DEVELOPMENT OF REALTIME PATIENT HEALTH TRACKER WEBSITE USING MICROCONTROLLER AND LARAGON

SHAZWAN HAZMI BIN SHAMSHIR

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

2025

DEVELOPMENT OF REALTIME PATIENT HEALTH TRACKER

WEBSITE USING

MICROCONTROLLER AND LARAGON

SHAZWAN HAZMI BIN SHAMSHIR

A project report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Electronics Engineering Technology (Industrial Electronics) with Honours

Faculty of Electronics and Computer Technology and Engineering
Universiti Teknikal Malaysia Melaka

2025

UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI DAN KEJURUTERAAN ELEKTRONIK DAN KOMPUTER

BORANG PENGESAHAN STATUS LAPORAN
PROJEK SARJANA MUDA II

Tajuk Projek : Development of realtime patient health tracker website using Microcontroller and Laragon.

Sesi Pengajian : 2025

Saya SHAZWAN HAZMI BIN SHAMSHIR mengaku membenarkan laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓):

□ **SULIT*** (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

□ **TERHAD*** (Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan.

□ **TIDAK TERHAD**

Disahkan oleh:

(TANDATANGAN PENULIS)

(COP DAN TANDATANGAN PENYELIA)

Alamat Tetap:

Tarikh : 07 February 2025

Tarikh : 10 February 2025

*CATATAN: Jika laporan ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I declare that this project report entitled “Project Title” is the result of my own research except as cited in the references. The project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Student Name : SHAZWAN HAZMI BIN SHAMSHIR

Date : 07 FEBRUARY 2025

APPROVAL

I hereby declare that I have checked this project report and in my opinion, this project report is adequate in terms of scope and quality for the award of the degree of Bachelor of Electrical Engineering Technology with Honours.

Signature :

Supervisor Name : TS. NIZA BINTI MOHD IDRIS

Date : 10/2/25

Signature :

Co-Supervisor :

Name (if any)

Date :

DEDICATION

My parents' unwavering love and support, along with their sacrifices and encouragement, have been the cornerstone of my academic journey. This project is dedicated to them. I am so appreciative of your financial support because it has made it possible for me to pursue my dreams.

Futhermore, I would like to express my sincere gratitude to Ts. Niza Binti Mohd Idris that the nice supervisor. Your direction, tolerance, and unwavering support throughout this project's inception and completion have been crucial to its success. Your guidance has been invaluable, and I sincerely appreciate the knowledge and abilities I have acquired as a result of your instruction.

I also want to express my gratitude to my friends for being my rock and for their friendship, which helped me to overcome the difficulties. Your inspirational support has helped to make this semester at Universiti Teknikal Malaysia Melaka unforgettable. This accomplishment is the result of a team effort as much as mine, and I am appreciative of everyone's priceless contributions.

ABSTRACT

The goal of this research project is to create a web-based platform that combines Laragon and microcontroller technology to provide a comprehensive real-time patient health monitoring solution. The main goal is to develop an intuitive system that can track and monitor patient health parameters continuously and in real-time. The goal of the project is to give medical professionals accurate and timely information about patients' health status by using microcontrollers for data acquisition and transmission and Laragon’s powerful data management features. The process entails designing and implementing the system after doing extensive research to determine the best web development frameworks and microcontroller technologies. Prototyping, testing, and iterative refinement are among the research methods used to make sure the system is secure, dependable, and functional. The creation of a real-time patient health tracker website prototype, which successfully illustrates the viability and effectiveness of the suggested solution, is one of the primary outcomes. The study's findings demonstrate how web development and microcontroller technology can be combined to enhance patient care and healthcare administration. Additional features and functionalities to expand the system's capabilities should be investigated, and usability testing involving medical professionals should be done to gauge the system's applicability in clinical settings.

ABSTRAK

Matlamat projek penyelidikan ini adalah untuk mencipta platform berasaskan web yang menggabungkan Laragon dan teknologi mikropengawal untuk menyediakan penyelesaian pemantauan kesihatan pesakit masa nyata yang komprehensif. Matlamat utama adalah untuk membangunkan sistem intuitif yang boleh menjejak dan memantau parameter kesihatan pesakit secara berterusan dan dalam masa nyata. Matlamat projek adalah untuk memberi profesional perubatan maklumat yang tepat dan tepat pada masanya tentang status kesihatan pesakit dengan menggunakan mikropengawal untuk pemerolehan dan penghantaran data dan ciri pengurusan data yang berkuasa Laragon. Proses tersebut memerlukan mereka bentuk dan melaksanakan sistem selepas melakukan penyelidikan yang meluas untuk menentukan rangka kerja pembangunan web dan teknologi mikropengawal terbaik. Prototaip, ujian dan penghalusan berulang adalah antara kaedah penyelidikan yang digunakan untuk memastikan sistem selamat, boleh dipercayai dan berfungsi. Penciptaan prototaip tapak web pengesan kesihatan pesakit masa nyata, yang berjaya menggambarkan daya maju dan keberkesanan penyelesaian yang dicadangkan, adalah salah satu hasil utama. Penemuan kajian menunjukkan bagaimana pembangunan web dan teknologi mikropengawal boleh digabungkan untuk meningkatkan penjagaan pesakit dan pentadbiran penjagaan kesihatan. Ciri dan fungsi tambahan untuk mengembangkan keupayaan sistem harus disiasat, dan ujian kebolehgunaan yang melibatkan profesional perubatan harus dilakukan untuk mengukur kebolehgunaan sistem dalam tetapan klinikal.

ACKNOWLEDGEMENTS

In order to finish this final year project, I would like to express my sincere gratitude to a number of people who have contributed significantly to its realization. First and foremost, I owe my parents a debt of gratitude for their steadfast financial and emotional support. Their support and selflessness have been the cornerstones of my academic career, and this accomplishment is a direct result of their unwavering faith in my potential.

My supervisor, Ts. Niza Binti Mohd Idris, deserves special recognition for her guidance, knowledge, and patience throughout the project. I gained priceless insights and a successful framework from her mentoring. I am really appreciative of the chance she gave me to develop as a student and learn.

In addition, I would like to thank my peers and friends for their moral support. Because of our encouragement and shared experiences, this semester's challenges were easier to handle.

Furthermore, I would like to express my gratitude to Universiti Teknikal Malaysia Melaka for its resources and support, which were instrumental in the progress and accomplishment of this project and i want to thank everyone who has been mentioned and the larger network of supporters who have helped to make this project possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF DIAGRAMS

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Real-time heart rate tracker is the less manageable it has gotten, the harder it has become to keep track of. Hospitals need to communicate with patients' families in order to treat them. However, patients sometimes forget and afraid to tell doctors vital information about their health in the midst of the chaos that comes with being sick. It can be intimidating to keep track of the people a patient sees and their movements. Especially, if they've ever been sick and have visited multiple doctors as a result. There may occasionally be a line even though a patient is scheduled to see a new specialist on time due to an issue. Either they have realized what had forgotten their records as soon as they sat down with a bunch of forms. Now, they cannot just waste a great deal of time, and the appointment needs to be rescheduled.

Nowadays, the innovation and the introduction of information technology (IT), the healthcare sector is currently one of the largest and fastest-growing industries in the world. The healthcare industry is more productive and appealing thanks to the introduction of IT, which also increases the effectiveness and efficiency of business processes and activities. The needs of individuals and populations are met by interdisciplinary teams of trained professionals and paraprofessionals in the modern healthcare industry, which is divided into several sections.

1.2 Problem Statement

Real-time patient health monitoring outside of clinical settings is limited in traditional healthcare systems, which causes subpar results and delays in intervention. Widespread adoption of current solutions is hampered by their frequent lack of seamless integration, affordability, and accessibility. An innovative strategy is desperately needed to improve patient care and well-being by enabling continuous health tracking and prompt intervention.

1.3 Project Objective

The objective of this project is to revolutionize remote health monitoring by creating a real-time patient health tracker website with Laragon and a microcontroller. The system will make it possible for patient health data to be seamlessly collected, transmitted, stored, and visualized, enabling prompt intervention and individualized treatment.

The objective of this project:

1. To develop a user-friendly interface for interactive real-time data visualization.

2. To produce a system that can monitor health signs, such as blood pressure and temperature.

3. To manage databases for efficient storage of patient health data.

1.4 Scope of Project

The primary objective of the project is to develop a web-based platform with the following crucial components that tracks patients' health in real time:

Hardware Context:

1. Using ESP32, NodeMCU V2 Expansion Board, XD-58D XD58C Pulse sensor, Non-Contact Infrared Temperature Sensor, and Push Button Switch.

2. Choosing assembling microcontroller devices that can transfer and acquire sensor data.

3. Creation of interfaces to link sensors and microcontrollers so that key health indicators can be recorded.

Software Development:

1. The creation and execution of a flexible online interface enabling instantaneous data visualization and interaction.

2. Creating algorithms to validate, preprocess, and transfer data from the microcontroller to the web server.

3. Laragon integration for database administration, encompassing data storage, retrieval, and schema design features.

Data Management:

1. Building a safe database architecture with Laragon to house patient health information. For example, using Laragon and Laravel v10.

2. Putting in place backup plans, access restrictions, and data encryption to guarantee data confidentiality and integrity.

Authentication and User Access:

1. Putting in place user authentication procedures to restrict access to patient data according to roles and authorization.

2. Creation of intuitive user interfaces that make the system easy for patients and healthcare professionals to use.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Real-time patient health tracking systems have become more common in recent years as a result of the integration of technology into healthcare systems. These systems use web-based databases such as Laragon and microcontroller-based sensors to continuously monitor various patient health parameters. The purpose of this review of the literature is to investigate the state of the art regarding the approaches, tools, and results related to these systems. The technological frameworks for choosing microcontrollers and integrating sensors, web-based database management platforms, clinical applications that show promise for patient monitoring and disease management, as well as obstacles and potential future directions for innovation in the field, are some of the main areas of focus. This review contributes to the advancement of technology-enabled healthcare solutions by offering insights for researchers and practitioners involved in the development and implementation of real-time patient health tracking systems, by synthesizing findings from pertinent studies.

2.2 Previous of Related Project

2.2.1 Internet of Things (IoT)

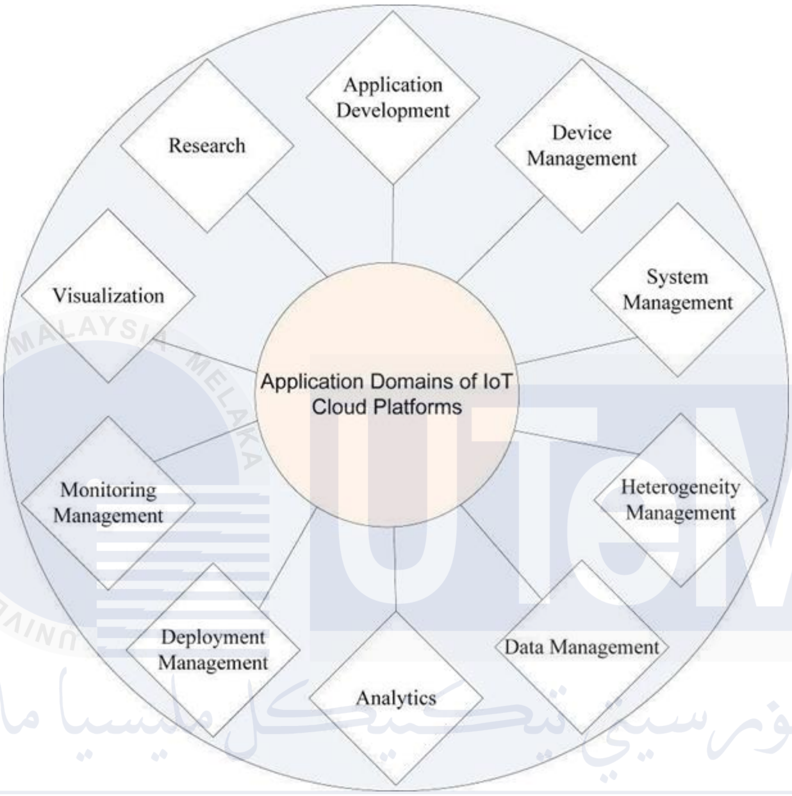


Figure 2.1 Application domains of IoT cloud platforms

According to Figure 2.1, The internet's integration has changed many industries in the modern digital age, including healthcare. Our understanding of healthcare has completely changed as a result of the introduction of Internet of Things (IoT) technology in health monitoring systems [1]. With the widespread usage of the internet, the increasing efficiency of gadgets and devices and the widespread use of the internet, we can now use Internet of Things (IoT)-based health monitoring systems to continuously monitor patients. These gadgets continuously analyze factual information and generate vital signs that can be viewed online from a distance. This makes it possible to monitor patients in real time and to provide quick crisis response services. Therefore, devices powered by the Internet of

Things (IoT) offer emergency response and recognition capabilities, both of which are essential for guaranteeing the best possible patient care.

2.2.2 E Patient Monitoring System using Arduino

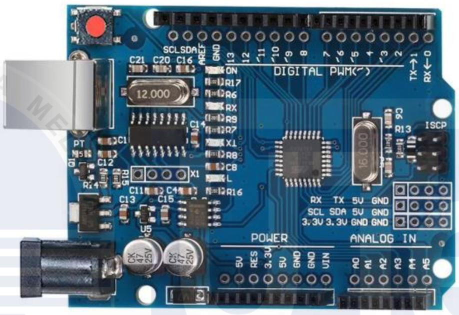


Figure 2.2 The Arduino UNO R3 board.

The authors of this paper discussed the various applications of IoT in healthcare and also detailed the challenging circumstances that IoT in healthcare must overcome. The primary concern with IoT use is security [2]. In the Figure 2.2, the Arduino Uno R3 serves as an interface for a variety of sensors that gather vital signs like temperature and pulse rate. It then processes, filters, and aggregates this data for analysis that makes sense. It then uses communication protocols like HTTP or MQTT to send the processed data to the web server that hosts the patient health tracker website. The Arduino Uno R3 makes it easier to safely store patient data in a MySQL database for effective management and retrieval by integrating with Laragon.

