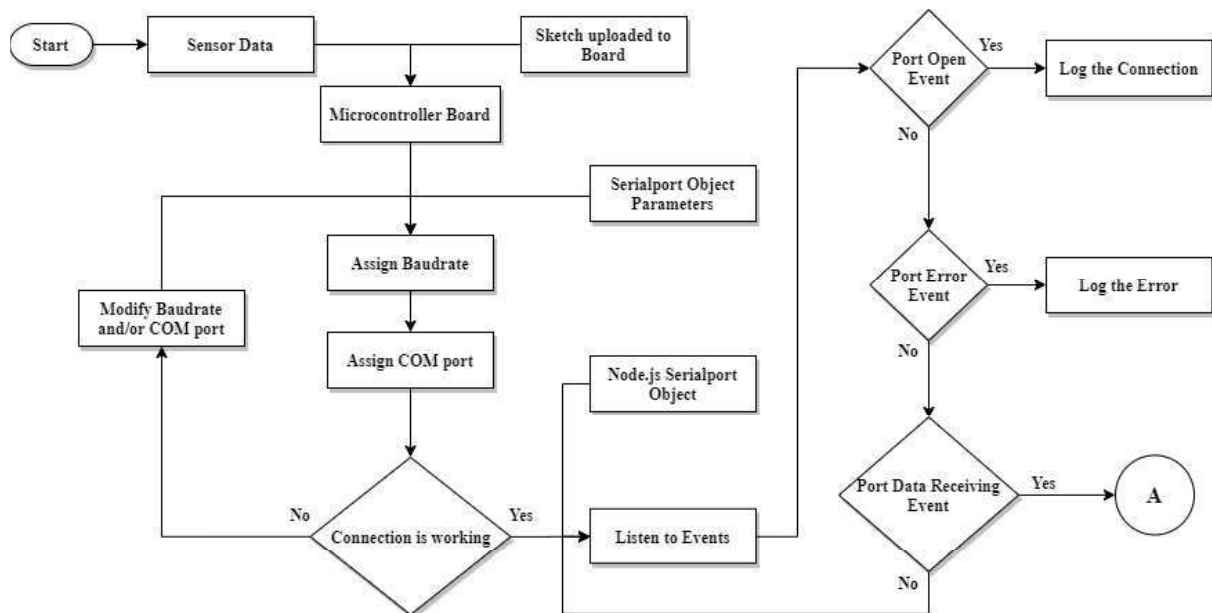


Figure 4.3: Flowchart of the Web Server Handling the Sensor Data

Opening a serial port connection in Node.js requires the library's import at the beginning of the Node.js project main file. The file name is "app.js", and a local variable instantiates the library.

After this, the web server creates a Serialport object (called "mySerial"), containing the port name (COM3) and the Baudrate (9600) parameters, as can be seen in Figure 4.4. The COM port number is automatically attributed and appears when using the Arduino IDE or the Windows device manager. Simultaneously, the Arduino IDE sketch uploaded to the DFRduino

board defines the baudrate. These values need to be equal to the Serialport object parameters, or else the data will not be readable.

The new Serialport object can listen for events from the serial port. Signifying that when the DFRduino board and the heart rate sensors are connected and running, specific actions will generate events, and the Node.js web server will provide functions to deal with those events called callback functions.

The events used for this project are opening a new serial port connection, receiving new sensor data, and an error incident. Using the method "mySerial.on(...)", it is possible to create a specific function that will get called when each event transpires.

Figure 4.4 shows the serialport object (mySerial) creation and the events he listens. When the serial port opens, a function will get called to log the new connection event in the console. When new data arrives, a parser will read the data from the serial port, log it in the console and send it to a socket which displays it on a webpage. Finally, if an error occurs, a function will get called to log the error in the console.

```
var mySerial = new serialport('COM3',{
  baudRate : 9600
});

mySerial.on("open", function(){
  console.log('opened serial port');
});

mySerial.pipe(parser);
parser.on("data", function(data){
  console.log(data.toString());
  io.emit("arduino:data", {
    value: data.toString()
  });
});

mySerial.on("err", function(err){
  console.log(err.message);
});
```

Figure 4.4: Serialport Object Creation and Events

To conclude, the code showed in Figure 4.4 demonstrates and explains how the Node.js web server can connect to the DFRduino microcontroller board that transmits the data sent from the heart rate sensors. The web server creates a serial port object that listens to events, such as receiving data from the sensor, and then invokes a function whenever it calls a specific event.

4.3.2: Visualizing the Sensor's Data

Continuing the previous flowchart (Figure 4.3), Figure 4.5 explains what happens after the Serialport object receives the new sensor data event (A) and how the web server handles that data to visualize it on the web page.

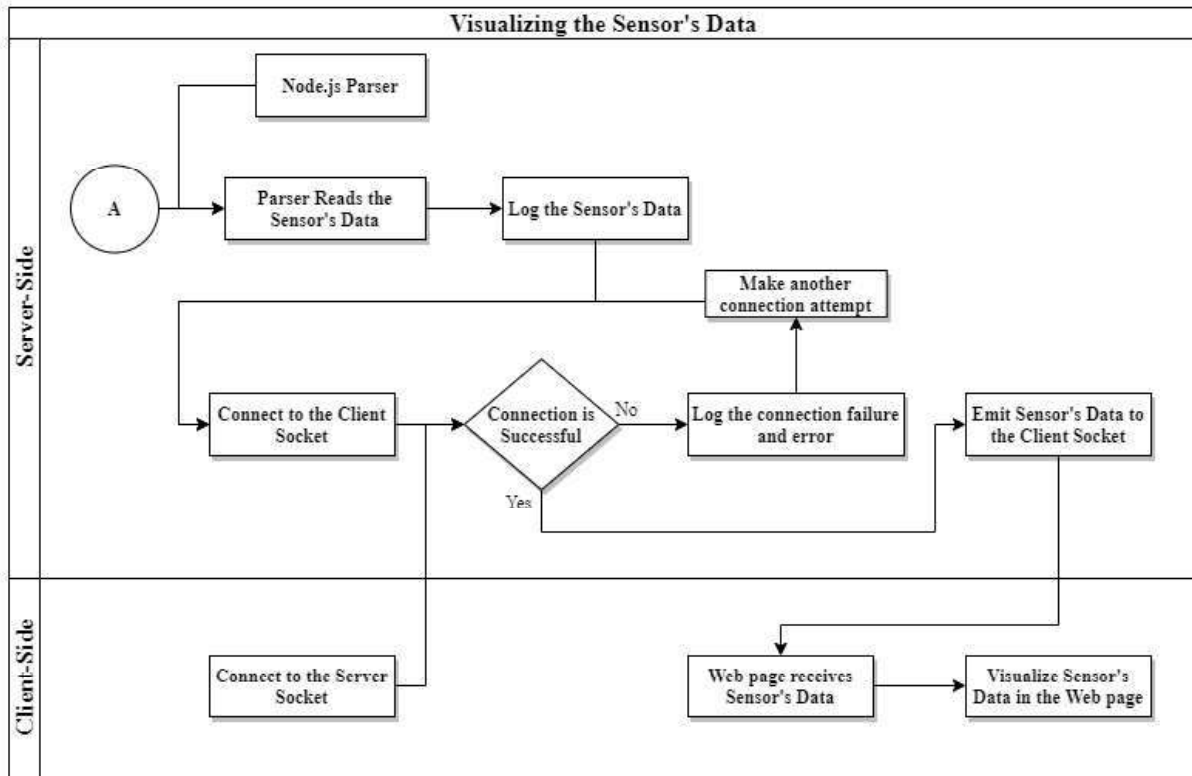


Figure 4.5: Sensor's Data Visualization

First, the web server creates a parser object to read the data lines that arrive from the Serialport object. The purpose of using a parser is because the Serialport object listener function generates a data event once every byte. In contrast, the parser object allows generating a data event once every newline instead.

Using the Serialport object explained before, it is now possible to handle the data from the heart rate sensors inside the web server. Nonetheless, the sensor's data connection between the server-side and the client-side of the Node.js project is still not operational. Even though the sensor's data reaches the web server, the client-side web page does not yet have the means to obtain that same data. To solve this, the communication between them employs the "Socket.IO" library.

"Socket.IO" [81] is an available Node.js library that allows client-side web pages and web servers to communicate using a bidirectional and event-based method. For the

bidirectional communication to be effective, both the Node.js web server and the client-side web page need to implement the Socket.IO library.

First, the Node.js web server main file "app.js" includes the Socket.IO library. The web server creates and setups a socket named "io". The socket will listen for a "connection" event and runs the provided function anytime it happens. For example, whenever a socket on the client-side is connected, the server socket will call the "connection" event and run a function to log that information in the console.

The web server uses the same method above to send the heart rate sensor's data to the client-side web page, visualizing the data. The server-side handles this function and calls it whenever the Serialport object handles the event of receiving new data from the sensor. Inside that function, seen in section 4.3.1 and Figure 4.4, the socket uses the method "io.emit("SensorData")" to send a message to the connected clients (the web page) containing the heart rate sensor's data. As explained before, the Serialport object in the Node.js web server connects to the same port containing the DFRduino board that consequentially connects to the heart rate sensors. Figure 4.6 shows the client-side code where its socket is created and receives the server-side sensor's data.

```
<script>
  const socket = io();
  let counter = 0;
  socket.on('arduino:data', function(datasensor){
    myChart.data.labels.push(counter);
    myChart.data.datasets.forEach(dataset => {
      dataset.data.push(datasensor.value);
    });
    counter++;
    myChart.update();
  });

```

Figure 4.6: Client-Side Socket

A socket instance needs to be on the client-side of the Node.js project (Figure 4.6), ready to receive the information sent from the server, in this case, the heart rate sensor data. The client-side web page creates this socket instance, in this case, inside a JavaScript script of an EJS file. After this, the socket on the client-side will be listening for the heart rate data from the sensor, just like the socket on the server-side, and then using that data, the same script will plot the values inside a graph and update it continuously to the web page. Figure 4.7 shows the final visualization result of the heart rate sensor's data on the web page.



Figure 4.7: Sensor's Heart Rate Data Visualization

The tested heart rate showed above (Figure 4.7) starts a bit low, averaging 50 BPM, probably due to slight movement or noise in the sensor's readings, and then stabilizes around the 60-75 BPM range.

4.3.3: Programming Languages, Essential Classes and Files

It is also vital to understand what programming languages, significant classes, documents, or any other elements crucial to the dissertation's web development are in use. The main programming languages used for the project's web development are:

- C and C++ for the Arduino IDE sketch code;
- JavaScript for the Node.js web server, or in other words, the backend framework;
- CSS for some of the styles used in frontend development;
- EJS [82] (a combination of HTML and JavaScript) for the frontend framework used to visualize the client-side web pages.