2.2.3 CarePro: A Complete Arduino and Android-based Elderly Care Health and Security Monitoring System

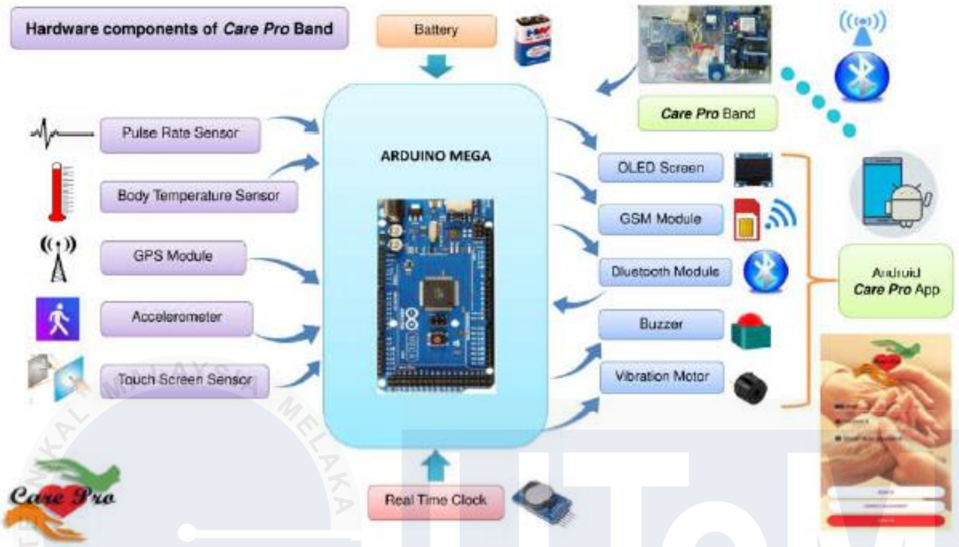


Figure 2.3 Hardware modules in CarePro.

The Arduino Mega is the main hardware module in the CarePro band, coordinating a number of smaller modules to improve its performance. The Arduino Mega is coupled to sub-modules such as body temperature sensors, GPS, accelerometer, GSM, Bluetooth, buzzer, and vibration motor [3]. While the buzzer and vibration motor provide tactile and auditory feedback, the accelerometer detects motion and changes in orientation, the GSM enables communication for alerts and notifications, the Bluetooth technology makes wireless connectivity possible, and the pulse rate and body temperature sensors monitor vital signs that show in Figure 2.3. Futhermore, all of these parts work together to make the CarePro band a complete health monitoring and assistance tool. It can track vital signs, offer location-based services, facilitate communication, and send out timely alerts to ensure the safety and wellbeing of its wearer.

2.2.4 Multi-Parameter Smart Health Monitoring System using Arduino-UNO

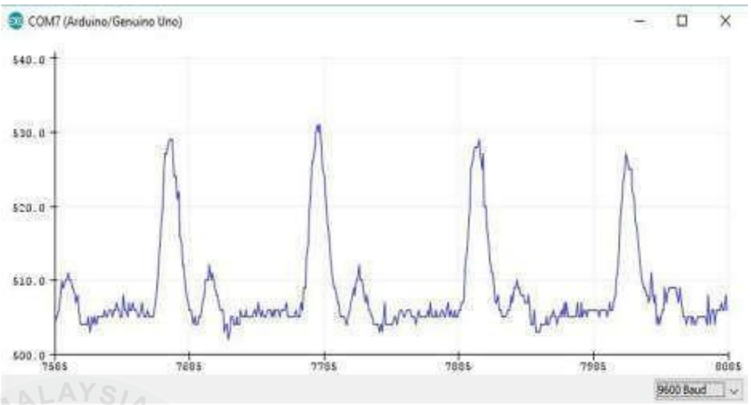


Figure 2.4 Example Pulse Rate.

Patient values were displayed on monitor desktop. Additionally, these same values were added to the Laravel v10 website. We used the EP8266 WiFi module to transfer measured data to the website, thereby making the project IoT based. Through internet connectivity, the sensor measurements were directly displayed on a website [4]. Also, the IoT platform ThingSpeak is essential allows to create a real-time patient health tracker website with Laragon and a microcontroller. ThingSpeak functions as a bridge between the microcontroller and the web server, facilitating the smooth transfer of sensor data from gadgets such as body temperature, pulse rate, and GPS to its cloud-based platform for analysis and storage. ThingSpeak's integrated real-time visualization tools enable the creation of individualized graphs and charts that show important health metrics right on the patient health tracker website. It accomplish thorough data management, analysis, and visualization by utilizing ThingSpeak in conjunction with Laragon. This will enable to develop a strong solution for managing and monitoring patient health in real time. Also, the focus of the health care plan is on measuring and tracking the biological parameters of the patient's body, such as heart rate, blood oxygen saturation level, and temperature, using an Android application and web server. This allows the doctor to continuously check the patient's condition on his smartphone.The patients' collected blood pressure and pulse signals are the primary source of information used by Wang et al.'s intelligent system to operate the heart automatically [5]. By offering thorough monitoring, this seamless integration of technology not only improves patient care but also gives healthcare professionals insightful information for individualized treatment plans and preventative measures.

2.2.5 Design and Implementation of Remote Health Monitoring System using IoT

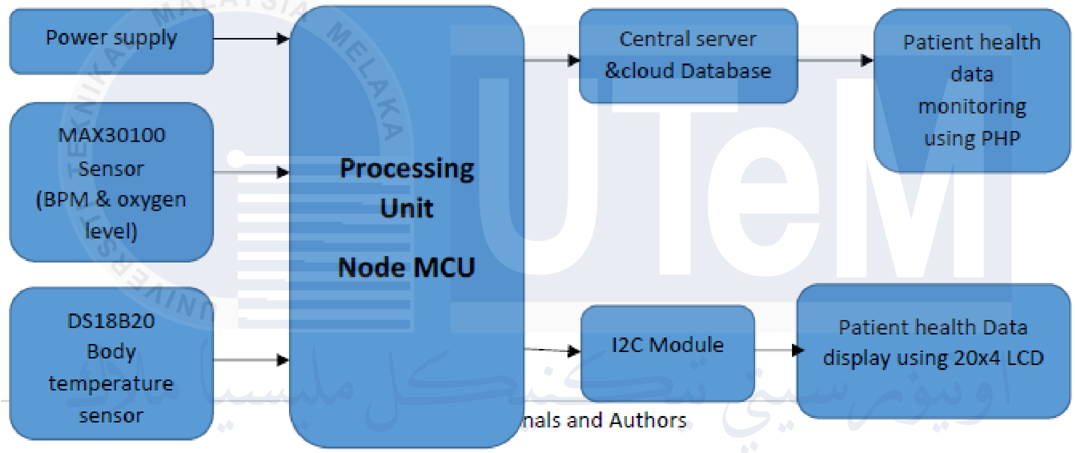

Power supply
MAX30100 Sensor (BPM & oxygen level)
DS18B20 Body temperature sensor
Processing Unit
Node MCU
Central server &cloud Database
Patient health data monitoring using PHP
I2C Module
Patient health Data display using 20x4 LCD
nals and Authors

Figure 2.5 Block Diagram.

By referring to [6], this Figure 2.5 shows the block diagram for remote health monitoring system using IoT. The system as a whole receives its power from the power supply. The max30100, which measures the patient's blood oxygen level (SpO2) and pulse rate, and the DS18B20, which measures the patient's body temperature, are connected to the Node MCU. The processing unit that will gather and handle sensor data is called the Node MCU. After processing, the data will be moved to a web server or cloud server and stored in a database. In the prototype, the LCD 20X4 and Node MCU are connected via an I2C module to enable patient-side health data display. Lastly, a webpage allows a doctor to remotely monitor the data that has been collected.

2.2.6 Development of Smart Healthcare Monitoring System in IoT Environment

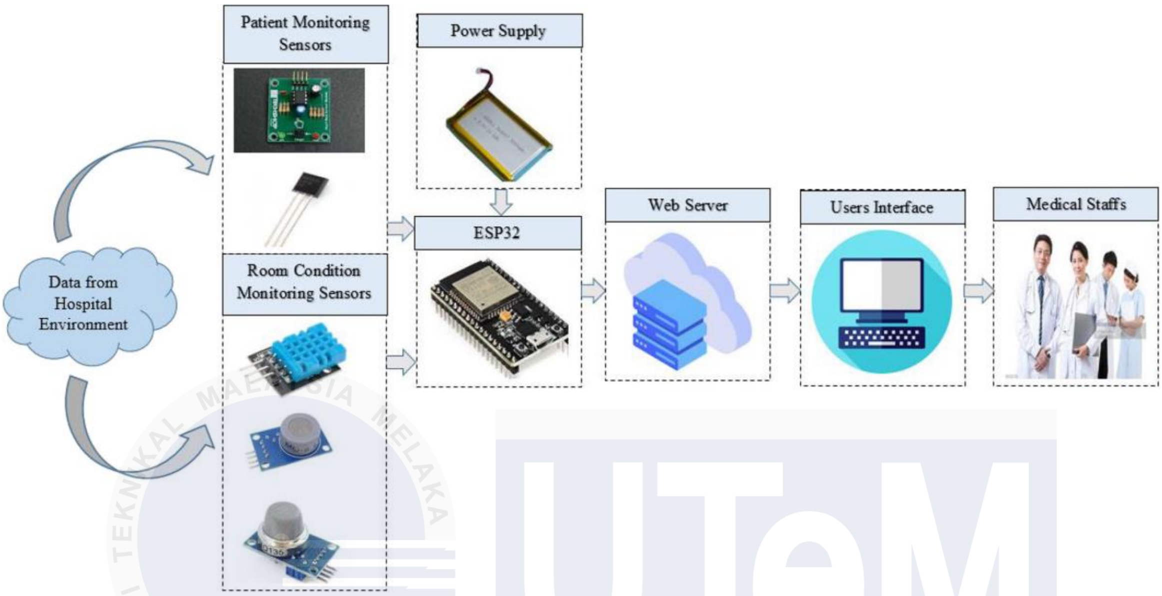

Data from Hospital Environment
Patient Monitoring Sensors
Room Condition Monitoring Sensors
Power Supply
ESP32
Web Server
Users Interface
Medical Staffs

Figure 2.6 System architecture of the healthcare monitoring system.

Referring Figure 2.6, the sensors that are used to gather data from the hospital environment make up the system architecture of the healthcare monitoring system [7]. These sensors are positioned strategically throughout the hospital to continuously monitor various parameters, including equipment status, temperature, humidity, and patient vitals. The collected data is sent in real-time to a central processing unit for analysis and storage via a variety of communication networks, including wired, Bluetooth, Zigbee, and WiFi. The Data Acquisition Layer with sensors, the Communication Network for data transmission, the Data Processing Layer for analysis, the Data Storage Layer for archiving historical records, the User Interface Layer for informing medical professionals, and the Security Layer for safeguarding private health information are the main elements of this architecture. The efficient monitoring and management of the hospital environment made possible by this well-integrated system improves both operational efficiency and patient care.

2.2.7 Real-time Location Tracker for Critical Health Patient using Arduino, GPS Neo6m and GSM Sim800L in Health Care

The Quality of Service (QoS) in terms of communication, routing, energy, and processing should be maintained by the Internet of Things (IoT) based wireless system with a limited number of sensors. Because the quality of service (QoS) deteriorates when a random defective node enters the system [8]. Maintaining Quality of Service (QoS) in an Internet of Things (IoT) based wireless system is crucial to ensuring dependable communication, efficient routing, economical energy use, and timely data processing. Randomly defective nodes can lead to processing bottlenecks, energy depletion, routing inefficiencies, and communication disruptions, which poses a serious threat to quality of service (QoS). To lessen these risks, proactive measures like fault-tolerance mechanisms, dynamic routing algorithms, energy-saving methods, and dependable communication protocols must be implemented. By addressing these problems, IoT-based wireless systems can ensure consistent and efficient performance across a variety of scenarios and applications, thereby maintaining the necessary QoS levels. Different types of sensors are connected to the various human boundaries within this framework body. The GPS area locater communicates with a sequential pin on a microcontroller. Every sensor will examine every one of the body's boundaries and indicators. The sensor will communicate with the microcontroller by sending data based on that body boundary [9]. In addition to improving the granularity of data collection, this distributed sensor network makes it possible to provide individualized healthcare solutions and focused interventions that are catered to each patient's unique needs.

2.2.8 The NIST Definition of Cloud Computing

Software as a Service (SaaS)

By reffering of SaaS, the applications can be accessed via a program interface or a thin client interface, like a web browser (for example, web-based email), from a variety of client devices [10]. With the possible exception of certain user-specific application configuration settings, the has no management or control over the underlying cloud infrastructure, including the network, servers, operating systems, storage, or even the capabilities of individual applications.

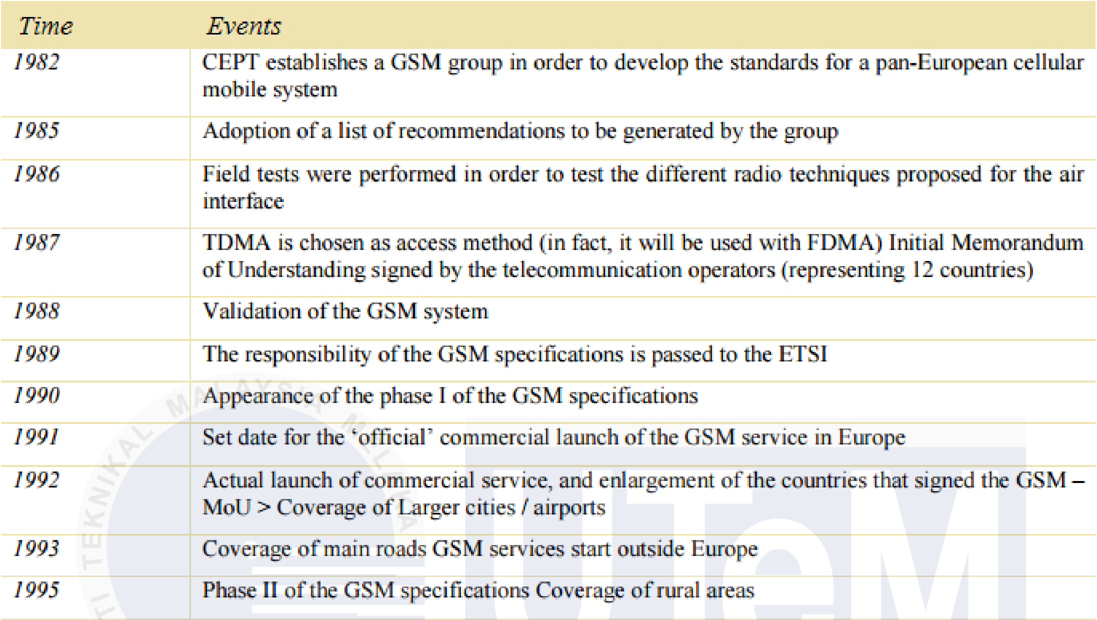
Platform as a Service (PaaS)

An extra abstraction level can be provided by cloud systems: rather than offering a virtualized infrastructure, they can offer the software platform that hosts systems. The amount of hardware resources required to carry out the services is scaled transparently. Platform as a Service is the term for this (PaaS) [11]. Furthermore, PaaS solutions frequently come with integrated features like resource provisioning, load balancing, and automated scaling, which further streamline the development and deployment process while maximizing resource efficiency and cost effectiveness.