To obtain a better understanding of EJS (Embedded JavaScript), it is a modest templating language that permits HTML markup generation with pure JavaScript. It enables usage of both languages without any concern and both in the server-side or client-side of the Node.js project. For example, in Figure 4.8, "div" is a container class that defines a division or an HTML document section. This "div" contains an unordered list "ul" which comprises of list

items "li". All of this is HTML code, but there is also JavaScript code delimited by the "<%>" marks inside the "div" container, which performs backend operations on the client-side. In Figure 4.8, the JavaScript merely finds out if a user is logged at that moment, and if there is one, then it writes the username on the page. This JavaScript code would be much more demanding and complex to write in a plain HTML file and might invoke more errors, but using EJS files simplifies it.

```
<div class="collapse navbar-collapse">
  <ul class="nav navbar-nav navbar-right">
    <li><a href="/sensor">BPM Sensor</a></li>
    <li><a href="/information">About us</a></li>
    <li><a href="/contacts">Contacts</a></li>
    <% if(!currentUser){ %>
      <li><a href="/login">Login your account</a></li>
      <li><a href="/register">Register</a></li>
    <% } else { %>
      <li><a href="#">Welcome, <%= currentUser.username %></a></li>
      <li><a href="/logout">Logout</a></li>
    <% } %>
  </ul>
</div>
```

Figure 4.8: EJS Example

A tool that helped develop the project and is frequently used in combination with Node.js is npm (Node Package Manager) [83]. Npm is an online repository that publishes open-source Node.js projects. It and can also easily interact with the same repository to install and organize packages, manage versions and more. For example, the modules/libraries “serialport”, “express”, “socket.io” and “mongoose” were installed using npm.

One of the most critical files for running the web server is the JavaScript file called “app.js” (usually it is called “app.js” or “main.js”), which is used to run the main program and contains all the most crucial steps of the Node.js web server. All Node.js projects contain a similar file, and it usually is the first file to be created and the most modified one. The “app.js” file used for this dissertation Node.js project contains, for example, the connection to the database, requiring/importing of essential libraries, creation of a host to run the server on the internet, connection to the serial port with the heart rate sensor data and more.

The “node_modules” [84] folder contains all the modules imported or required for the project using the tool npm, described above. In Node.js, a module can be considered the same as a JavaScript library. All the installations made through npm are inserted by default inside the node_modules folder.

The “package.json” [85] is a JSON type file created at the beginning of the project with npm. The package.json file exists at the root of the Node.js project and holds various metadata

relevant to the project. It provides information to npm to identify the project and manage the project's dependencies, scripts, versions, and more. For example, inside this project's package.json file, all the modules/libraries versions ("express:4.17", "serialport:7.1.5") can be read.

4.4: Database

The database used for the Node.js project is a document-oriented database program named MongoDB [45].

Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. The MongoDB database has many advantages when combined with the Node.js software and is commonly used with it, as is the case with this project.

Like the other Node.js modules or libraries, the installation of MongoDB in our node project uses the npm tool.

4.4.1: Mongoose

Mongoose [46] is an Object Data Modeling (ODM) Node.js library used for MongoDB databases.

An example of a schema used in this project is the "user.js", which stores a user's login information (name and password) inside a document data structure that is later used to gather information. That information determines whether there is a particular username in the MongoDB database to confirm a login attempt. Figure 4.9 shows the “user” schema, the mongoose library import, and the export of that schema to mongoose.

```
var mongoose = require("mongoose");
var passportLocalMongoose = require("passport-local-mongoose");

var UserSchema = new mongoose.Schema({
  username: String,
  password: String
});

UserSchema.plugin(passportLocalMongoose);
module.exports = mongoose.model("User", UserSchema);
```

Figure 4.9: “User” Schema

Using the mongoose module requires to have the MongoDB module first installed. Npm handles both installations. Much like the other modules used in this project, the first thing

needed is to import the mongoose module/library into our “app.js” file. The "app.js" file, as mentioned before, handles the database connection and creation. A cloud database instance of MongoDB called MongoDB Atlas establishes the database online. The next sections explain in more detail how it works and how Node.js connects with it.

4.4.2: MongoDB Atlas

MongoDB Atlas [47] is a fully managed cloud MongoDB database and was used to develop this project.

To connect to a MongoDB database hosted on the cloud using MongoDB Atlas from a Node.js server, a MongoDB://URL is used, which the MongoDB Atlas platform provides for us. The URL also contains the login information (name and password) used to connect to the MongoDB Atlas, shown blurred in black in Figure 4.10.

The web server uses the method “mongoose.connect(“MongoDB://URL”)” on the main file of the Node.js project “app.js”, the connection between the project and the MongoDB database hosted on the cloud by MongoDB Atlas is established, and it is now possible to easily store and manage the project data to it through the use of the mongoose library. Figure 4.10 shows the code used to establish that connection and a function called when the MongoDB database is connected successfully or when there is an error.

```
mongoose.connect("mongodb+srv://[redacted]:[redacted]@cluster0-2yslj.mongodb.net/test?retryWrites=true&w=majority", {
  useNewUrlParser: true,
  useCreateIndex: true
}).then(() => {
  console.log("Connected to DB");
}).catch(err => {
  console.log("ERROR:", err.message);
});
```

Figure 4.10: Mongoose Connection

To send data to the database, a mongoose “Schema” is created, as shown before with the “user.js” schema (Figure 4.9). For this example, whenever a user is created through a registration form in the website, a new user mongoose schema is created and stored in the MongoDB database via mongoose, connecting to the cloud-host MongoDB Atlas. The connection between the Node.js server and the MongoDB Atlas database is always online, whenever there is no errors, and the server is kept running.

The MongoDB database collections can be seen online in MongoDB Atlas and can always be managed there. An example of a document containing the login information for a “Francisco” user can be seen at the bottom of Figure 4.11.

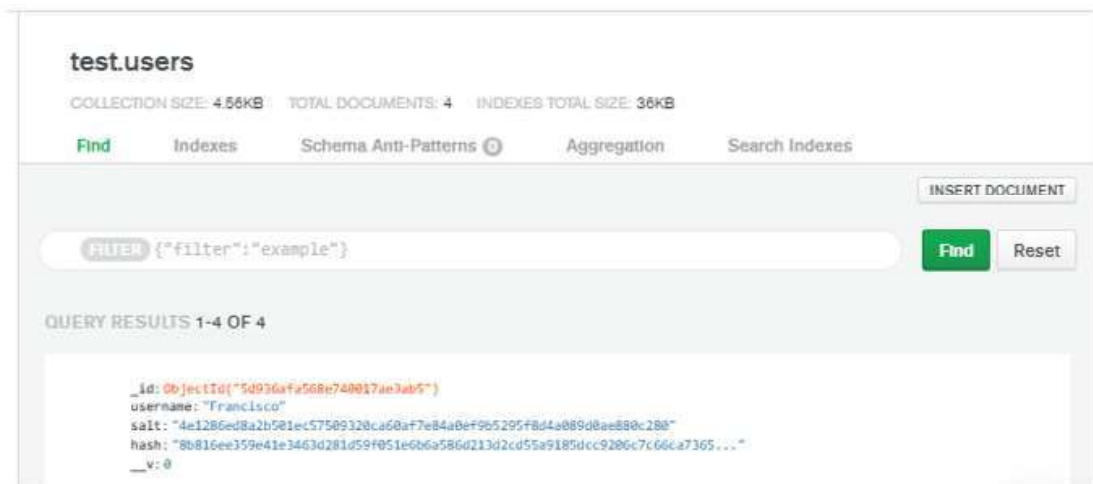


Figure 4.11: Collection Containing a Document for a “Francisco” User

4.5: Website Deployment

To deploy the Node.js project to the internet and make it run online, some steps were realized. Such as buying a domain name to use for the website and deploying the Node.js web server to a PaaS named "Heroku", which provides an online server running continuously to host the Node.js web server, which was running locally on the computer before.

4.5.1: Online Server Hosting “Heroku”

Heroku [60] has some advantages when combined with a Node.js web server and a MongoDB database. GitHub was also used to establish the connection between the Node.js project and the Heroku services, such as pushing and committing the project's files to Heroku's remote server.

Figure 4.12 contains a flowchart explaining how the Node.js web server was deployed through the Heroku platform.

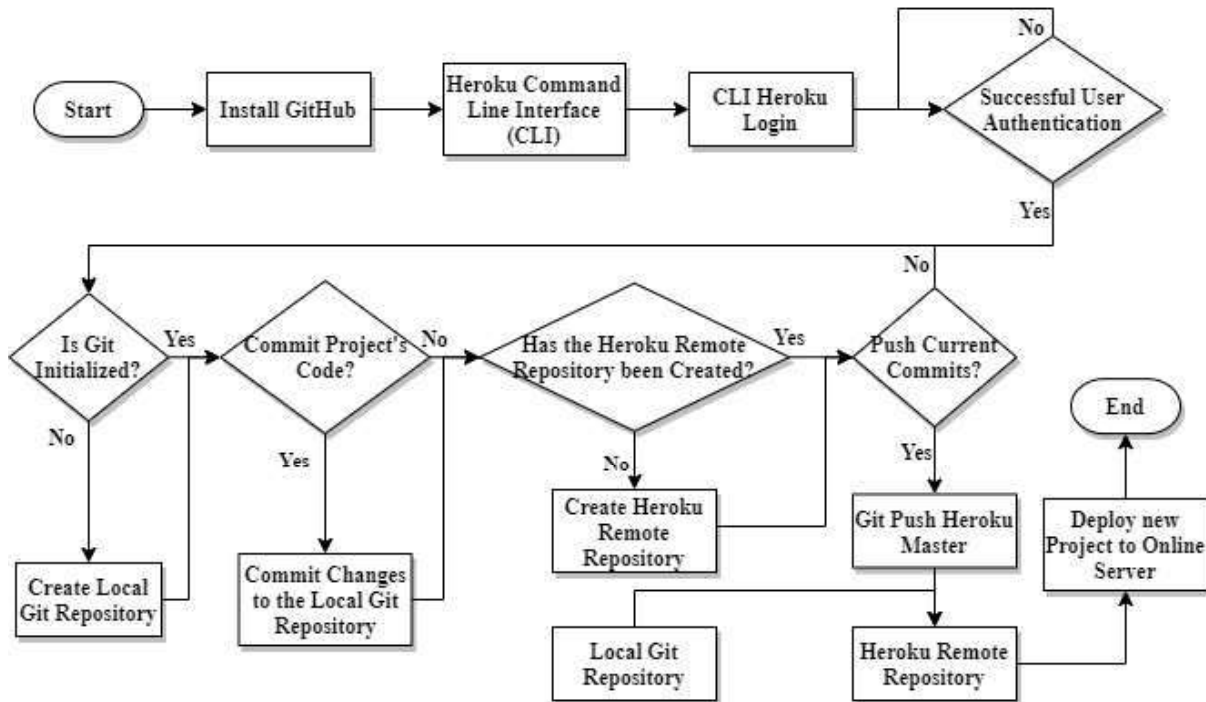


Figure 4.12: Flowchart of the Node.js Web Server Deployment

First, a local git repository is initialized, and the project's code changes are committed to it. The commits on the local git repository are then pushed to a Heroku git remote, which is a version of the local repository that lives on the remote server. After pushing the project's code to the Heroku git remote, it is then associated with the specific application that deploys the online website.

Installing GitHub and the Heroku Command Line Interface (CLI) [86] is the first step to establish the connection between the Node.js project and the hosting service Heroku. The CLI allows to manage and scale the application, view its logs, and run it locally.

Most of the project's website deployment is made through GitHub commands on the Heroku CLI. The first command, "heroku login", is to login into the Heroku CLI, which will then request authentication using the user's username and password. This login is necessary in order to continue the website deployment to Heroku.

After performing the login, a local git repository needs to be initialized if there is not one already, and the application's code needs to be committed to it. It is essential to initialize the local git repository in the website project's root directory; otherwise, it will not be possible to push the code.

Next, a Heroku git remote is created, if there is not one already, and is associated with the application that deploys the online website. Finally, by pushing the local git repository commits to the Heroku git remote that lives on the remote server, the application deploys the online website. Figure 4.13 explains most of the Heroku CLI commands necessary for the Node.js web server deployment.

```
heroku login
//login is necessary to continue deploying the website

git init
//initializes a local git repository for the project

git commit -m "testing commit"
//commit project's code changes to the local git repository, with a comment

heroku create
/*creates the heroku git remote and the application associated with it
the created heroku git remote is linked to the project's local git repository*/

git push heroku master
//push the committed code in the local git repository to the heroku git remote
```

Figure 4.13: Heroku CLI Commands Used for the Deployment

4.5.2: DNS Connection to the Remote Server

DNS [55] stands for Domain Name System and its purpose is to translate domain names into IP Addresses.

A domain is a name used for a website. For the Node.js web server project used in this dissertation, three domains were bought for a year, and two of them were renewed and are still ongoing. The three domains bought were: psytechy.com, psytechy.org (no longer working), and psytechy.pt. These domains were bought through the domain name registrar “GoDaddy” [58] (for .com and .org) and “dominios.pt” [59] (for .pt).

After the deployment is done, Heroku will provide a random domain name by default to access the online website, for example, "serene-example-4269.herokuapp.com". To change this random domain name to a custom one ("psytechy.com") that was bought previously on the domain name registrar GoDaddy, the DNS provided by GoDaddy needs to point to the DNS provided by Heroku. Using the command, "heroku domains add", on the Heroku CLI or checking the project's applications settings on the browser, it is possible to add custom domain names (in this case, www.psytechy.com). Next, a DNS target will be shown that points to the domain name registrar GoDaddy, as seen in Figure 4.14. This DNS target needs to be inputted into the GoDaddy DNS to finalize the connection between the two services and conclude the

online hosted website's domain name setup. Figure 4.14 shows the domain names used in this project and their designated DNS target.

You have multiple custom domains enabled Add domain

Q Filter domains

Domain Name	DNS Target ?	
psytechy.com	concentric-springs-qh695zk4a6cprts2xyuo...	
psytechy.pt	amorphous-tundra-z3li7cj14ogplka2kwsxe...	
www.psytechy.com	mammalian-penguin-ubtue0u5q024j2ktlc...	
www.psytechy.org	limitless-falcon-pjbdjrhcxfkkk5w79rss5rvf...	
www.psytechy.pt	opaque-badger-jg7dk0q95yngm3kp8xmh...	

Figure 4.14: Project Domain Names and their DNS Target

Heroku also contributes with optional config vars, shown in Figure 4.15, which are variables established inside the project's code but can be accessed and modified on the Heroku browser settings manually, without changing the code internally. This functionality is used by creating a config var named "DATABASEURL" (Figure 4.15) containing the value of the URL provided by MongoDB Atlas needed to establish the connection from the Node.js web server and the cloud MongoDB database (the same URL shown in section 4.4.2, on the `mongoose.connect(URL)` method).

Config Vars Hide Config Vars

DATABASEURL	mongodb+srv://[REDACTED]@cluster0-2ysij.	
KEY	VALUE	Add

Figure 4.15: DATABASEURL Config Var

Chapter 5 : Results

This chapter describes the results obtained from the dissertation project's development. First, the tests implemented are explained and showed. Following the tests, an analysis of them was conducted to determine any main points where improvement can be applied.

5.1: Tests

The entire project requires a stable and synchronized connection between all the hardware and software used.

This connection is fundamental to transmit the heart rate sensor's data to the personal computer and consequentially to the Node.js web server that carries out its visualization. Therefore, both the project's connections and the sensors' data transmission were tested, and the results are explored here.

The two heart rate sensors used are tested and compared, concerning the data quality and accuracy of the heart rate readings gathered. Both sensors were tested equally, but some of the VMA340 sensor tests are not shown for the sake of simplicity. Instead, the Gravity sensor tests are used, containing similar results.

Next, the local server running the local website is tested on its features, such as visualizing the sensor's data and registering new users on the database.

Finally, the remote server running the remote website is tested on its domains, accessibility, and database communication.

5.1.1: Connection Tests

The first tests done were connection tests, ensuring the hardware components were being linked together successfully and then connecting to the necessary software on the PC.

Recognition tests were employed through the Arduino IDE software (Figure 5.1) to test the DFRduino microcontroller board's connection to the PC and the appropriate software.

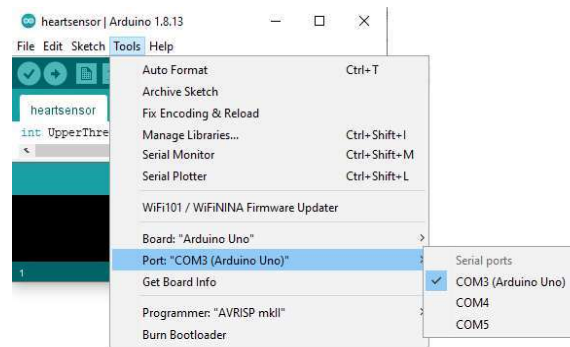


Figure 5.1: Microcontroller Board Recognition Test

A sketch test is also uploaded to the microcontroller board (Figure 5.2) further validating its connection and future sketch uploads.



Figure 5.2: Sketch Upload Test

Next, the heart rate sensors were inputted onto the microcontroller board and the I/O extension board (if necessary). The sensors' connection was tested with the LED light (green) implemented in the sensor, necessary for the PPG technique, and that turns on whenever the sensor has a power connection. The microcontroller board has a LED red light that verifies its connection to a power supply (PC). Figure 5.3 demonstrates the test, with the VMA340 sensor on the left and the Gravity sensor on the right.

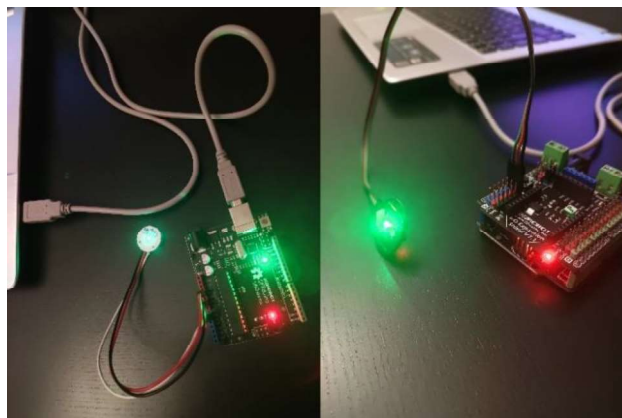


Figure 5.3: VMA340 (Left) and Gravity (Right) LED Lights Test

5.1.2: Sensors Data Transmission Tests

After finishing the hardware components connection tests, the heart rate sensors' data communication and transmission need to be tested. It is crucial to verify that the hardware and software involved is synchronized and working together, as expected.

The sensor's data transmission was first verified by uploading a sketch to the microcontroller board that tested if both sensors could send their raw data whenever they detected a pulse reading. The sensors raw data was then observed over the Arduino IDE Serial monitor and Serial plotter, demonstrated in Figure 5.4.

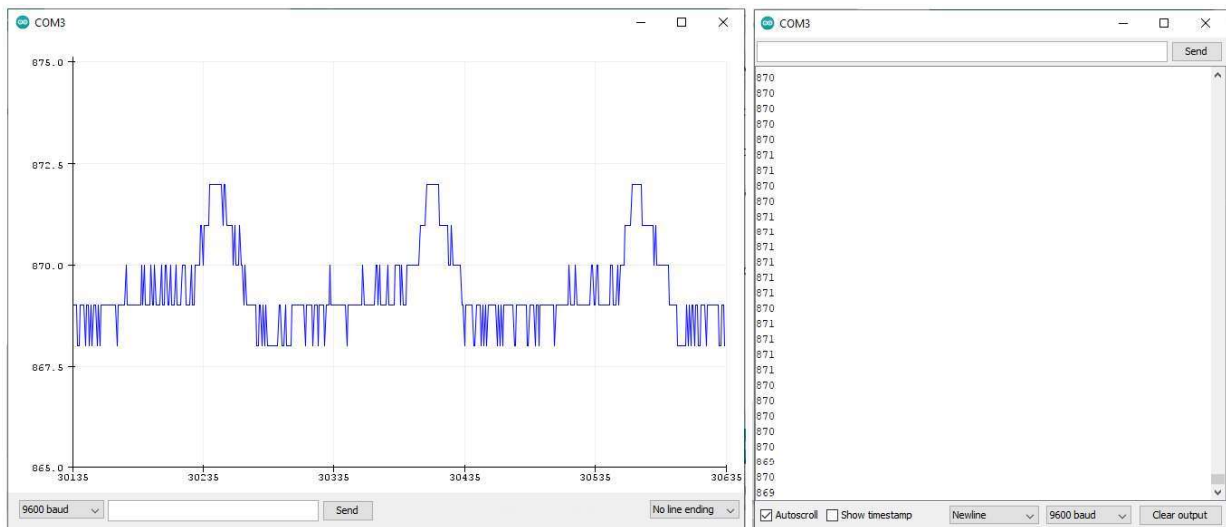


Figure 5.4: Sensor's Raw Data Serial Plotter (Left) and Serial Monitor (Right) Tests

After testing the sensors' data transmission to the Arduino IDE, the central sketch program containing the heart rate BPM calculating algorithm is also tested for errors. It is then uploaded to the microcontroller board. Just as before, the new heart rate BPM calculated data sent from the sensors is tested on the Arduino IDE software, demonstrated in Figure 5.5.