Infrastructure as a Service (IaaS)

With this capability, service providers (SPs) can design ad hoc systems that are customized to meet the unique needs of their customers without having to invest in specialized physical infrastructure. IaaS allows SPs to concentrate on effectively deploying and managing software stacks by abstracting away the complexity of hardware management [11].

2.2.9 Developement of Application based Health Monitoring Sytem using GSM module



Time	*Events*
1982	CEPT establishes a GSM group in order to develop the standards for a pan-European cellular mobile system
1985	Adoption of a list of recommendations to be generated by the group
1986	Field tests were performed in order to test the different radio techniques proposed for the air interface
1987	TDMA is chosen as access method (in fact, it will be used with FDMA) Initial Memorandum of Understanding signed by the telecommunication operators (representing 12 countries)
1988	Validation of the GSM system
1989	The responsibility of the GSM specifications is passed to the ETSI
1990	Appearance of the phase I of the GSM specifications
1991	Set date for the 'official' commercial launch of the GSM service in Europe
1992	Actual launch of commercial service, and enlargement of the countries that signed the GSM – MoU > Coverage of Larger cities / airports
1993	Coverage of main roads GSM services start outside Europe
1995	Phase II of the GSM specifications Coverage of rural areas

Table 2.1 Remote health management system architecture.

By refer to Table 2.1, time division multiple access (TDMA) is a digital mobile telephony system that is used by the GSM (Global System for Mobile communication). Of the three digital wireless telephony technologies (TDMA, GSM, and CDMA), it is the most commonly used. GSM digitalizes and data is first compressed before being sent over a channel with two additional user data streams, each running in a separate time slot [12]. The underlying technology that makes this time slot allocation possible is called Time Division Multiple Access (TDMA), which guarantees that users can access the network in a coordinated and orderly fashion. Since it has been shown to be dependable, scalable, and effective at allocating network resources, TDMA especially in the context of GSM implementation has become the most extensively used standard in digital wireless telephony when compared to other technologies like Code Division Multiple Access (CDMA). Futhermore, SIM 908-C is the GSM module that was used in this project. The purpose of this module is to cover the worldwide market. It is paired with an extremely powerful GSM engine. It operates at GSM

850MHz frequency [13]. Else, The project's goal is to create a dependable and stable communication link between the Raspberry Pi device, sensor nodes, and external systems by utilizing the SIM 908-C module. This will allow real-time data transmission and monitoring for healthcare applications. Example pin diagram of GSM Module based on Figure 2.7.

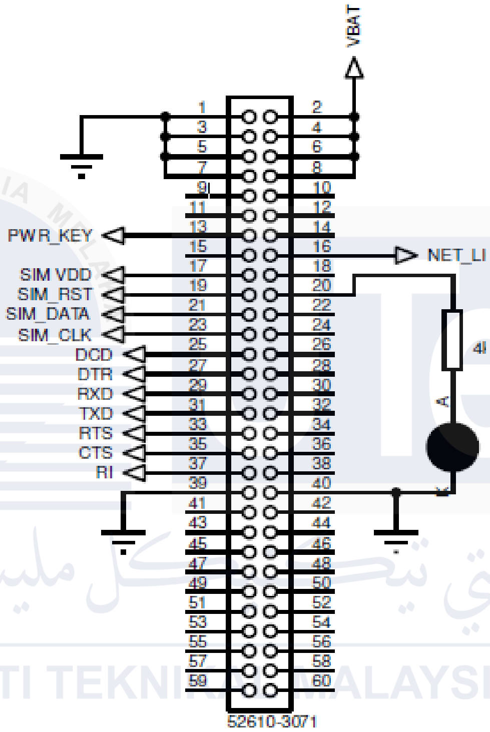


Figure 2.7 Pin Diagram of GSM Module

2.2.10 Implementation of Wireless Body Area Network Based Patient Monitoring

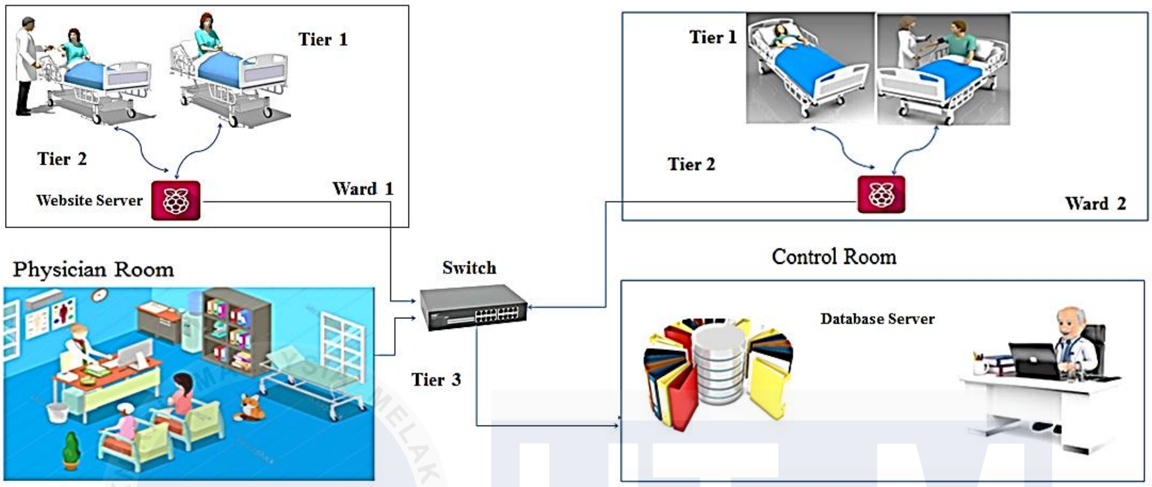


Figure 2.8 Three-tier architecture of the proposed system

By follow the Figure 2.8, the Raspberry Pi (RPi) device functions as the focal point of every ward and the base station for gathering sensor data. The RPi continuously receives messages from the sensor nodes placed throughout the ward thanks to Bluetooth communication technology [14]. Essential components of every message include sensor data (e.g., temperature or pulse rate) and accompanying metadata (e.g., date, time, sensor code, "PR" for pulse rate, "T" for temperature), as well as unique ID numbers for identification. The RPi receives these messages, extracts the data, separates it, and processes it appropriately. After processing, the data is arranged and kept in appropriate tables either in the memory of the Raspberry Pi or on external storage.

2.3 Comparison of previous related projects

Table 2.2 Comparison of Previous related Projects

No.	Reference	Components	Purpose	Advantages	Disadvantages
1.	[1]	Automation Module: -Pulse sensor -Arduino -Temperature sensor Monitoring module: -Laptop -Web server	- To provide early detection of health issues, timely intervention in case of emergency, and enable proactive management of chronic conditions.	-Allows for constant observation and prompt abnormality detection. - Lowers expenses, minimizes in person visits, and streamlines healthcare procedures.	-Data leaks and device failures. -Difficult implementation and management tasks on a technical level.
2.	[3]	-Photodetectors are among the components that make up biometric sensors. -A CPU, memory, input/output ports, and communication interfaces make up a microcontroller. - Processors, memory, network interfaces, and security features are all parts of a gateway.	-The heart rate, blood oxygen saturation, temperature, and other physiological parameters are measured by the sensors, and microcontrollers handle sensor data processing, algorithm execution, and external device communication. - Gateways carry out preprocessing operations, further aggregate sensor data, and help local sensors communicate with distant	-Give vital sign monitoring in real time. -Permit the early identification of anomalies in health. -Assign decision-making and processing powers in real-time. -Allow for flexibility and customization when implementing software. -At the network's edge, enable data filtering	- Patients may find certain sensors bothersome or invasive. -vulnerability to malfunctions in software or hardware. -The expense and intricacy involved in setting up and maintaining gateway infrastructure.

			servers or cloud platforms.	and aggregation. -Ensure interoperability and protocol translation amongst heterogeneous devices.	
3.	[4]	-WIFI Module: Wirelessly transmits patient data to a mobile application and website. -Microcontroller: Handles sensor communication and processes patient data. -Web server: Provides a website on which medical professionals can access patient data. -IoT Platform: Allows data analysis and storing by connecting sensors to a web	-To use IoT technology to continuously monitor patients' health, giving medical professionals access to vital signs like temperature and heart rate. -To provide physicians with access to patient data via a mobile app and web-based platform, allowing for better patient care and prompt intervention.	- Real-time monitoring: Assists physicians in immediately monitoring the health of their patients. -Accessibility: Provides doctors with anytime, anywhere access to patient data.	- Problems with connectivity : Depends on a reliable internet connection, which isn't always available. - Security Concerns: There are privacy and security issues when transmitting patient data online.

		server. -Using a smartphone app, doctors can remotely check on their patients' health.			
4.	[5]	-Consists of a web server or cloud server for data storage, an LCD 20X4 display, a Node MCU for data processing, and sensors such as the Max30100 and DS18B20.	- To use Internet of Things (IoT) technology to remotely monitor patient health, allowing for the continuous tracking of vital signs like body temperature, pulse rate, and blood oxygen level.	- Real-time monitoring gives you immediate access to patient health information so you can take appropriate action. - Accessibility: Enables medical professionals to remotely check on patients' conditions from any place.	-Dependency on Connectivity: Data transmission requires a reliable internet connection, which isn't always available. -Security Concerns: There are privacy and security issues when transmitting patient data online.
5.	[12,14]	- Used in digital mobile telephony through the GSM (Global System for Mobile Communication). - divides time into slots so that several users can	- Allows users to connect to the network in a coordinated and orderly manner. - Guarantees the effective use of network resources. - Establishes a communication channel between external systems, sensor nodes, and	- Scalable and dependable for digital wireless phone systems. - Efficient in assigning resources within the network. - Broadens its market reach due to its global	- Possibly unable to handle heavy traffic volumes. - For certain applications, extra configurations might be needed. -Depends on steady network coverage in order to

		access the network. - Hardware that was employed in the undertaking. - operates at the GSM 850MHz frequency and serves the global market.	Raspberry Pi. - Permits real-time data monitoring and transmission for applications related to healthcare.	compatibility. -Matched with a strong GSM engine to facilitate effective communication.	communicate consistently.

2.4 Summary

After a thorough reading and observation done on the previous related projects, how the rise in real-time patient health tracking systems is a result of the healthcare industry's adoption of technology. With an emphasis on technological frameworks like microcontrollers and sensors, web-based databases like Laragon, and clinical applications, the review seeks to examine the current state of these systems. It also discusses obstacles and potential avenues for innovation in the future. The review aims to advance technology-enabled healthcare solutions by synthesizing findings from pertinent studies and providing insights for researchers and practitioners involved in developing in systems.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The project development and goal-achieving strategy is presented on this chapter. The three sections of this chapter are the hardware and software specification, the study design, and the process flow elaboration. In order to have a clear vision, better understanding of how to handle it, and the best model to use for this project, extensive research on the hardware used was carried out. This will help to ensure that the project runs smoothly. Gaining a general understanding of the project flowchart created is another important goal of this chapter. Following the detailed description of the process flow, the hardware specifications will follow. Lastly, this chapter concludes with a brief discussion and display of the project's connection diagram.

3.2 Methodology

By start by identifying stakeholders and gathering requirements such as real-time health data collection, storage, user authentication, data visualization, and alerts in order to develop a real-time patient health tracker website using a microcontroller and Laragon. The system has been create the architecture using PHP for the backend, MySQL via Laragon for the database, HTML, CSS, and JavaScript for the frontend, and a microcontroller ESP32 for data collection. With configure the microcontroller to read sensor data and transmit it via HTTP requests to the server and create the frontend to offer user interfaces for data visualization and chart libraries manage data submission, validation, and database interactions. Utilizing Laragon to manage the database will guarantee effective indexing and schema design. The unit, integration, and user acceptability testing should be carried out prior to system deployment and server, domain, hosting, and SSL configuration for secure communication. With that, the system is maintained through extensive technical documentation, user manuals, updates, frequent monitoring, and user feedback. Elaboration of Process Flow

3.3 Elaboration of Process Flow

3.3.1 Project Block Diagram

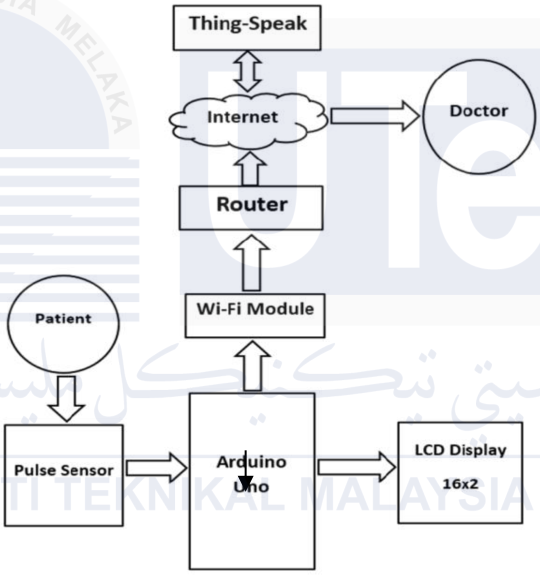


Diagram 3.1 Block Diagram Arduino UNO

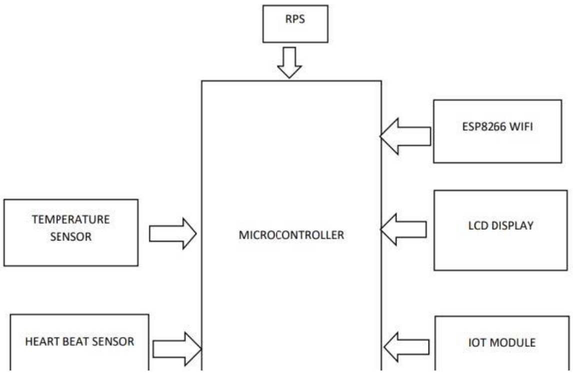


Diagram 3.2 Block Diagram Microcontroller

Based on Diagram 3.1, which talks to the Arduino and transmits data to Thing-Speak. This data on the Thing-Speak is accessible via the internet from any location and is presented in a graph format that includes historical readings as well. The connected LCD displays the BPM as well [15]. The Arduino serves as the main hub, gathering information from multiple sensors that track the vital signs of patients, including heart rate (BPM). The IoT analytics platform ThingSpeak receives this real-time data and uses it to create interactive graphs that show the data in addition to storing it. With the help of these graphs, healthcare professionals can monitor past trends and decide wisely by looking at long-term data patterns. Moreover, by integrating an LCD that is directly connected to the Arduino, vital information is readily available at a glance and instantaneous, on-site feedback is provided by the display of the current BPM. This all-inclusive configuration guarantees easy access to historical and real-time data, making them actionable and improving patient monitoring and care.