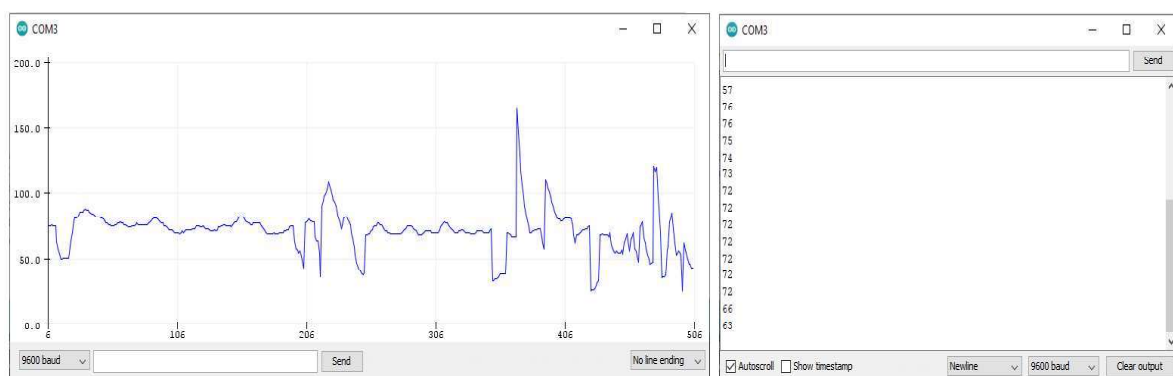


Figure 5.5: Sensor's Calculated Data on the Serial Plotter (Left) and Serial Monitor (Right) Tests

5.1.3: Heart Rate Sensors Tests

After transmitting the sensors' heart rate BPM calculated data, the next step was testing that same heart rate data quality and accuracy, and then compare the results between the two sensors.

The two sensors calculated heart rate BPM is tested and gathered via the Arduino IDE software serial monitor. Figures 5.6, 5.7, and 5.8 show the graphs containing the heart rate BPM data gathered from both sensors for three minutes and around the same pulse counts (156). The tests involved three people and were done while resting and after a one-minute workout exercise. Person A is male, 23 years old, and is semi-athletic. Person B is female, 59 years old, and is not athletic. Person C is male, 30 years old, and is semi-athletic.

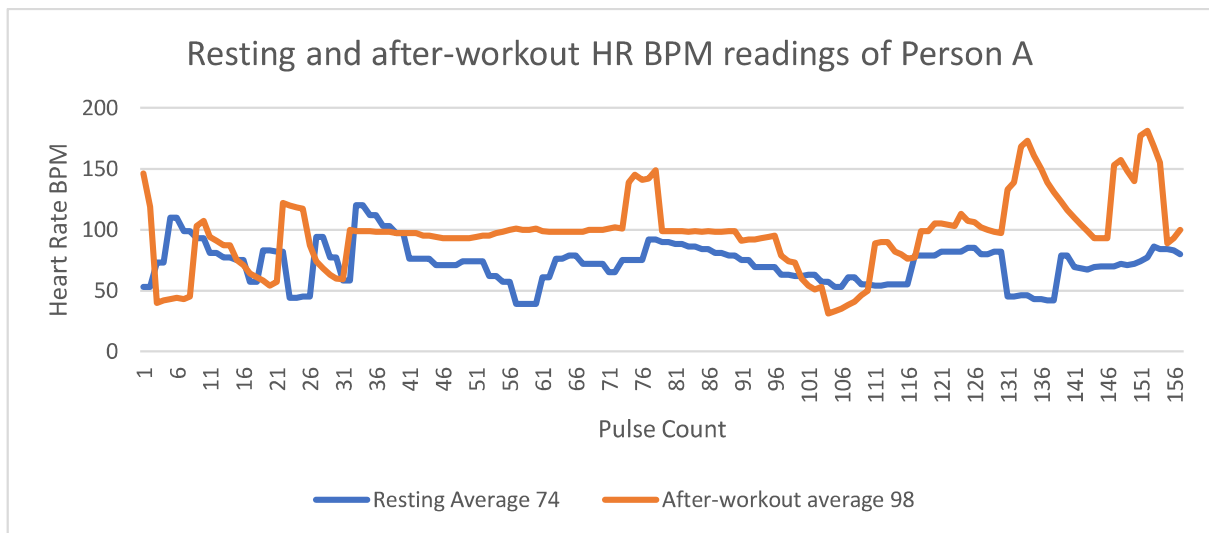


Figure 5.6: Heart Rate BPM Person A Test

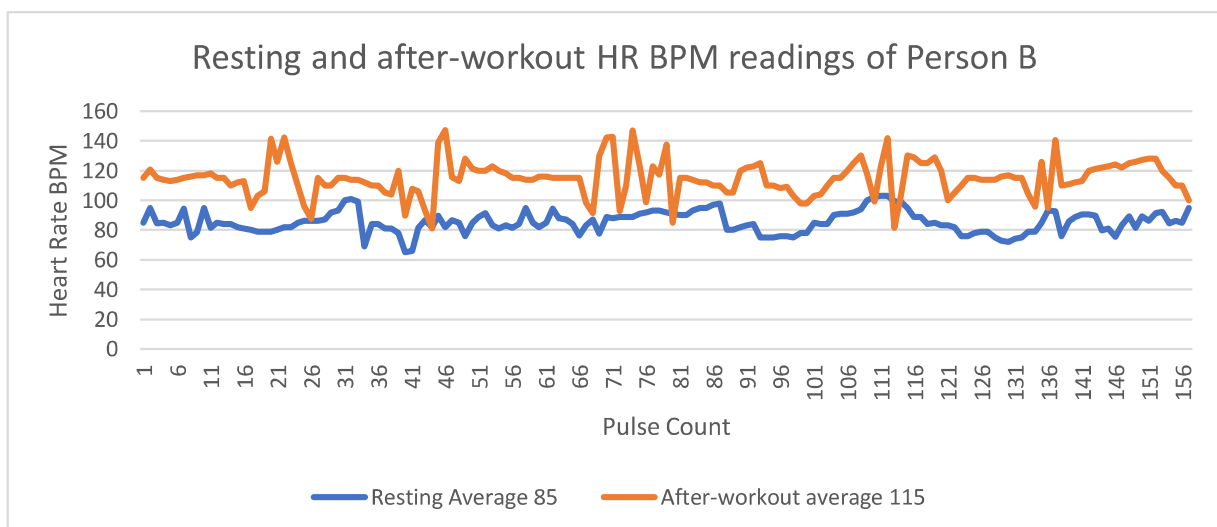


Figure 5.7: Heart Rate BPM Person B Test

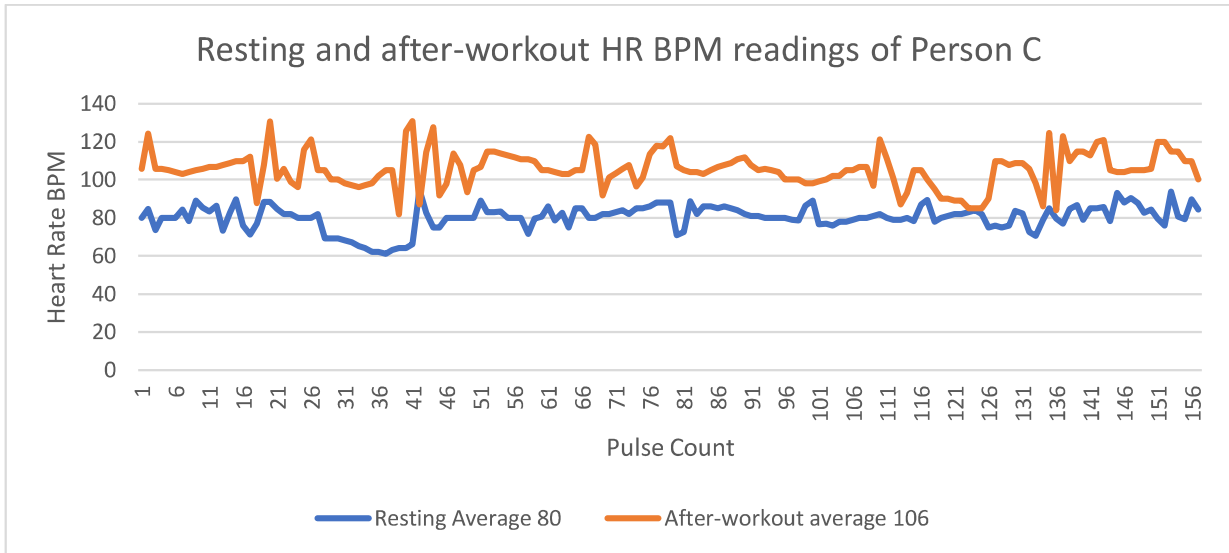


Figure 5.8: Heart Rate BPM Person C Test

5.1.4: Local and Remote Server Tests

The local and remote servers have a crucial role in this dissertation's project since they host the local and remote websites. The websites visualize the sensors' data and enable access to their information and results.

The local server hosting the local website was the first to be tested. The most important feature of the local server was to visualize the sensors' data on the webpage. For this to happen, the sensor's data was tested by outputting its contents to the Node.js web server console, shown in Figure 5.9.

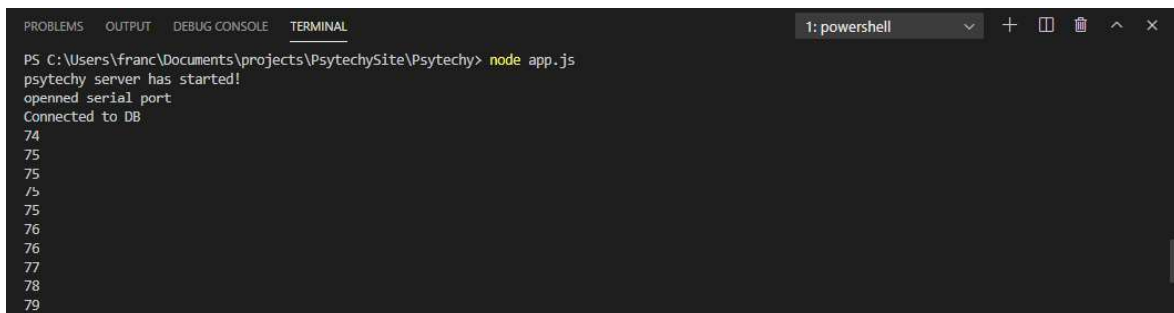


Figure 5.9: Sensor's Data Transmission to Web Server Test

After, the local Node.js web server (server-side) transmission of the sensor's data to the client-side web page was tested, resulting in its visualization as a graph that continuously updates, shown in Figure 5.10.