3.3.2 Project Flowchart

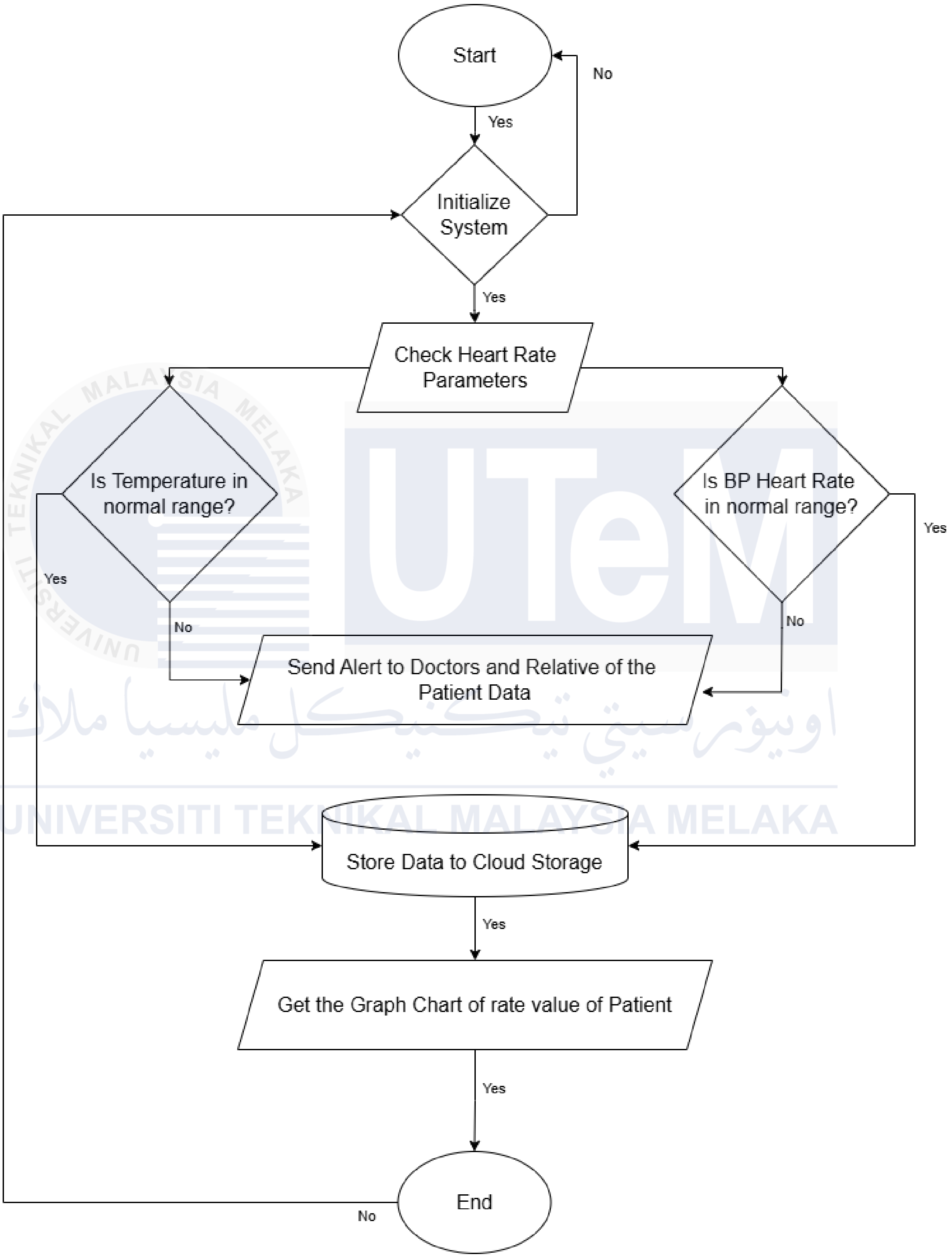

Start
No
Yes
Initialize System
Yes
Check Heart Rate Parameters
Is Temperature in normal range?
Is BP Heart Rate in normal range?
Yes
Yes
No
No
Send Alert to Doctors and Relative of the Patient Data
Store Data to Cloud Storage
Yes
Get the Graph Chart of rate value of Patient
Yes
End
No

Figure 3.1 Flowchart of the system.

Figure 3.3 shows the stages that went into creating this system. First and foremost, the project's required hardware components were carefully chosen. The software platform for coding was selected to be the Arduino Integrated Development Environment (IDE) because of its intuitive interface and capacity to process input signals and produce matching output signals [16]. This choice ensured seamless communication and facilitated the integration of various sensors and modules, such as temperature sensors and wireless communication modules. Additionally, the system's flow chart at the time of system initialization. Ethier examines the health rate parameters and looks for any symptoms that the patient may be experiencing. When it detects any symptoms, it will notify physicians and patients' families along with the patient's location. Furthermore, all of the data you obtain will be saved on the cloud, allowing the server to verify the patient information once more. Following that, the system will display a graph showing the patient's rate value. Additionally, the IDE's ability to interface with other software platforms made functionality possible such as data logging, automated alerts, and remote monitoring and control via IoT platforms. Extensive testing and validation were done to ensure the accuracy and dependability of both hardware and software components as well as the system's consistent performance under a variety of conditions. This meticulous approach, in conjunction with the carefully selected hardware and a powerful software development environment, laid the groundwork for the project's successful implementation and scalability.

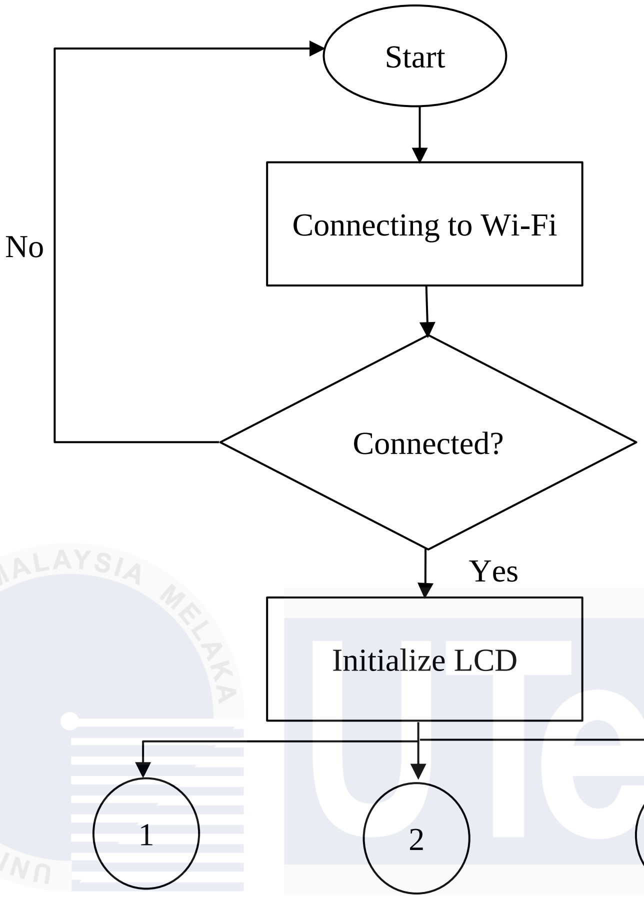


Figure 3.2 Flowchart to Connected Wi-Fi

Based on Figure 3.4, it shows the flowchart for the Wi-Fi connected system using the ESP32 module begins with the initialization of the ESP32 to establish a Wi-Fi connection. Following this, the system checks whether the Wi-Fi connection is successfully established. If the connection is successful, the system proceeds to the next step. It then moves into the main loop, where it gathers information from linked sensors, verifies the accuracy of the data, and looks for situations that might alert users. The prepared processed data is sent over the Wi-Fi connection to a designated endpoint or cloud server for remote monitoring and analysis. The system ensures continuous monitoring and effective operation by entering a wait period after data transmission to conserve power before repeating the main loop.

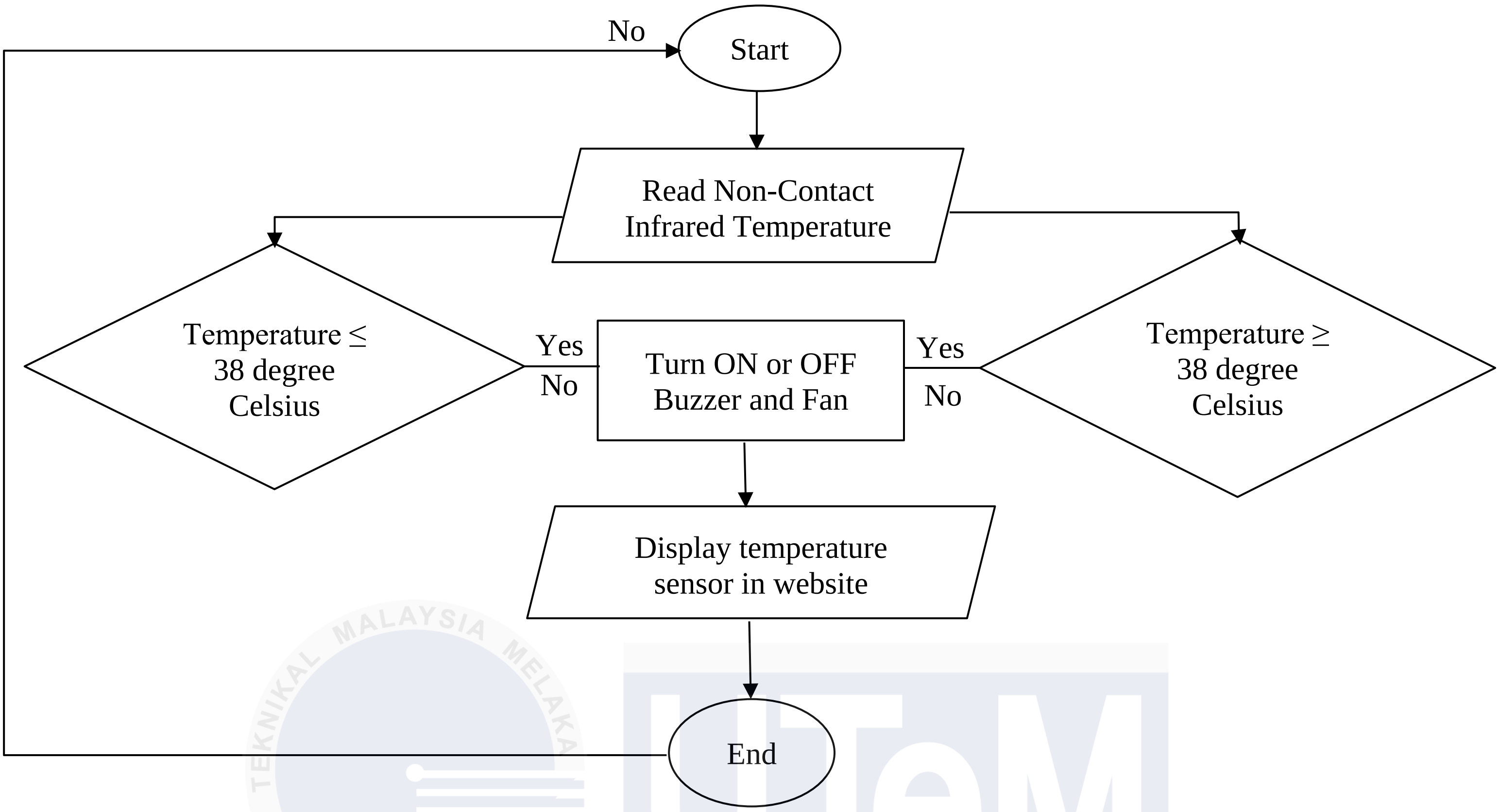

No
Start
Read Non-Contact Infrared Temperature
Temperature ≤ 38 degree Celsius
Yes
No
Turn ON or OFF Buzzer and Fan
Yes
No
Temperature ≥ 38 degree Celsius
Display temperature sensor in website
End

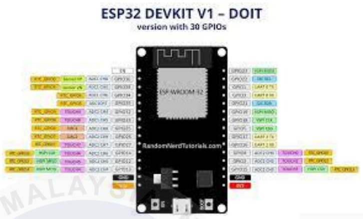
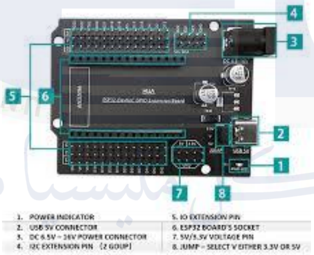
Figure 3.3 Flowchart for Non-Contact Infrared Temperature Sensor

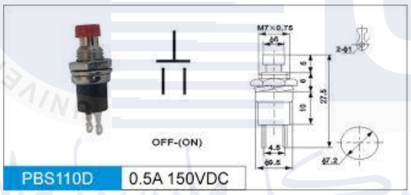
The Non-Contact Infrared Temperature Sensor (NCIT) detects the target's temperature by detecting infrared radiation that is emitted, as shown in Figure 3.5. The sensor records, processes, and displays the data on an LCD after being triggered (by a button or automated event). The system logs the data, resets for the subsequent measurement cycle, and sounds an alert, such as an LED or buzzer, if the temperature rises above predetermined thresholds.

3.4 Equipment Requirements

3.4.1 Hardware Requirement

Table 3.1 List of Components

No.	Components	Description	Quantity
1.	NodeMCU ESP32  ESP32 DEVKIT V1 - DOIT version with 30 GPIOs ESP-WROOM-32 RandomNerdTutorials.com	The NodeMCU-ESP32's breadboard-compatible design, easy programming using Luascript or the Arduino IDE makes comfortable prototyping possible. This board has a BT wireless connection in addition to dual-mode 2.4 GHz WiFi.	1
2.	NodeMCU ESP32 Expansion Board  1 2 3 4 5 6 7 8 1. POWER INDICATOR 2. USB 5V CONNECTOR 3. DC 6.5V ~ 16V POWER CONNECTOR 4. I2C EXTENSION PIN (2 GOUP) 5. IO EXTENSION PIN 6. ESP32 BOARD'S SOCKET 7. 5V/3.3V VOLTAGE PIN 8. JUMP - SELECT V EITHER 3.3V OR 5V	In this breakout pins for connecting to different components, an ESP32 expansion board is a tool that facilitates prototyping projects. Both the narrow and wide versions of the ESP32 development kits are compatible with it.	1
3.	XD58C Pulse Sensor 	Heart rates are measured using the XD-58C pulse sensor module. Creating interactive works related to heart rate is a common application for students, artists, athletes, inventors, game developers, and mobile terminal designers. The sensor can be worn on your finger or earlobe, and Arduino can be connected to it.	1

4.	Temperature Sensor Non-contact MLX90614 Temperature Sensor ICONTECH © 2021 ICONTECH VIN GND SCL SDA	Range of temperature measurements: -70°C to 382.2°C Error in temperature measurement: At room temperature, ±0.5°C 0.02°C resolution Voltage used for operation: 3.3V to 5V The temperature of the surroundings: -40 to 125 degrees Celsius It is simple to install and repair thanks to the M3 fixing screw holes.	1
5.	Push Button Switch  	A push-button switch is a basic electrical part that, when pressed, opens or closes a circuit. There are two primary varieties: normally closed (NC) and normally open (NO). It either completes or breaks the connection when pressed, causing power to be controlled or actions to be triggered.	1

3.4.2 Software Requirement

3.4.2.1 Arduino IDE

Cloud computing and the Internet of Things (IoT) are essential components of the modern telemonitoring health system. This technique uses a microcontroller to collect data from body sensors to monitor the physiological parameters of the patient. The physicians create the patient's health card, which is then posted on a website so that patients and physicians can interact virtually even when they are not in the same room [17]. Together with a design plan that would use IOT-based pulse sensors to produce a heart rate condition, the Arduino IDE software was used to simulate running a system. The device for gathering data was a laptop (PC) or smartphone, Thing Speak, inventor, and Arduino IDE software version 1.8.3 were utilized to analyze sensor output data.

Example code for Arduino IDE:

```
#define samp_siz 4

#define rise_threshold 4



// Pulse Monitor Test Script

int sensorPin = 0;



void setup() {

    Serial.begin(9600);

}
```

```
void loop ()

{

    float reads[samp_siz], sum;

    long int now, ptr;

    float last, reader, start;

    float first, second, third, before, print_value;

    bool rising;

    int rise_count;

    int n;

    long int last_beat;


    for (int i = 0; i < samp_siz; i++)

      reads[i] = 0;

    sum = 0;

    ptr = 0;


    while(1)

    {

      // calculate an average of the sensor

      // during a 20 ms period (this will eliminate

      // the 50 Hz noise caused by electric light

      n = 0;

      start = millis();

      reader = 0.;
```

```
do
{
  reader += analogRead (sensorPin);
  n++;
  now = millis();
}
while (now < start + 20);
reader /= n;  // we got an average


// Add the newest measurement to an array
// and subtract the oldest measurement from the array
// to maintain a sum of last measurements
sum -= reads[ptr];
sum += reader;
reads[ptr] = reader;
last = sum / samp_siz;
// now last holds the average of the values in the array


// check for a rising curve (= a heart beat)
if (last > before)
{
  rise_count++;
  if (!rising && rise_count > rise_threshold)
   {
    // Ok, we have detected a rising curve, which implies a heartbeat.

    // Record the time since last beat, keep track of the two previous

    // times (first, second, third) to get a weighed average.

    // The rising flag prevents us from detecting the same rise more than once.

    rising = true;

    first = millis() - last_beat;

    last_beat = millis();



    // Calculate the weighed average of heartbeat rate

    // according to the three last beats

    print_value = 60000. / (0.4 * first + 0.3 * second + 0.3 * third);



    Serial.print(print_value);

    Serial.print('\n');



    third = second;

    second = first;



   }

  }

  else

  {

   // Ok, the curve is falling

   rising = false;
```

```
      rise_count = 0;

     }

     before = last;

     ptr++;

     ptr %= samp_siz;

   }

}
```

Explanation from the coding:

Using a pulse sensor attached to sensor Pin, this Arduino code is made to track and determine the heartbeat rate. The serial communication is first initialized at a baud rate of 9600. The sensor data is continuously read in the main loop, and noise is reduced by averaging multiple readings over a 20 ms period. In order to make it simple to calculate the current average, these averaged readings are kept in a rolling array of the previous four readings. In order to confirm a heartbeat, the code counts consecutive rising values after searching for a rising edge in the averaged signal. When a heartbeat is detected, a weighted average heart rate is calculated by taking note of the time elapsed since the last beat and calculating the intervals between the last three beats. The serial monitor is then printed with this heart rate. Real-time heart rate monitoring is ensured by the code, which updates and verifies the computed averages and sensor values continuously.

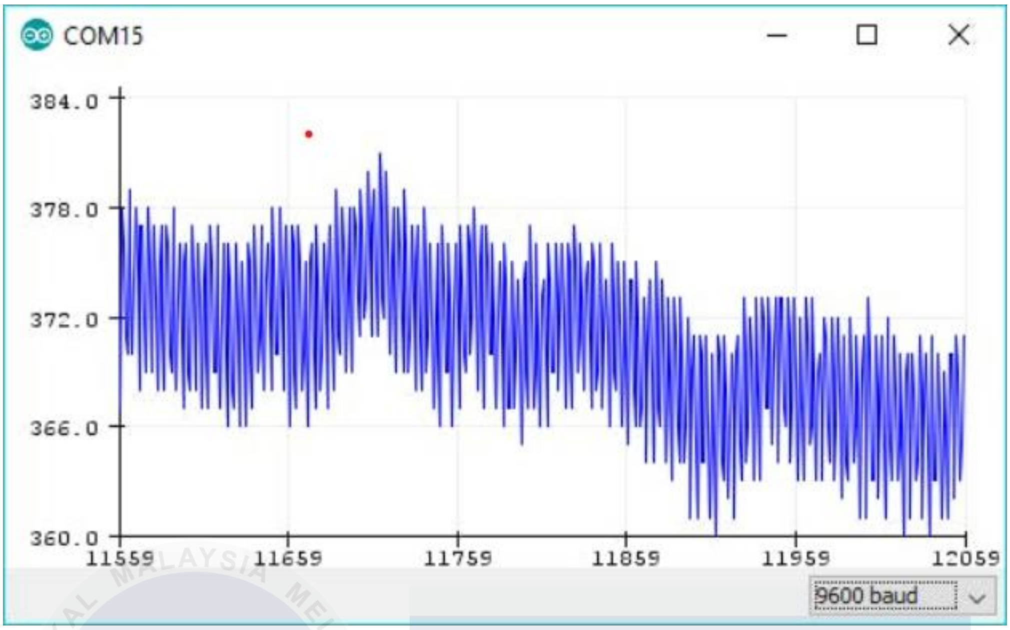


Figure 3.4 Plot with arbitrary analogue value.

Based on Figure 3.4 show entails gathering real-time data from an analogue sensor, using a microcontroller to convert it to a digital format, and presenting the data graphically so that it can be examined. Real-time data collection from the pulse sensor was sent to the IoT analytics platform Thing Speak, where it was processed, stored, and visualized. With this configuration, heart rate data could be continuously monitored and analysed. The successful monitoring and analysis of heart rate conditions made possible by the integration of these tools showcases the potential of IoT in health monitoring applications. The visualization of trends and patterns in sensor data depends on this process.

3.4.2.2 Laragon Laravel V2

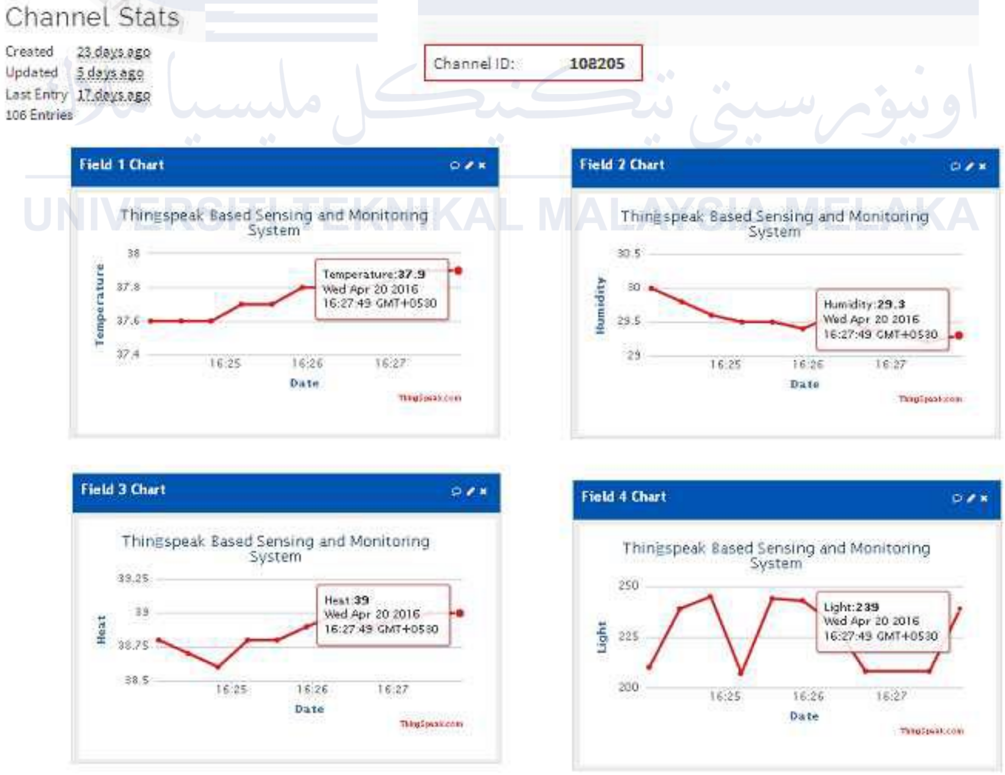
The multi-layer architecture of this Laravel-based web application improves scalability, maintainability, and security by keeping concerns separate between the presentation, business logic, and data layers. The business logic layer receives requests from the presentation layer, which manages the user interface and interactions, and uses controllers and services to process them in accordance with business rules [18]. Database interactions are abstracted in the data layer by Eloquent ORM, which is part of Laravel. In addition to managing HTTP requests and providing resources, the web server serves as the client's "proxy" and communicates with the database by sending and receiving queries. Complex web applications can be easily maintained and have reliable performance thanks to this configuration, which offers dynamic, real-time, and interactive customer information services. Based on table given, there are the structure layered of Web application framework.

Table 3.2 The layered structure table of Web application framework based on Laravel Framework.

First layer	Second layer	Third layer
Laravel core component extension	Laravel label extension	Batch Query for Analysis page
Laravel framework	Persistence layer of database	Common data persistence framework

3.4.2.3 Thing Speak

Thing Speak is an open-source Internet of Things application and Application Programme Interface (API) that uses a Local Area Network (LAN) or the Hyper Text Transfer Protocol (HTTP) on the internet to store and retrieve databases from objects. Thing Speak triggers the creation of detector logs for an application, places applications for identification, and creates a social network of things with current conditions [19]. Enabling users of Thing Speak to assess and visualize uploaded databases through the use of the software tool. The whole collection of Thing Speak documents has been set up to be located in Math Works, and it has even been made possible to register Math Works users' accounts with the correct login and password on the Thing Speak website [20] . Based on the Figure 3.5, it shows that the output results work on the program.



Channel Stats

Created 23 days ago
Updated 5 days ago
Last Entry 17 days ago
106 Entries

Channel ID: 108205

Figure 3.5 Thing Speak IoT Cloud Output results.

3.5 Summary

This chapter describes the whole process of using Laragon and a microcontroller to create a real-time patient health tracker website. As part of the methodology, stakeholders are identified and requirements for the collection, storage, user authentication, data visualization, and alerting of real-time health data are gathered. The system architecture makes use of an ESP32 microcontroller for data collection, HTML, CSS, and JavaScript for the frontend, MySQL via Laragon for the database, and PHP for the backend. Accurate data collection and transmission are made possible by the hardware, which includes parts like the DS18B20 temperature sensor, Node MCU ESP32, Node MCU V2 Expansion Board, Arduino UNO R3, and XD58C pulse sensor. Real-time data visualization and analysis are made possible by Thing Speak; while prototyping and troubleshooting are supported by software tools like the Arduino IDE. A reliable system for managing health-related data with secure online access and patient authentication is made possible by the integration of PHP and SQL by Laragon. The system's dependability is ensured by thorough testing, validation, and secure server configurations, which makes it a useful tool for tracking and evaluating patient health metrics instantly.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

The "Development of Real-time Patient Health Tracker Website Using Microcontroller and Laragon" offers a thorough description of how hardware and software are integrated in the chapter on implementation and development. Three sections make up this chapter: Data Analysis, Hardware Development, and Software Development. The project's goal is to develop a comprehensive system that can track patients' vital signs in real time and present the data on an easy-to-use web interface. The device uses the capabilities of microcontrollers to gather information from a variety of patient-affixed sensors, including blood pressure, temperature, and heart rate monitors. After that, this data is sent to a web server that has been configured with Laragon, where it is processed, saved, and made available to healthcare providers.

In the Software Development section, critical patient health metrics can receive real-time updates and alerts thanks to the application's highly interactive and responsive design. The choice and setup of microcontrollers and sensors, which guarantee precise and trustworthy data collection, are covered in the Hardware Development section. Lastly, the Data Analysis section covers how the gathered data was processed and interpreted, including how algorithms were put into place to find anomalies and produce reports. With this all-encompassing approach, the system is guaranteed to track patient health metrics in real-time and to offer insightful data that can facilitate prompt medical interventions.