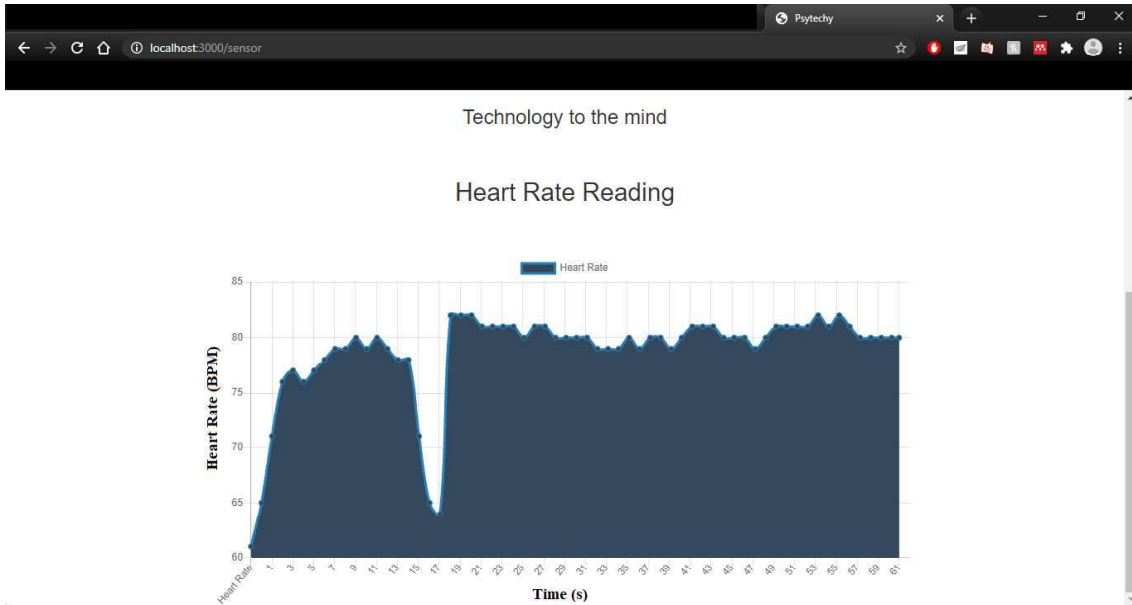


Figure 5.10: Sensor's Data Visualization Test on the Local Server

Tests involving the MongoDB database were also conducted. The connection from the local Node.js web server to the MongoDB Atlas cloud platform that hosts the project's MongoDB database was tested. Following the database connection, a new user was registered on the website to test the new user's schema information being sent to the cloud database. Figure 5.11 shows the registration form and the MongoDB collection that obtained the newly registered user document. Login is then submitted to the local server to test if the new information was registered and requested.

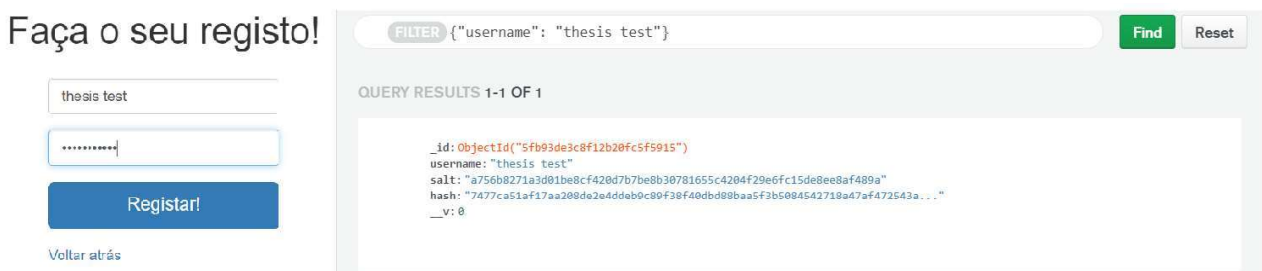


Figure 5.11: User Register Form (Left) and its Designated MongoDB Document (Right)

After testing the local server, the remote server and the remote website were tested. The first test was to verify the domain names being used, “psytechy.com” and “psytechy.pt”, shown in Figure 5.12.

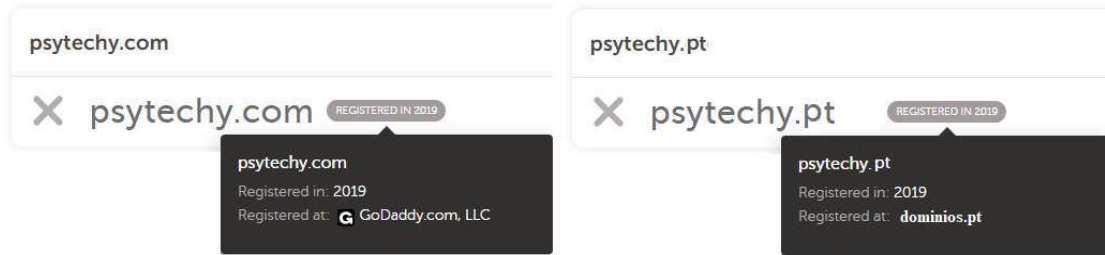


Figure 5.12: Psytechy.com and Psytechy.pt Domain Name Validation Tests

Finally, the connection to the MongoDB Atlas cloud platform was tested by login with a previously registered user from the local website (Figure 5.11, user "thesis test"), which would imply that the database is accessed by both the local and remote servers. The login was performed on the remote server, and Figure 5.13 shows its result.

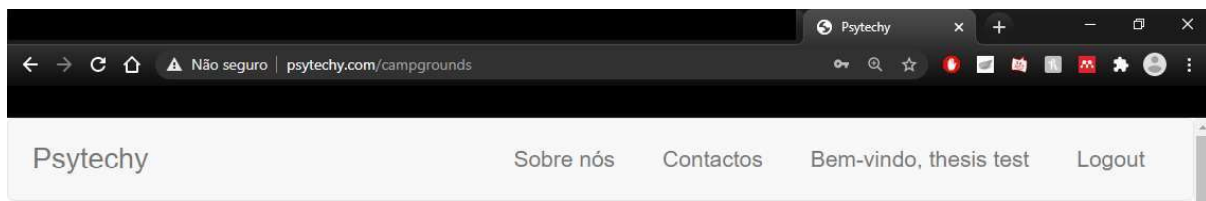


Figure 5.13: Remote Server "thesis test" Login Test

5.2: Results Analysis

The results obtained from the connection and data transmission tests are both satisfying. They serve as a good baseline for the rest of the tests and the dissertation project's overall development.

The hardware connections and the transmission of the two sensors' data might not be the most exciting results to analyze. However, they were entirely crucial and required for the continuation of the project. The tests done to establish a working connection and communication with all the hardware and software used had an overall good result and no errors.

Both heart rate sensors were tested on three people. The tests validated that the Gravity and the VMA340 sensors gathered similar results and did not show any specific differences in their readings. Therefore, only the Gravity tests are shown and analyzed for simplicity, space, and time.

For person A, the results show a resting average reading of 74 BPM and an after-workout average reading of 98 BPM. The resting results demonstrate some BPM variation in the beginning and then stabilize around 75 BPM. The after-workout results were a lot more diverse in the beginning and at the end. Although in the middle, they averaged 100 BPM.

The results showed a similar curve and more stability for persons B and C compared to person A results. The readings stayed stable and had small variations most of the time. Person B showed a resting average reading of 85 BPM and an after-workout average reading of 115 BPM. In comparison, Person C had a resting average reading of 80 BPM and an after-workout average reading of 106 BPM.

Overall, the heart rate readings were successful. Both sensors had similar results and did not represent any differences in their readings. Person A had significant variations and shifts in the readings, which could be a consequence of noise in the sensor's gathering or small movements when testing. Person B and C showed much more stable results. The averages of both resting and after-workout readings seem accurate and realistic as desired.

The local web server and the local website showed positive results, and the features that were tested carried out their desired outcomes. The most challenging hindrance in developing the dissertation's project was transmitting the sensors' heart rate BPM to the local Node.js web server, using the "serialport" component. The test results show that the obstacle is resolved, and the sensor's data is visualized.

The remote web server hosting the remote website results show that the website is always indeed accessible and verify that the domain names are being used in accordance.

Finally, the MongoDB Atlas cloud platform's database tests show that the database can be accessed by both the local and remote web servers. The tests show a new user's activity registering on the local website by connecting to the cloud database and posting it. After, the same user's login is attempted through the remote website by connecting to the cloud database and requesting/authenticating the login. Both the register and login activities are successful, demonstrating positive results.

Chapter 6 : Conclusion and Future Work

6.1: Main Conclusion

This dissertation aimed to create a working online platform that could contribute to psychological assessment and Telehealth help. Web-based methods are getting more attention, and their advantages compared to traditional methods are starting to get more noticeable. A gap was observed in the literature review. There is a lack of guides and research on developing online platforms using wearable sensor data to provide remote healthcare, especially using recent software applications and frameworks, such as Node.js. Telehealth applications have been proved to be useful and a great option to offer healthcare help. Therefore this project was developed to try and help contribute research in that area.

The project's main idea is based on the heart rate sensor's data, heart rate, and heart rate variability values, which would be communicated to the local and remote servers hosting the local and remote websites. The website then processed those values and delivered important information concerning them, such as alerts or reports. The results did not reach that objective. Instead, a solid foundation for the hardware and software communication and connection is developed. This foundation then resulted in the transmission of the sensor's heart rate BPM data to the Node.js web server that visualizes it on the website page.

The hardware components work well together, as was desired, and are all fully compatible with each other. This compatibility was a critical requirement to achieve in order to create a stable groundwork on which the software and the project's future development could stand.

The heart rate PPG technique is explained in detail and how the heart rate sensors communicate with the microcontroller board. The microcontroller board is an essential part of the project's hardware and software communication. A sketch program is uploaded to the board, calculating the sensor's desired heart rate BPM output and sending that data to the Node.js web server.

There is room to grow and improve on the hardware components side of the project, and now that the groundwork is established, new sensors and boards could be tested and implemented in the future.

The project's software side communicates with the microcontroller board, which is transmitting the calculated heart rate BPM data. All the software applications and frameworks communicate accordingly, and the connections to the hardware components work without an issue. The result is the visualization of the sensor's heart rate BPM on the website.

First, the sketch program written with the Arduino IDE software was uploaded to the microcontroller board. The sketch uses an algorithm to calculate the desired heart rate BPM using the sensor's raw data readings. After uploading the sketch, the microcontroller board's connection to the Node.js web server was established. This connection was the most challenging hurdle to overcome in the project's development and the most important. If the Node.js web server did not connect to the microcontroller board and consequently receive the sensor's data, then the whole purpose of the dissertation project's development would come to a stop. This obstacle was eventually surpassed, and the outcome was the sensor's heart rate BPM visualization on the web page. The MongoDB Atlas cloud platform hosting the MongoDB database is connected to both local and remote servers. The remote server hosted by the PaaS Heroku has deployed the website online, using the bought domains (psytechy.com and psytechy.pt).