4.2 Results and Analysis

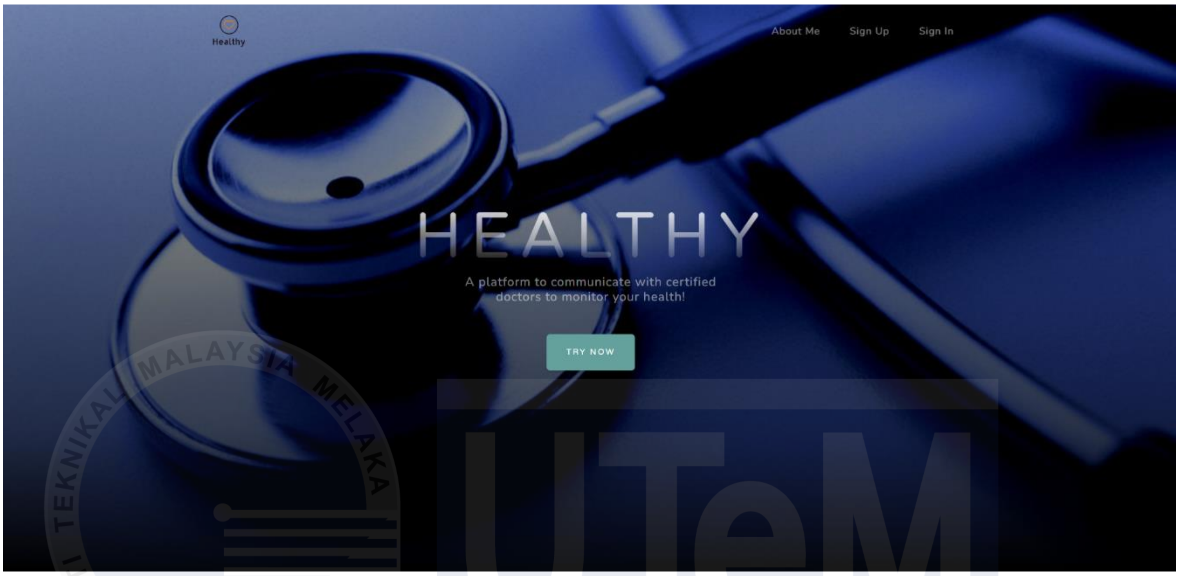
As the result, this project's build show that how outcomes a web-based platform combining Laragon and microcontroller technology can be developed step-by-step for real-time patient health monitoring. The prototype successfully illustrates its potential to offer a complete health monitoring solution, stressing both its advantages and shortcomings prior to full implementation.

In order to ensure dependable and real-time performance, the analysis determined either the most accurate or not configuration for tracking patient health data and body temperature. The user-friendly interface made it possible to visualize data in real time with ease. Furthermore, the system effectively managed patient health data in a structured database while monitoring vital health indicators like heart rate and temperature.

These findings support the suggested methodology and show that using Laragon and microcontroller technology together is a practical strategy for applications involving real-time health monitoring. The efficacy and accuracy of the classification models are demonstrated by a thorough analysis of their performance, which also offers insights into potential areas for additional optimization to improve system reliability.

4.2.1 Design of Software

4.2.1.1 Landing Page

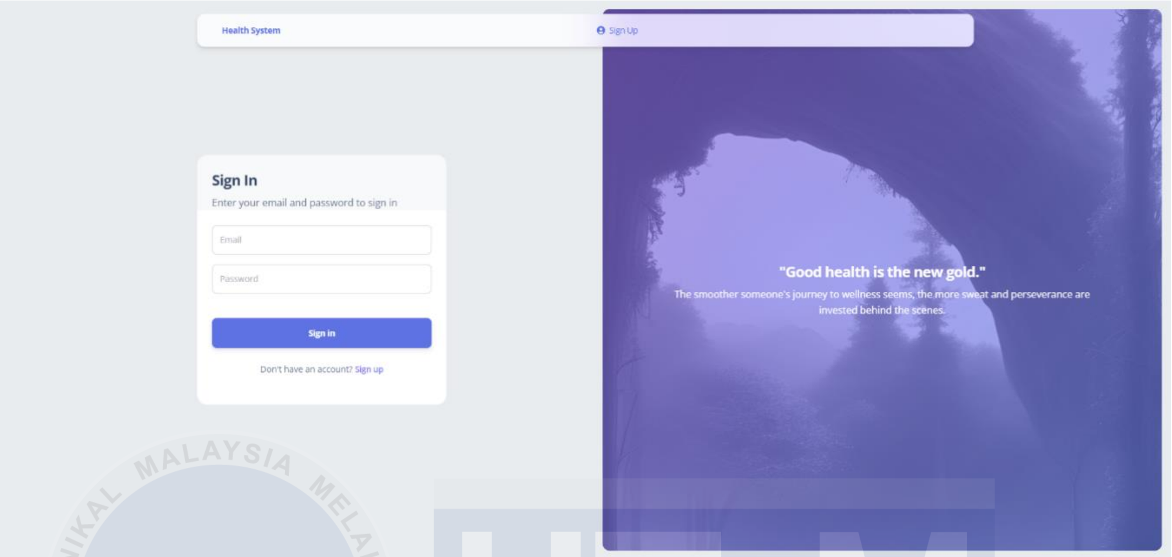


A platform to communicate with certified doctors to monitor your health!

Figure 4.1 Landing Page

To create a unified and useful interface, the "Healthy" healthcare platform incorporates important design elements such as architectural systems, styles, displays, and materials [21]. The close-up of a stethoscope in Figure 4.1 represents health monitoring, and the serene blue background communicates dependability and professionalism qualities crucial to healthcare services. The tagline, "A platform to communicate with certified doctors to monitor your health!" emphasizes the platform's goal of enabling real-time communication between patients and medical professionals, while the platform's name, "HEALTHY," is presented in a contemporary white font for clarity and brand emphasis. The "Healthy" platform successfully strikes a balance between functional requirements and a user-friendly design to foster trust and usability. It does this by adhering to best practices like simplicity, clear messaging, and professional aesthetics when compared to similar designs covered in related articles.

4.2.1.2 Login and Register Page Layout



Sign In

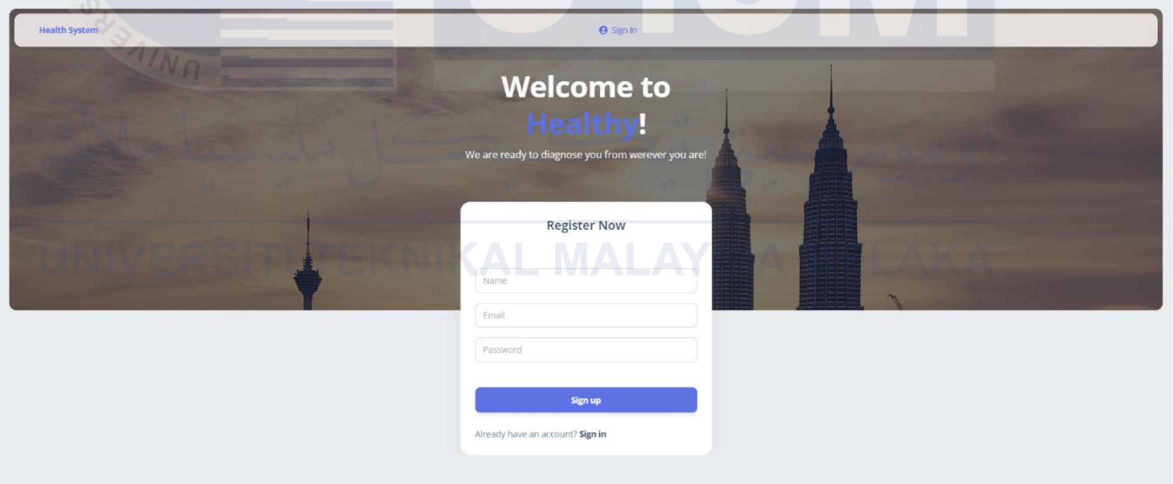
Enter your email and password to sign in

Don't have an account? Sign up

"Good health is the new gold."

The smoother someone's journey to wellness seems, the more sweat and perseverance are invested behind the scenes.

Figure 4.2 Login Page



Welcome to Healthy!

We are ready to diagnose you from werever you are!

Register Now

Already have an account? Sign in

Figure 4.3 Registration Page

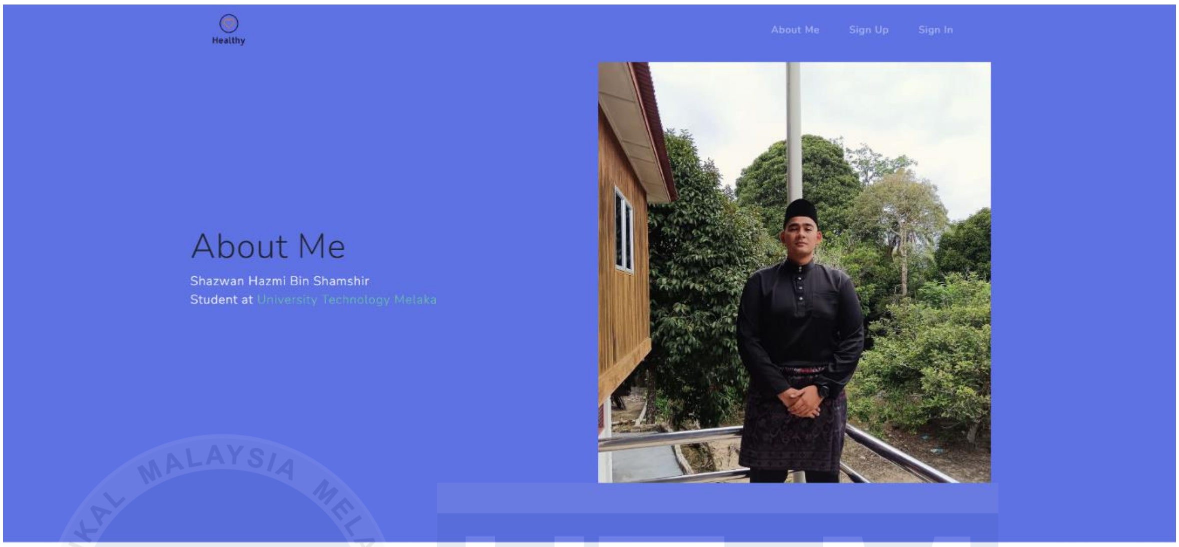
The patient health tracker system's user authentication and registration procedure are made to guarantee safe access while offering an easy-to-use interface. Users are taken to a registration page on their first visit, where they must enter their email address, password, and full name, among other necessary details which that show at Figure 4.3 and Figure 4,2.

New users can quickly register and access the system's features thanks to this simplified approach, which makes navigation simple.

Laragon's incorporation into this authentication procedure improves efficiency and security. Laragon uses strong security protocols to safeguard user credentials, making sure that private data is encrypted prior to storage. In particular, the user's password is protected from unwanted access by being encrypted using Laragon's built-in features. This degree of security is essential, especially for applications in the healthcare industry where patient data confidentiality is critical. Compared to other web-based authentication systems, Laragon provides clear benefits. Due to their reliance on conventional database management systems that lack built-in security features, many traditional web applications may leave user data vulnerable. On the other hand, Laragon's all-inclusive framework strengthens the authentication procedure with cutting-edge encryption methods while also streamlining it. Because users can be sure that their information is being handled with the highest care, this leads to a more secure and effective user experience.

Additionally, the patient health tracker system's use of Shared Preferences for data storage makes key-value pair management effective and facilitates rapid user data access. In contrast, other web applications might use more sophisticated data storage solutions, which could result in slower performance and more complicated data retrieval. All things considered, the patient health tracker system is positioned as a dependable and effective platform for managing patient health information thanks to Laragon's security features and the simple registration process. The benefits of utilizing Laragon over alternative web-based solutions are emphasized in this analysis, especially with regard to user experience, efficiency, and security.

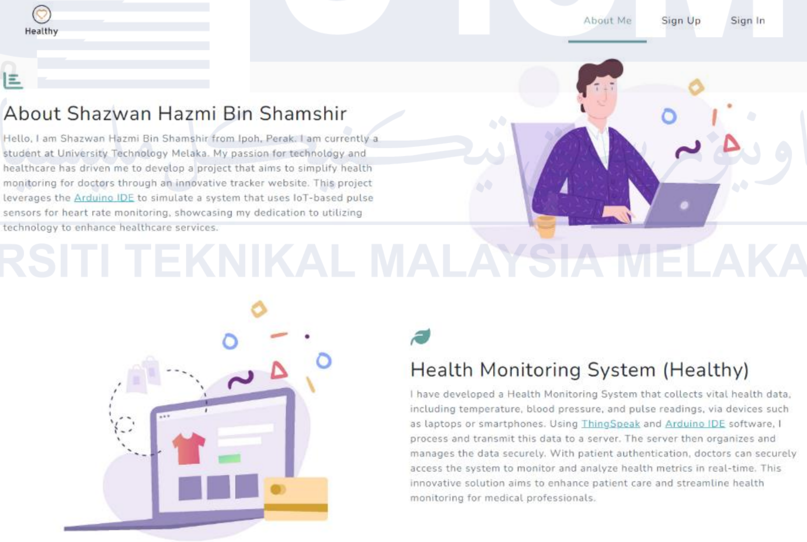
4.2.1.3 About Me Page



About Me

Shazwan Hazmi Bin Shamshir
Student at University Technology Melaka

Figure 4.4 Front Page About Me



About Shazwan Hazmi Bin Shamshir

Hello, I am Shazwan Hazmi Bin Shamshir from Ipoh, Perak. I am currently a student at University Technology Melaka. My passion for technology and healthcare has driven me to develop a project that aims to simplify health monitoring for doctors through an innovative tracker website. This project leverages the Arduino IDE to simulate a system that uses IoT-based pulse sensors for heart rate monitoring, showcasing my dedication to utilizing technology to enhance healthcare services.

Health Monitoring System (Healthy)

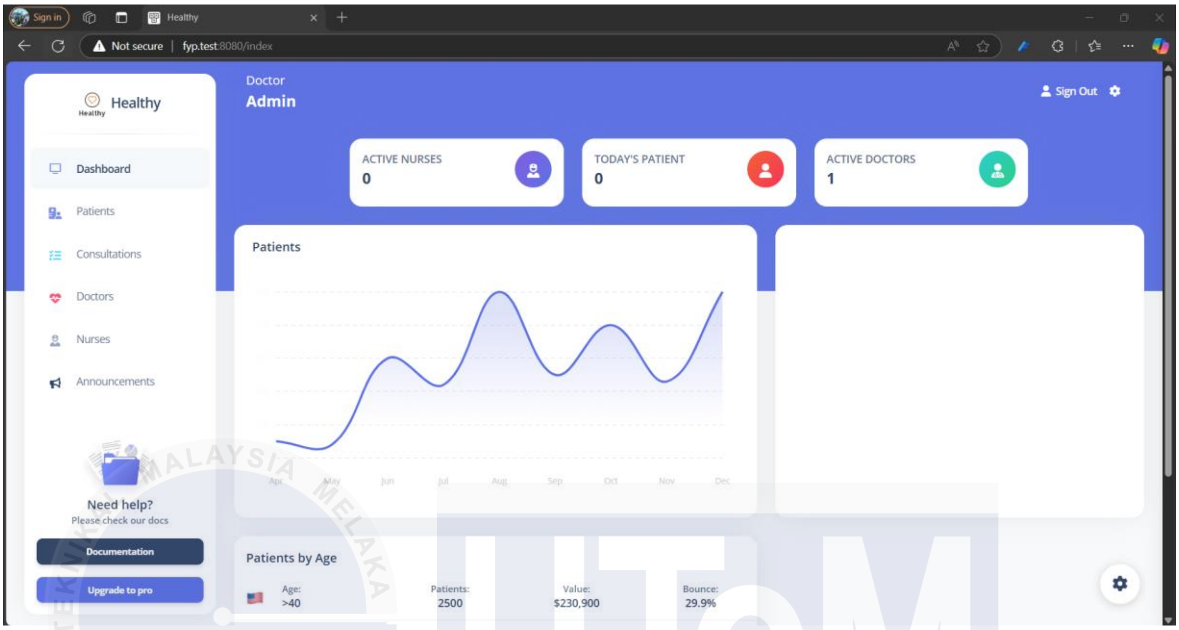
I have developed a Health Monitoring System that collects vital health data, including temperature, blood pressure, and pulse readings, via devices such as laptops or smartphones. Using ThingSpeak and Arduino IDE software, I process and transmit this data to a server. The server then organizes and manages the data securely. With patient authentication, doctors can securely access the system to monitor and analyze health metrics in real-time. This innovative solution aims to enhance patient care and streamline health monitoring for medical professionals.

Figure 4.5 The Detail About Me

As seen in Figure 4.4, the health monitoring system's profile page offers an aesthetically pleasing and intuitive user interface. A straightforward white header that prominently displays the title "About Me" and the user's current status as a University Technology Melaka student contrast with the design's blue backdrop. In addition to improving readability, this design decision makes the site feel more inviting to users. The system's emphasis on health and wellness is further reinforced by the addition of a heart icon with the label "Healthy" in the upper left corner. Furthermore, the image of a person dressed in traditional Malay clothing on the right side of the page, surrounded by a wooden house and verdant surroundings, exudes friendliness and approachability. With a cartoon image of a man using a laptop to further emphasize the digital aspect of the health monitoring system, the website successfully conveys the project's goals and highlights the integration of technology and healthcare.

This platform's user-centric design and integration of Internet of Things (IoT) technologies set it apart from other web-based health monitoring systems. The absence of a visually appealing interface in many conventional health monitoring systems may make it more difficult for users to interact and become involved. On the other hand, this system's use of a vivid color scheme and relatable imagery creates a more welcoming user experience. Furthermore, this health monitoring system is positioned as a progressive solution due to the integration of contemporary data collection and processing technologies. This platform's use of IoT technologies enables more effective data handling and real-time analytics, whereas other systems might rely on traditional data management techniques. All things considered, the examination of the profile page and the features of the health monitoring system shows a well-designed platform that puts the user experience first and uses contemporary technology to enhance patient outcomes, potentially making it a useful tool for increasing healthcare efficiency.

4.2.1.4 Admin Dashboard Page



ACTIVE NURSES 0

TODAY'S PATIENT 0

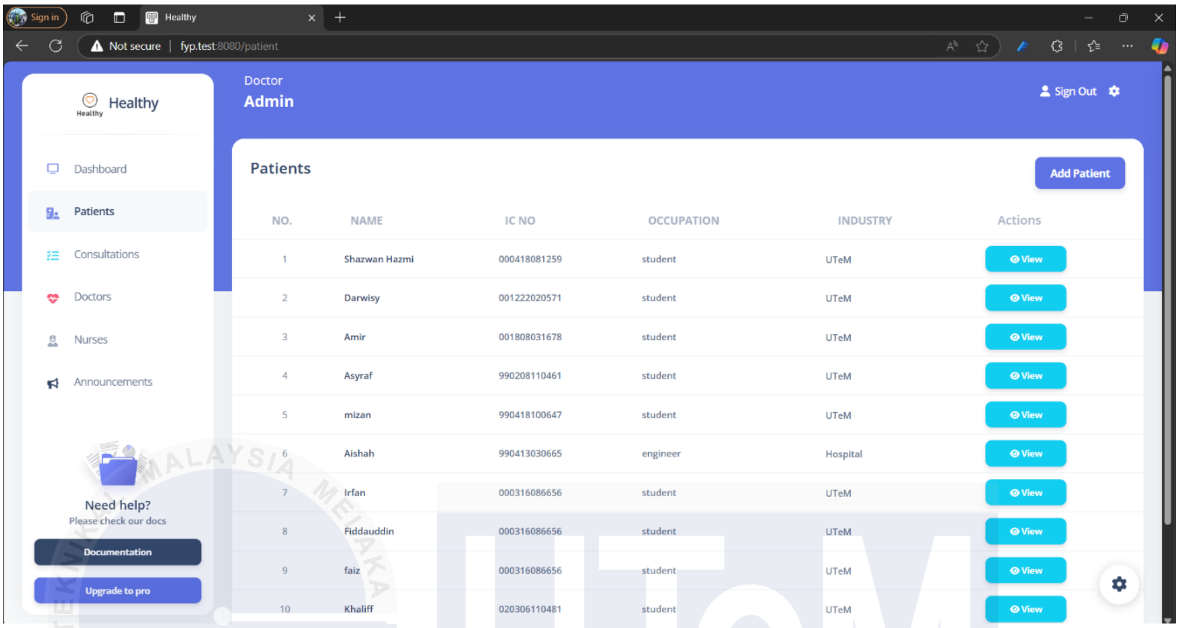
ACTIVE DOCTORS 1

Age:	Patients:	Value:	Bounce:
>40	2500	$230,900	29.9%

Figure 4.6 Admin Dashboard Page

In the Figure 4.6, web application dashboard that was probably created for a hospital or medical office and provides important metrics and data points for physicians and nurses to monitor patient data, consultations, and other pertinent activities. A chart showing patient trends over time, a distribution of patients by age group, indicators for the number of active nurses and doctors, and the total number of patients seen today are among the features.

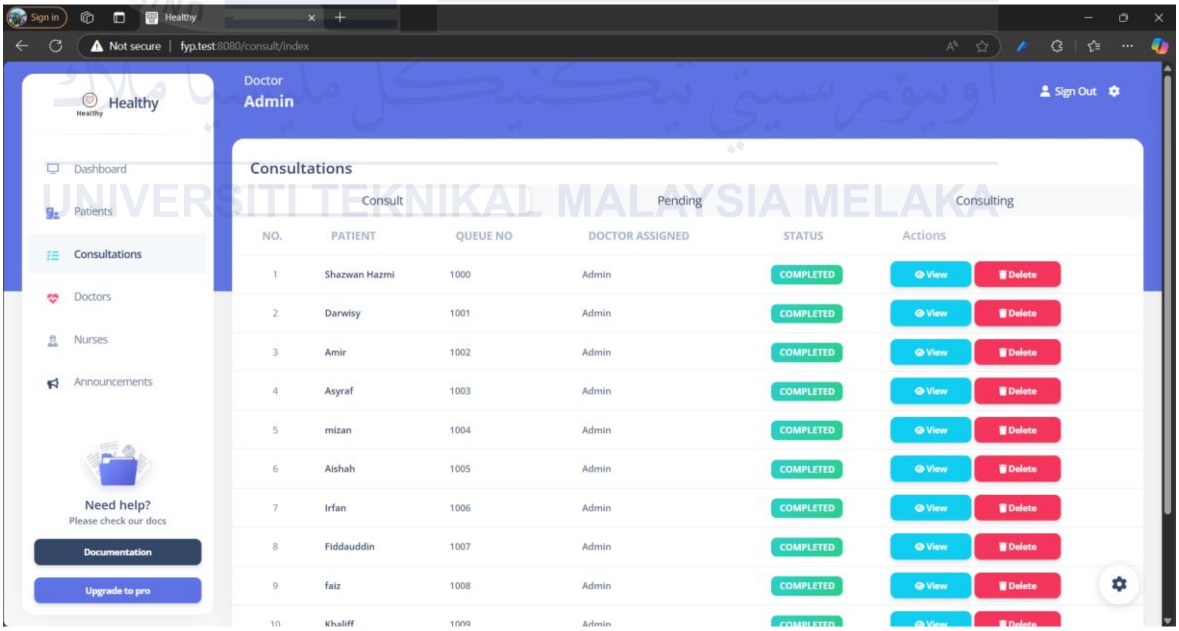
4.2.1.5 Patients and Consult Page



Patients

NO.	NAME	IC NO	OCCUPATION	INDUSTRY	Actions
1	Shazwan Hazmi	000418081259	student	UTeM	View
2	Darwisy	001222020571	student	UTeM	View
3	Amir	001808031678	student	UTeM	View
4	Asyraf	990208110461	student	UTeM	View
5	mizan	990418100647	student	UTeM	View
6	Aishah	990413030665	engineer	Hospital	View
7	Irfan	000316086656	student	UTeM	View
8	Fiddauddin	000316086656	student	UTeM	View
9	faiz	000316086656	student	UTeM	View
10	Khaliff	020306110481	student	UTeM	View

Figure 4.7 Patients Page



Consultations

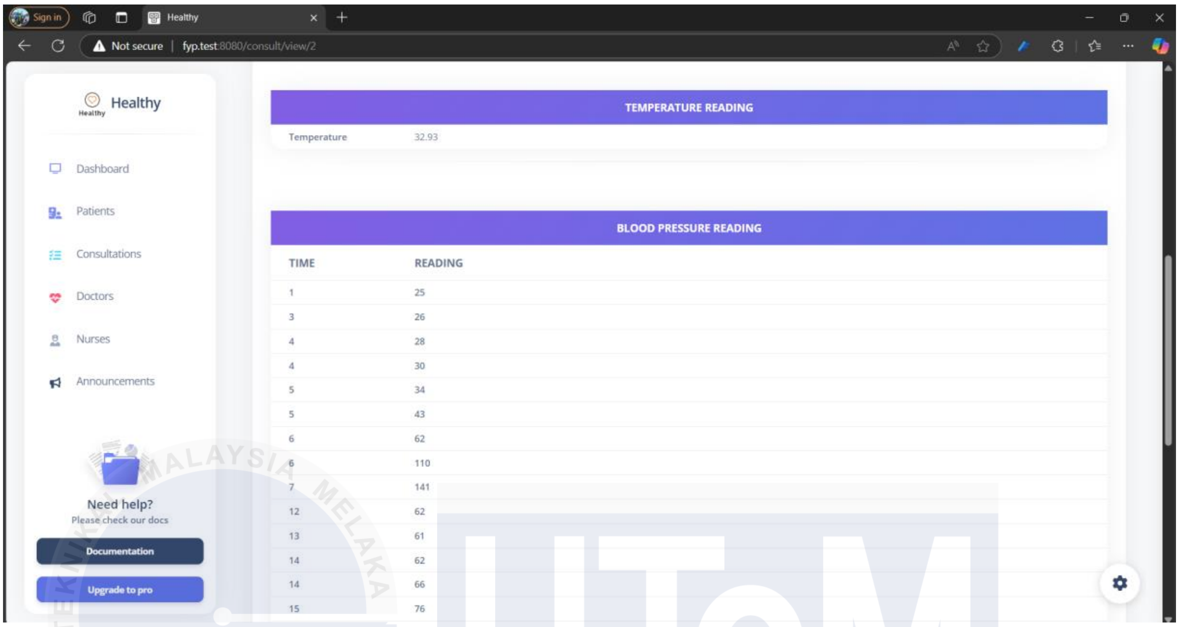
NO.	PATIENT	QUEUE NO	DOCTOR ASSIGNED	STATUS	Actions
1	Shazwan Hazmi	1000	Admin	COMPLETED	View Delete
2	Darwisy	1001	Admin	COMPLETED	View Delete
3	Amir	1002	Admin	COMPLETED	View Delete
4	Asyraf	1003	Admin	COMPLETED	View Delete
5	mizan	1004	Admin	COMPLETED	View Delete
6	Aishah	1005	Admin	COMPLETED	View Delete
7	Irfan	1006	Admin	COMPLETED	View Delete
8	Fiddauddin	1007	Admin	COMPLETED	View Delete
9	faiz	1008	Admin	COMPLETED	View Delete
10	Khaliff	1009	Admin	COMPLETED	View Delete

Figure 4.8 Patients Consult Page

As seen in Figure 4.7, the patient management system has a well-organized web application interface that makes it easier to manage patient records and consultations effectively. With key navigation options like "Dashboard," "Patients," "Consultations," "Doctors," "Nurses," and "Profile," the sidebar makes it simple for users to access various application sections. The primary section presents an extensive table with patient data, such as "No.," "Name," "IC No.," "Occupation," "Industry," and "Actions." This design suggests that managing patient records, viewing consultations, and possibly making appointments are the main functions of the application. Furthermore, usability is improved by the inclusion of a search function and a button for adding new patients, which makes it simple for medical professionals to effectively manage patient data.

This application's user-friendly interface and extensive functionality set it apart from other web-based patient management systems. An interface that further helps doctors efficiently manage and view patient consultations is shown in Figure 4.8. In addition to a table with important data like consultation ID, patient name, queue position, assigned doctor, and current status, it has a dashboard that shows the number of ongoing consultations. Physicians can rapidly prioritize tasks and evaluate their workload with this level of detail. Additionally, the application makes patient profiles, doctor schedule management, and nurse-related data accessible, facilitating smooth provider coordination. This patient management system unifies necessary tools in a single interface, enabling doctors to effectively update consultation statuses, take notes, and schedule follow-up appointments unlike many traditional systems that might have limited functionality or require multiple logins to access different features. Overall, the analysis shows how this patient management system is a useful tool for improving healthcare delivery because of its benefits in terms of functionality, integration, and user experience.

4.2.1.6 Reading Information



TEMPERATURE READING

Temperature	32.93

BLOOD PRESSURE READING

TIME	READING
1	25
3	26
4	28
4	30
5	34
5	43
6	62
6	110
7	141
12	62
13	61
14	62
14	66
15	76

Figure 4.9 Reading Information

As the given Figure 4.9 Label "Healthy," the web application that is visible in the picture seems to be a system for keeping track of patient medical records. With an emphasis on patient-specific data, including blood pressure and temperature readings, the interface offers an easy-to-understand layout. Medical practitioners can monitor changes over time because blood pressure readings are displayed in a tabular format and the temperature is expressed as a numeric value. The "Healthy" application exhibits simplicity and ease of use in contrast to a more feature-rich system, like "MedTrack," but it is devoid of sophisticated functionalities. For example, "MedTrack" provides dynamic visualizations like graphs and trends, which make it simpler for medical professionals to interpret data over time, whereas "Healthy" displays patient data in static formats. Both systems have fields for diagnosing and tracking patient symptoms, but "MedTrack" improves on these features by storing patient history and sending out medication reminders [22].