Although the website receives the sensor's heart rate BPM data, it is only visualized on the webpage. The initial expectation was for the platform to provide useful information in a report or send alerts based on the sensor's data.

This dissertation provides some examples, scenarios, and steps to develop an online platform that provides health and psychological help using wearable sensors. The gap identified in the literature review was the lack of well-documented guides and results in developing a platform such as the one in this dissertation, transmitting wearable sensor's data to a website, especially using recent software applications and frameworks, such as Node.js and MongoDB Atlas.

Hopefully, this dissertation's project and findings contribute to developing similar web-based approach applications and platforms that use Telehealth or wearable sensors. As mentioned in the literature review, there is a growing impact of Telehealth applications contributing to the healthcare industry and our everyday lives. Multiple studies prove the area's usefulness, and new findings being researched and investigated by the hour. It is an area that will impact the world's future, and as new technologies emerge, so will the possibilities for Telehealth.

6.2: Future Work

There are multiple ways to advance the research and development of this dissertation's project. Due to lack of time and the appearance of some obstacles in the project's development, some expectations were not entirely fulfilled.

Although this dissertation delivers stable research findings and a decent description of the development steps and events carried out, these are some of the areas where future work can be applied:

- **Hardware components** – The hardware used in this dissertation's project established a stable foundation for the rest of the development. This foundation was a critical requirement to make sure everything would be operational and synchronized. Consequentially, the hardware used was more superficial, and it did not involve the risk of a more complex and sophisticated alternative. Now that the dissertation's project is completed and all the connections and communications between the hardware and software are functional, new hardware components experiments can be conducted. More complex microcontroller boards can be used, such as the “Arduino UNO Wifi Rev2” [37], which would remove the need for a USB cable and instead establish the communication to the computer through Bluetooth or Wi-Fi, resulting in a more comfortable usage of the heart rate sensor. An Arduino Lilypad [35] board can be experimented with for a more wearable solution approach or even using Raspberry Pi [26] compatible hardware instead of Arduino. Experiments on different heart rate sensors with an alternative heart rate reading technique of PPG can also be examined, such as ECG [24] sensors or chest straps with heart rate monitors.
- **Mobile application** – Just as the website, a mobile application would also be an optimal way of divulging the sensor's data information and alerts. A notification system could be employed, where the user is alerted whenever there is a potential emergency. The smartphone might also have features that allow for activity monitoring or fall detection, which could be implemented in the project and deliver useful healthcare information. The mobile application would be connected to the same MongoDB Atlas cloud database as the local and remote servers hosting the websites, and the sensor's data shared through them all.
- **Website features** – As mentioned before, one of the website expectations was not only to be able to show the sensor's heart rate BPM data but to also provide more detailed information about it, in the form of a report, and create alerts for the user, in case of an

emergency. Other features could also improve the website's functionality and usefulness, such as more detailed psychological assessment and Telehealth services information. Informational videos, interactive games, and questionnaires are possible website features that would make the website more impactful and contribute to the project's development.

Appendix

Heart Rate BPM Algorithm

All the Arduino IDE sketches come with two built-in default functions named `setup` and `loop`:

The setup: The “`setup`” function executes the statements inside it at the beginning of the sketch/program. Doing this initializes variables, input and output pin modes, and other libraries. After the execution is completed, the program will then go on to the `loop` function.

In the `setup` function (Figure A.1) the method “`Serial.begin(9600)`” is going to be called in order to open up the serial port (COM3) and set its data rate, also known as “baudrate”, to 9600 bits per second (bps). With this, the heart rate sensors can now exchange messages with the serial monitor at the established baudrate.

```
void setup(){  
  Serial.begin(9600);  
}
```

Figure A.1: Setup Function

Later, the local Node.js web server will try to connect and listen to the same serial port (COM3) and it will also be necessary to assign the exact matching baudrate of 9600 bps or the web server will not receive any data from the microcontroller board transmitting the sensor’s data. Through that connection the heart rate sensor’s data will be handled in the web server and then visualized online on a website.

The loop: The “`loop`” function will automatically execute after the `setup` function is completed and it will repeat the statements inside itself indefinitely until the program is stopped.

The code written inside the `loop` function reads the raw data sent through the sensor and calculates the associated heart rate in BPM. After calculating the desired heart rate BPM the function then outputs that result. The heart rate BPM in a PPG sensor is calculated by counting the number of systolic peaks and the interval between them per minute. Figure A.2 shows an example of raw data gathered from a PPG sensor and the mentioned systolic peaks necessary to calculate the heart rate BPM output.

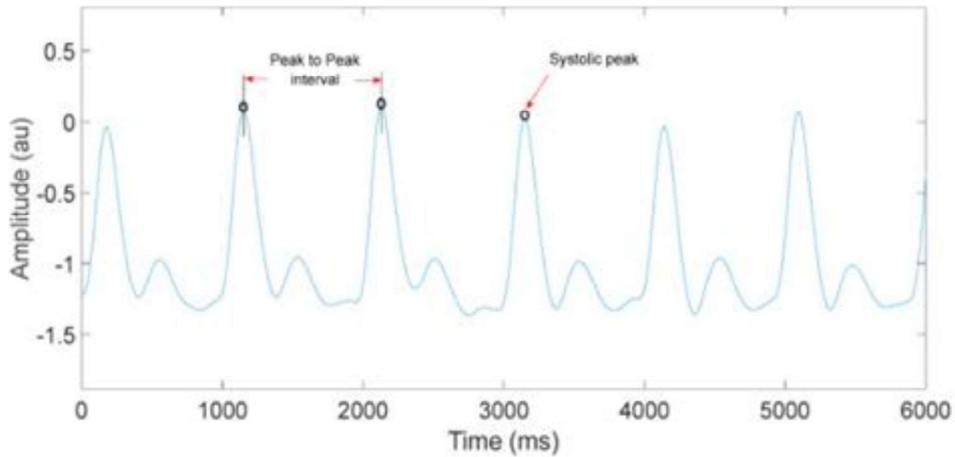


Figure A.2: Example of Raw Data from a PPG Sensor

Knowing the time interval (peak_interval or Peak to Peak Interval) between systolic peaks it is possible to calculate the frequency using the following formula A.1:

$$frequency = \frac{1}{peak_interval} \quad (A.1)$$

In order to get the desired result in BPM all that's left to do is apply the formula A.2, where the frequency is converted from milliseconds to seconds and then to minutes.

$$Heart\ Rate\ BPM = frequency * 60 * 1000 \quad (A.2)$$

The raw data that is being sent from the heart rate sensors used in this project, before being calculated into BPM, can be seen in Figure A.3 (using Arduino IDE). This data, although imperfect, resembles the desired systolic peaks and their intervals which are needed in order to calculate the sensors heart rate BPM.

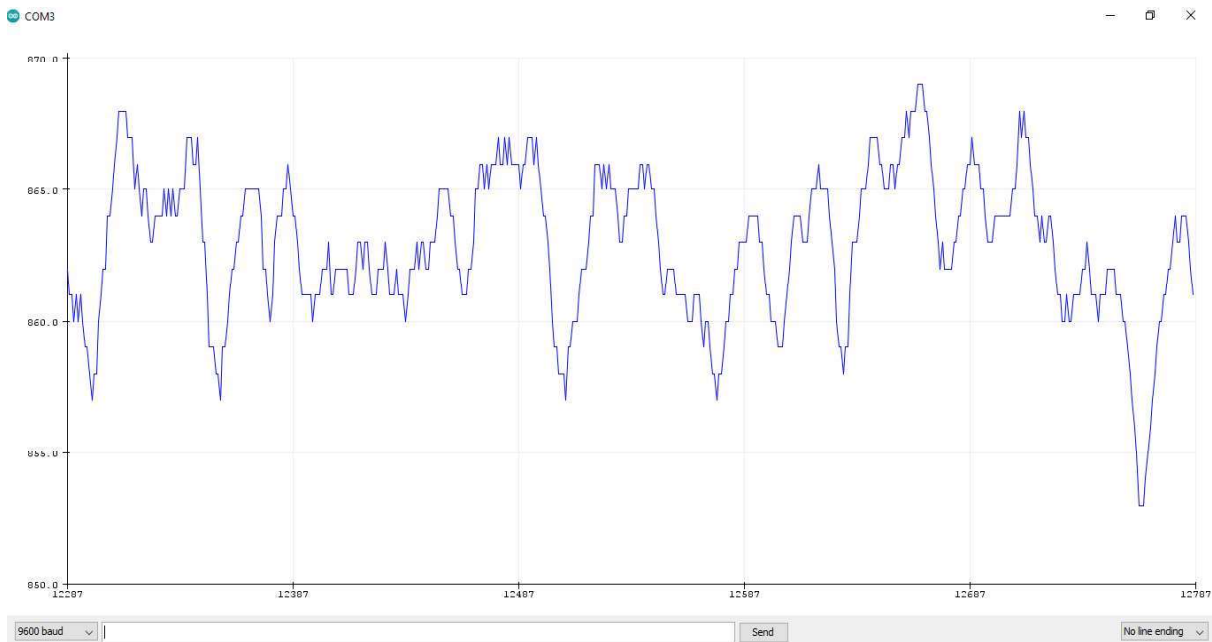


Figure A.3: Project Sensor's Raw Data Gathered from Arduino IDE

The loop function, as mentioned before, is going to be responsible for handling the sensor data and calculating the heart rate BPM result to output. Figure A.4 depicts a flowchart of the algorithm behind the loop function.

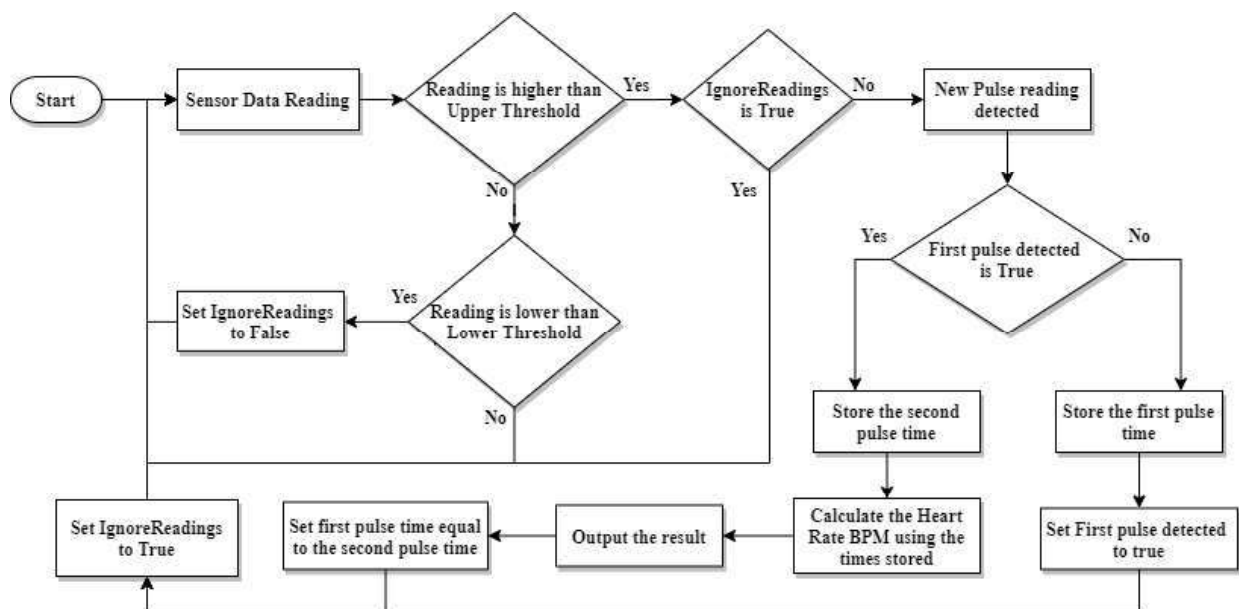


Figure A.4: Flowchart of the Loop Function Algorithm

These are the steps used in the loop function and demonstrated in the flowchart above (Figure A.4), to calculate the heart rate BPM to output:

Step1: To calculate and output the heart rate BPM values the loop function starts by reading and storing the data sent through the sensor inside a variable “dataReading”.

Step2: If the “dataReading” value passes a certain “upperThreshold” (518) and the “ignoreReading” Boolean is false, then it means a pulse was detected and the time when it was detected is stored inside the variable “firstPeakInterval”. After this the Booleans “ignoreReading” and “firstPeakDetected” are both set to true.

Step3: While the “ignoreReading” Boolean is true the loop function will ignore the readings until the “dataReading” value passes a “lowerThreshold” (490), which will set the “ignoreReading” Boolean to false and enable the next pulse detection or peak reading.

Step4: Since the first peak was already detected (“firstPeakDetected” is true), the loop function will repeat the step 2 and this time it will store the time when the peak or pulse was detected inside the variable “secondPeakInterval”.

Step5: The frequency or heart rate is the result of $1 / \text{secondPeakInterval} - \text{firstPeakInterval}$. Finally, the heart rate in BPM is the result of $\text{frequency} * 60 \text{ (minutes)} * 1000$ (convert to seconds), after it is calculated the result is outputted. The firstPeakInterval value is then set equal to the secondPeakInterval and a new secondPeakInterval will be assigned.

Step6: The loop function using the algorithm above will keep running and calculating the heart rate BPM to output until the program is stopped.

Figure A.5 shows the sketch program containing the heart rate sensor’s BPM algorithm code, developed in Arduino IDE.

```

int UpperThreshold = 518;
int LowerThreshold = 490;

void loop(){

    dataReading = analogRead(0);

    // Heart beat reaches upperThreshold meaning a pulse was detected.
    if(dataReading > upperThreshold && ignoreReading == false){
        if(firstPeakDetected == false){
            firstPeakInterval = millis();
            firstPeakDetected = true;
        }
        else{
            secondPeakInterval = millis();
            peakInterval = secondPeakInterval - firstPeakInterval;
            firstPeakInterval = secondPeakInterval;
        }
        ignoreReading = true;
    }

    // Heart beat leaves the upperThreshold and starts waiting for another peak/pulse.
    if(dataReading < lowerThreshold){
        ignoreReading = false;
    }

    BPM = (1.0/peakInterval) * 60.0 * 1000;

    Serial.println(BPM);
}

```

Figure A.5: Heart Rate Sensor's BPM Algorithm Code

Bibliography

- [1] “Applied Sciences | An Open Access Journal from MDPI.” <https://www.mdpi.com/journal/applsci> (accessed Nov. 30, 2020).
- [2] “International Conference on Applied Science and Technology ICAST in January 2021 in Bangkok.” <https://waset.org/applied-science-and-technology-conference-in-january-2021-in-bangkok> (accessed Nov. 30, 2020).
- [3] S. Devaraj and R. Kohli, “Performance Impacts of Information Technology: Is Actual Usage the Missing Link?,” 2003.
- [4] “Information Technology Payoff in the Health-Care Industry: A Longitudinal Study ’ SARV DEVARAJ AND RAHV KOHLI.”
- [5] M. Thouin, J. J. Hoffman, and E. W. Ford Thouin, “The Effect of Information Technology (IT) Investments on Firm-Level Performance in the Healthcare Industry,” 2008.
- [6] D. M. Cutler and M. McClellan, “Is technological change in medicine worth it?,” *Health Aff.*, vol. 20, no. 5, pp. 11–29, Aug. 2001, doi: 10.1377/hlthaff.20.5.11.
- [7] “Heart rate - Wikipedia.” https://en.wikipedia.org/wiki/Heart_rate (accessed Nov. 16, 2020).
- [8] F. S. Routledge, T. S. Campbell, J. A. McFetridge-Durdle, and S. L. Bacon, “Improvements in heart rate variability with exercise therapy,” *Canadian Journal of Cardiology*, vol. 26, no. 6. Pulsus Group Inc., pp. 303–312, 2010, doi: 10.1016/S0828-282X(10)70395-0.
- [9] U. R. Acharya, K. P. Joseph, N. Kannathal, C. M. Lim, and J. S. Suri, “Heart rate variability: A review,” *Medical and Biological Engineering and Computing*, vol. 44, no. 12. Springer, pp. 1031–1051, Dec. 17, 2006, doi: 10.1007/s11517-006-0119-0.
- [10] G. E. Billman, “Heart rate variability - A historical perspective,” *Front. Physiol.*, vol. 2 NOV, 2011, doi: 10.3389/fphys.2011.00086.
- [11] M. Malik and A. J. Camm, “Heart rate variability,” *Clin. Cardiol.*, vol. 13, no. 8, pp. 570–576, Aug. 1990, doi: 10.1002/clc.4960130811.

- [12] M. Javorka, I. Žila, T. Balhárek, and K. Javorka, “Heart rate recovery after exercise: Relations to heart rate variability and coplexity,” *Brazilian Journal of Medical and Biological Research*, vol. 35, no. 8. Associacao Brasileira de Divulgacao Cientifica, pp. 991–1000, Aug. 01, 2002, doi: 10.1590/S0100-879X2002000800018.
- [13] J. P. A. Delaney and D. A. Brodie, “Effects of short-term psychological stress on the time and frequency domains of heart-rate variability,” *Percept. Mot. Skills*, vol. 91, no. 2, pp. 515–524, Oct. 2000, doi: 10.2466/pms.2000.91.2.515.
- [14] C. F. Sharpley, “Heart Rate Reactivity and Variability as Psychophysiological Links Between Stress, Anxiety, Depression, and Cardiovascular Disease: Implications for Health Psychology Interventions,” *Aust. Psychol.*, vol. 37, no. 1, pp. 56–62, Mar. 2002, doi: 10.1080/00050060210001706686.
- [15] V. K. Yeragani *et al.*, “Heart rate variability in patients with major depression,” *Psychiatry Res.*, vol. 37, no. 1, pp. 35–46, 1991, doi: 10.1016/0165-1781(91)90104-W.
- [16] C. B. Taylor, “Depression, heart rate related variables and cardiovascular disease,” *Int. J. Psychophysiol.*, vol. 78, no. 1, pp. 80–88, Oct. 2010, doi: 10.1016/j.ijpsycho.2010.04.006.
- [17] A. Diaz, M. G. Bourassa, M. C. Guertin, and J. C. Tardif, “Long-term prognostic value of resting heart rate in patients with suspected or proven coronary artery disease,” *Eur. Heart J.*, vol. 26, no. 10, pp. 967–974, May 2005, doi: 10.1093/eurheartj/ehi190.
- [18] F. Fatehi and R. Wootton, “Telemedicine, telehealth or e-health? A bibliometric analysis of the trends in the use of these terms,” *Journal of Telemedicine and Telecare*, vol. 18, no. 8. SAGE PublicationsSage UK: London, England, pp. 460–464, Dec. 01, 2012, doi: 10.1258/jtt.2012.GTH108.
- [19] E. R. Dorsey and E. J. Topol, “State of Telehealth,” *N. Engl. J. Med.*, vol. 375, no. 2, pp. 154–161, Jul. 2016, doi: 10.1056/nejmra1601705.
- [20] D. Dias and J. P. S. Cunha, “Wearable health devices—vital sign monitoring, systems and technologies,” *Sensors (Switzerland)*, vol. 18, no. 8. MDPI AG, Aug. 01, 2018, doi: 10.3390/s18082414.
- [21] S. Majumder, T. Mondal, and M. J. Deen, “Wearable sensors for remote health monitoring,” *Sensors (Switzerland)*, vol. 17, no. 1. MDPI AG, Jan. 12, 2017, doi:

10.3390/s17010130.

- [22] E. S. Izmailova, J. A. Wagner, and E. D. Perakslis, “Wearable Devices in Clinical Trials: Hype and Hypothesis,” *Clin. Pharmacol. Ther.*, vol. 104, no. 1, pp. 42–52, Jul. 2018, doi: 10.1002/cpt.966.
- [23] J. Allen, “Photoplethysmography and its application in clinical physiological measurement,” *Physiological Measurement*, vol. 28, no. 3. IOP Publishing, p. R1, Mar. 01, 2007, doi: 10.1088/0967-3334/28/3/R01.
- [24] “(No Title).” <http://www.doctor33.it/cont/download-center-files/17519/cap-electrocardiography-x20968allp1.pdf> (accessed Nov. 17, 2020).
- [25] G. Lu, F. Yang, J. A. Taylor, and J. F. Stein, “A comparison of photoplethysmography and ECG recording to analyse heart rate variability in healthy subjects,” *J. Med. Eng. Technol.*, vol. 33, no. 8, pp. 634–641, Nov. 2009, doi: 10.3109/03091900903150998.
- [26] “What is a Raspberry Pi? | Opensource.com.” <https://opensource.com/resources/raspberry-pi> (accessed Nov. 15, 2020).
- [27] “What are the differences between Raspberry Pi and Arduino?” <https://www.electronicshub.org/raspberry-pi-vs-arduino/> (accessed Nov. 15, 2020).
- [28] “Gravity: Heart Rate Monitor Sensor For Arduino - DFRobot.” <https://www.dfrobot.com/product-1540.html> (accessed Nov. 15, 2020).
- [29] “VMA340: MÓDULO DE SENSOR DE PULSAÇÃO / FREQUÊNCIA CARDÍACA PARA ARDUINO® – Velleman – Wholesaler and developer of electronics.” <https://www.velleman.eu/products/view/?id=450580> (accessed Nov. 15, 2020).
- [30] “DIY Heart Rate Sensor - Arduino Project Hub.” <https://create.arduino.cc/projecthub/Ingeimaks/diy-heart-rate-sensor-a96e89> (accessed Nov. 15, 2020).
- [31] “Gravity: Analog Heart Rate Monitor Sensor (ECG) For Arduino | Varios.” <https://www.ptrobotics.com/sensores-variados/5740-gravity-analog-heart-rate-monitor-sensor-ecg-for-arduino.html> (accessed Nov. 15, 2020).
- [32] “DFRduino UNO R3 - Compatible with Arduino Uno - DFRobot.” <https://www.dfrobot.com/product-838.html> (accessed Nov. 15, 2020).

- [33] G. Gridling and B. Weiss, "Introduction to Microcontrollers," 2007.
- [34] "Various Kinds Of Microcontroller Boards and Its Applications." <https://www.elprocus.com/different-types-of-microcontroller-boards/> (accessed Nov. 18, 2020).
- [35] "Arduino - ArduinoBoardLilyPad." <https://www.arduino.cc/en/Main/ArduinoBoardLilyPad/> (accessed Nov. 15, 2020).
- [36] "Arduino - ArduinoBoardNano." <https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano> (accessed Nov. 15, 2020).
- [37] "ARDUINO UNO WiFi REV2 | Arduino Official Store." <https://store.arduino.cc/arduino-uno-wifi-rev2> (accessed Nov. 15, 2020).
- [38] "DFRobot IO_Expansion_Shield_for_Arduino_V7_SKU_DFR0265." https://wiki.dfrobot.com/IO_Expansion_Shield_for_Arduino_V7_SKU_DFR0265 (accessed Nov. 18, 2020).
- [39] "What is Arduino?" Accessed: Nov. 15, 2020. [Online]. Available: https://www.sparkfun.com/arduino_guide.
- [40] "Built-In Examples | Arduino." <https://www.arduino.cc/en/Tutorial/BuiltInExamples> (accessed Nov. 15, 2020).
- [41] "A professional collaborative platform for embedded development · PlatformIO." <https://platformio.org/> (accessed Nov. 15, 2020).
- [42] "Visual Studio Code - Code Editing. Redefined." <https://code.visualstudio.com/> (accessed Nov. 15, 2020).
- [43] "Arduino Playground - Eclipse." <https://playground.arduino.cc/Code/Eclipse/> (accessed Nov. 15, 2020).
- [44] "Solution stack," *Comput. Deskt. Encycl.*, 2015, Accessed: Nov. 15, 2020. [Online]. Available: <https://encyclopedia2.thefreedictionary.com/Solution+stack>.
- [45] "MongoDB: The Definitive Guide: Powerful and Scalable Data Storage - Kristina Chodorow - Google Livros." <https://books.google.pt/books?hl=pt-PT&lr=&id=uGUKiNkKRJ0C&oi=fnd&pg=PP1&dq=mongodb&ots=hayNfeWyf&si>

- g=tdg1DRWMPre0csVITsY1EpMpMoo&redir_esc=y#v=onepage&q=mongodb&f=false (accessed Nov. 15, 2020).
- [46] “mongoose - npm.” <https://www.npmjs.com/package/mongoose> (accessed Nov. 15, 2020).
- [47] “MongoDB Atlas — MongoDB Atlas.” <https://docs.atlas.mongodb.com/> (accessed Nov. 15, 2020).
- [48] “Express - Node.js web application framework.” <https://expressjs.com/> (accessed Nov. 15, 2020).
- [49] S. Tilkov and S. Vinoski, “Node.js: Using JavaScript to build high-performance network programs,” *IEEE Internet Comput.*, vol. 14, no. 6, pp. 80–83, Nov. 2010, doi: 10.1109/MIC.2010.145.
- [50] “6 Web Development Stacks to Try in 2017 - WebiNerds.” <https://webinerds.com/6-web-development-stacks-try-2017/> (accessed Nov. 15, 2020).
- [51] A. Karanjit, “MEAN vs. LAMP Stack,” 2016. Accessed: Nov. 15, 2020. [Online]. Available: https://repository.stcloudstate.edu/csit_etds/11.
- [52] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django Web Development Done Right*. 2008.
- [53] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: MongoDB vs. MySQL,” Jul. 2015, doi: 10.1109/EMES.2015.7158433.
- [54] “Domain name - Wikipedia.” https://en.wikipedia.org/wiki/Domain_name (accessed Nov. 15, 2020).
- [55] P. V. Mockapetris and K. J. Dunlap, “Development of the Domain Name System,” in *Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM 1988*, Aug. 1988, pp. 123–133, doi: 10.1145/52324.52338.
- [56] “What is top-level Domain | TLD meaning - Namecheap.” <https://www.namecheap.com/domains/what-is-a-tld-definition/> (accessed Nov. 15, 2020).
- [57] “Domain name registrar - Wikipedia.” https://en.wikipedia.org/wiki/Domain_name_registrar (accessed Nov. 15, 2020).

- [58] “GoDaddy - Wikipedia.” <https://en.wikipedia.org/wiki/GoDaddy> (accessed Nov. 15, 2020).
- [59] “Dominios.pt, Alojamento Web e Cloud Server - Dominios.” <https://www.dominios.pt/> (accessed Nov. 15, 2020).
- [60] “Heroku Dev Center.” <https://devcenter.heroku.com/> (accessed Nov. 15, 2020).
- [61] “Heroku Alternatives — Top 5 Picks | by Brenda Clark | Medium.” <https://medium.com/@brenda.clark/heroku-alternatives-top-5-picks-9095cef91d91> (accessed Nov. 15, 2020).
- [62] “Low-code backend to build modern apps | Back4App.” <https://www.back4app.com/> (accessed Nov. 15, 2020).
- [63] “AWS Elastic Beanstalk – Implantar Aplicativos da Web.” <https://aws.amazon.com/pt/elasticbeanstalk/> (accessed Nov. 15, 2020).
- [64] “Plataforma do App Engine para aplicativos | Google Cloud.” <https://cloud.google.com/appengine> (accessed Nov. 15, 2020).
- [65] S. D. Gosling, S. Vazire, and O. P. John, “Should We Trust Web-Based Studies? A Comparative Analysis of Six Preconceptions About Internet Questionnaires,” 2004, doi: 10.1037/0003-066X.59.2.93.
- [66] M. B. Sexton, M. R. Byrd, W. T. O’Donohue, and N. N. Jacobs, “Web-based treatment for infertility-related psychological distress,” *Arch. Womens. Ment. Health*, vol. 13, no. 4, pp. 347–358, Aug. 2010, doi: 10.1007/s00737-009-0142-x.
- [67] G. Alan, “Psychology of Addictive Behaviors Test-Retest Reliability of Alcohol Measures: Is There a Difference Between Internet-Based Assessment and Traditional Methods?,” Miller, 2002. Accessed: Nov. 18, 2020. [Online]. Available: <http://gateway1.ma.ovid.com/ovidweb.cgi>.
- [68] M. J. Gregoski *et al.*, “Development and validation of a smartphone heart rate acquisition application for health promotion and wellness telehealth applications,” *Int. J. Telemed. Appl.*, 2012, doi: 10.1155/2012/696324.
- [69] “Nonin 9560 9560 Bluetooth Finger Pulse Oximeter: Amazon.co.uk: Business, Industry & Science.” <https://www.amazon.co.uk/Nonin-9560-Bluetooth-Finger->

- Oximeter/dp/B019F6MVKG (accessed Nov. 18, 2020).
- [70] H. Gjoreski, A. Rashkovska, S. Kozina, M. Luštrek, and M. Gams, “Telehealth using ECG Sensor and Accelerometer,” 2014.
- [71] N. V. Panicker and A. S. Kumar, “Tablet PC Enabled Body Sensor System for Rural Telehealth Applications,” *Int. J. Telemed. Appl.*, vol. 2016, 2016, doi: 10.1155/2016/5747961.
- [72] “Fitbit heart rate monitoring explained.” <https://www.wareable.com/fitbit/fitbit-heart-rate-monitor-guide-330> (accessed Nov. 18, 2020).
- [73] “Apple Watch heart rate guide: How to use all of Apple’s HR features.” <https://www.wareable.com/apple/apple-watch-heart-rate-monitor-guide-340> (accessed Nov. 18, 2020).
- [74] “Garmin heart rate guide: Features, devices and accuracy.” <https://www.wareable.com/garmin/garmin-heart-rate-monitor-guide-230> (accessed Nov. 18, 2020).
- [75] “Heart rate monitor - Wikipedia.” https://en.wikipedia.org/wiki/Heart_rate_monitor (accessed Nov. 18, 2020).
- [76] “Garmin HRM-Dual™ | Heart Rate Monitor with Chest Strap.” <https://buy.garmin.com/en-US/US/p/649059> (accessed Nov. 18, 2020).
- [77] “Polar H10 | Heart rate monitor chest strap | Polar Global.” https://www.polar.com/en/products/accessories/H10_heart_rate_sensor (accessed Nov. 18, 2020).
- [78] “TICKR Heart Rate Monitor | Wahoo Fitness EU.” <https://eu.wahoofitness.com/devices/heart-rate-monitors/tickr/buy> (accessed Nov. 18, 2020).
- [79] “VitalPatch - VitalConnect.” <https://vitalconnect.com/solutions/vitalpatch/> (accessed Nov. 18, 2020).
- [80] “SerialPort Usage · Node SerialPort.” <https://serialport.io/docs/guide-usage> (accessed Nov. 25, 2020).
- [81] “Introduction | Socket.IO.” <https://socket.io/docs/v3/index.html> (accessed Nov. 25,

2020).

[82] “EJS -- Embedded JavaScript templates.” <https://ejs.co/> (accessed Nov. 25, 2020).

[83] “npm (software) - Wikipedia.” [https://en.wikipedia.org/wiki/Npm_\(software\)](https://en.wikipedia.org/wiki/Npm_(software)) (accessed Nov. 25, 2020).

[84] “folders | npm Docs.” <https://docs.npmjs.com/cli/v6/configuring-npm/folders> (accessed Nov. 25, 2020).

[85] “Understanding the package.json file.” <https://blog.ezekielekunola.com/understanding-the-package.json-file> (accessed Nov. 25, 2020).

[86] “The Heroku CLI | Heroku Dev Center.” <https://devcenter.heroku.com/articles/heroku-cli> (accessed Nov. 25, 2020).