Table 4.1 Functional Comparison

Feature	**Healthy**	**MedTrack**
Temperature Display	Yes	Yes
Blood Pressure Data	Tabular Format	Tabular and Graphical options
Medication Info	Basic, Text-based	Advanced, with reminders
Diagnosis Status	Stated explicitly	Includes additional history
Search Functionality	Basic	Advanced, with filters
Multi-User Support	Doctors and Nurses	Doctors, Nurses, and Admins

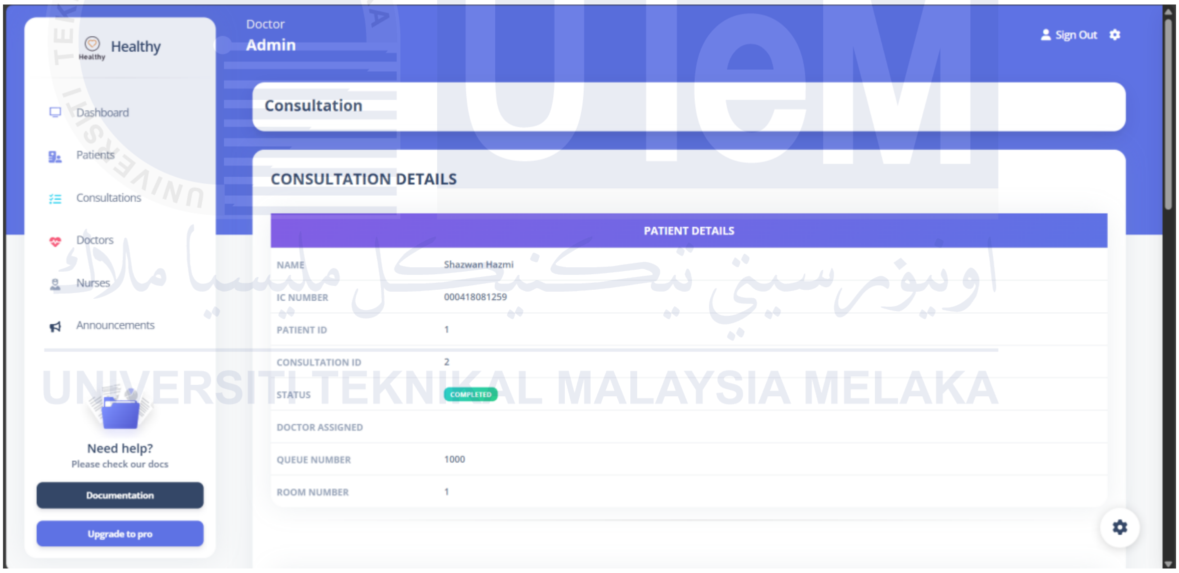
Table 4.2 User Interface Comparison

Aspect	**Healthy**	**MedTrack**
Layout	Minimalistic and structured	Advanced, with analytics panels
Visual Elements	Simple charts and tables	Graphs, heatmaps, and charts
Customization	Limited	Highly customizable
Device Responsiveness	Moderate	Fully responsive

Table 4.3 Usability Comparison

Aspect	**Healthy**	**MedTrack**
Ease of use	Beginner-friendly	Advanced, with analytics panels
Documentation	Basic, linked in the sidebar	Graphs, heatmaps, and charts
Integration Options	Unknown	Supports third-party
Data Export	API	CSV, PDF, and API

4.2.1.7 Details of Consultation Patient



Consultation

CONSULTATION DETAILS

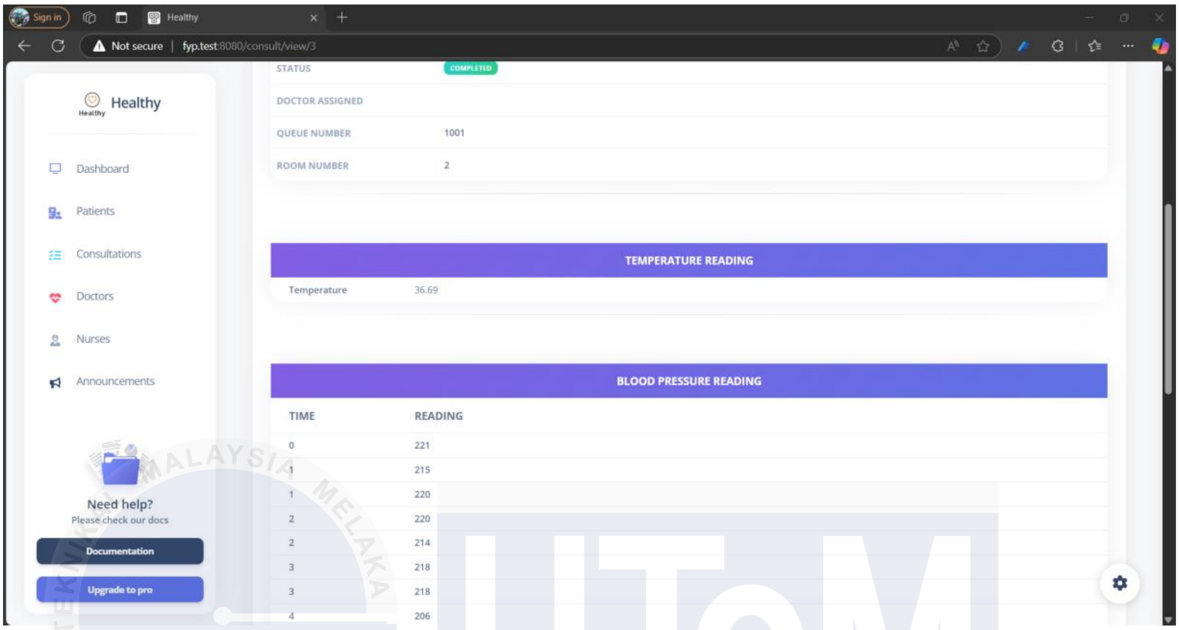
PATIENT DETAILS

NAME	Shazwan Hazmi
IC NUMBER	000418081259
PATIENT ID	1
CONSULTATION ID	2
STATUS	COMPLETED
DOCTOR ASSIGNED	
QUEUE NUMBER	1000
ROOM NUMBER	1

Figure 4.10 Details of Consultation Patient

A well-structured "Consultation Details" section that contains vital information like the patient's name (Shazwan Hazmi), IC number, patient ID, consultation ID, status “Complete”, assigned doctor, queue number, and room number is part of the "Healthy" web-based healthcare management system, which is intended to optimize patient consultation workflows see in Figure 4.18. With its professional purple and white colour scheme and well-organized navigation menus for dashboard access, patient records, consultations, and profile management, as well as a search bar and a "Sign Out" button for safe use, the system boasts a simple and easy-to-use interface. "Healthy" improves operational efficiency, lessens administrative burden, and streamlines queue and room management to improve the overall patient experience by automating the storage and display of consultation details. The system is excellent at its core functions and simplicity, but it is devoid of sophisticated features that are frequently found in other systems, such as "MediFlow." These features include data visualization, dynamic queue tracking, real-time updates, and integration with telemedicine platforms. "Healthy" might become more competitive and appropriate for larger healthcare settings with the addition of these features, as well as improved responsiveness and data visualization capabilities.

4.2.1.8 Data Temperature Patient



STATUS COMPLETED

DOCTOR ASSIGNED

QUEUE NUMBER 1001

ROOM NUMBER 2

TEMPERATURE READING

Temperature 36.69

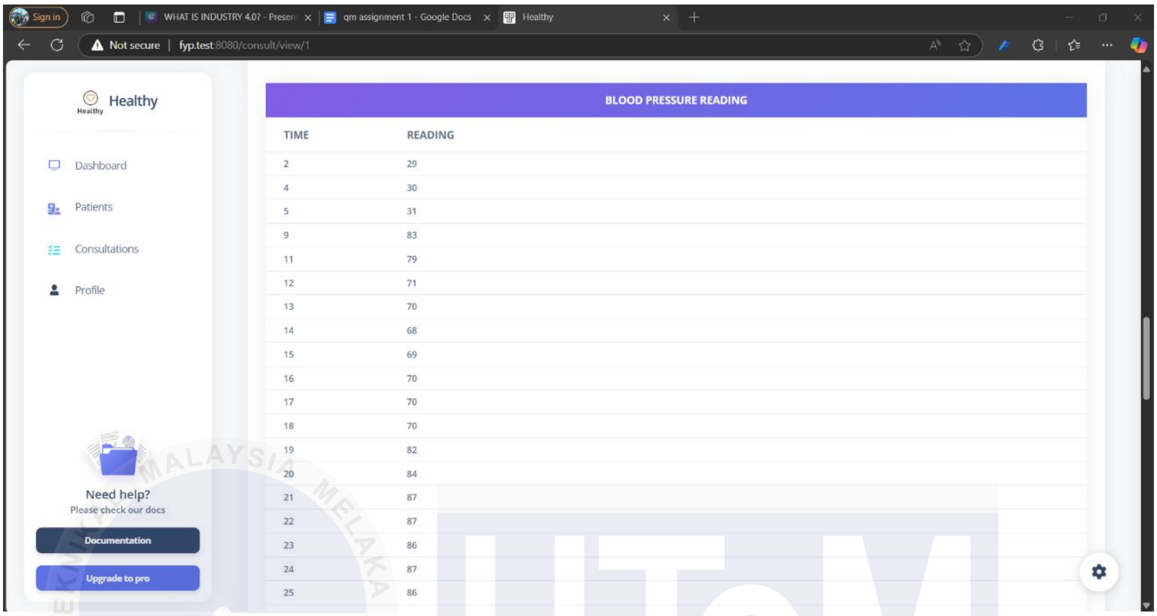
BLOOD PRESSURE READING

TIME	READING
0	221
1	215
1	220
2	220
2	214
3	218
3	218
4	206

Figure 4.11 Data Temperature Patient

The temperature reading module housed in a Laragon MySQL database that is controlled by the Laravel framework is depicted in the figure 4.11. The ESP32 microcontroller and hardware sensors are successfully integrated in the system to collect temperature data in real time. This data is then organized and stored in a database for quick access. Even though it works, reading anomalies like extreme values point to possible problems with calibration or testing conditions. While Laravel streamlines data handling and connectivity, Laragon's lightweight server environment guarantees seamless database operations. This setup prioritizes simplicity and real-time data management over other database solutions, but it could use some extra features like data validation and visualization.

4.2.1.9 Data Blood Pressure Patient



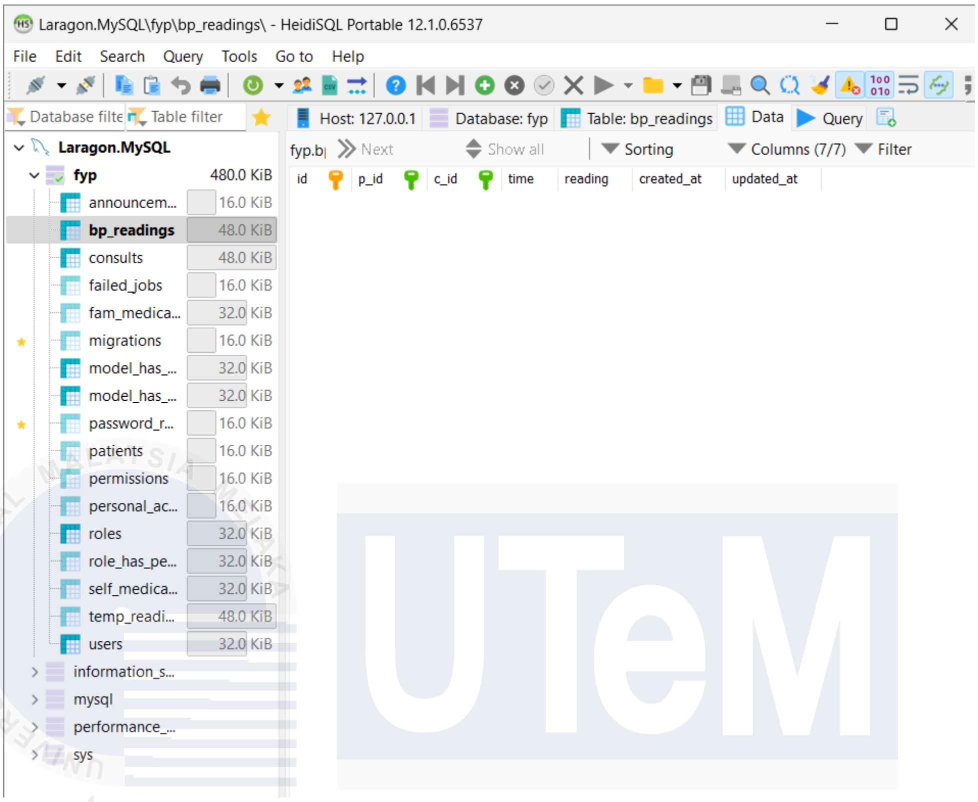
BLOOD PRESSURE READING

TIME	READING
2	29
4	30
5	31
9	83
11	79
12	71
13	70
14	68
15	69
16	70
17	70
18	70
19	82
20	84
21	87
22	87
23	86
24	87
25	86

Figure 4.12 Data Blood Pressure Patient

Figure 4.11 shows the blood pressure reading module, which is managed by the Laravel framework and stored in a Laragon MySQL database. In order to efficiently monitor patient health metrics, this system efficiently combines hardware sensors and the ESP32 microcontroller to gather blood pressure data in real time. It then arranges and stores this data in a database for easy access. Even though the system functions well, reading anomalies like extreme values could be a sign of possible problems with the testing or calibration conditions. The lightweight server environment of Laragon guarantees smooth database operations, while Laravel improves data handling and connectivity. Unlike other database solutions that might provide more complicated configurations, this setup places a higher priority on simplicity and real-time data management. The current implementation, however, might use some extra features, like data validation and visualization tools, to increase the accuracy of the readings and give medical professionals a better understanding of patient health trends, which would ultimately result in better patient care decisions.

4.2.2.1 Database Laragon



Laragon.MySQL	
fyp	480.0 KiB
announcem...	16.0 KiB
bp_readings	48.0 KiB
consults	48.0 KiB
failed_jobs	16.0 KiB
fam_medica...	32.0 KiB
migrations	16.0 KiB
model_has_...	32.0 KiB
model_has_...	32.0 KiB
password_r...	16.0 KiB
patients	16.0 KiB
permissions	16.0 KiB
personal_ac...	16.0 KiB
roles	32.0 KiB
role_has_pe...	32.0 KiB
self_medica...	32.0 KiB
temp_readi...	48.0 KiB
users	32.0 KiB
information_s...	
mysql	
performance_...	
sys	

Figure 4.13 Database Laragon

A real-time data storage solution that can manage structured health-related data, including blood pressure readings, ambient temperature, and adjusted body temperature, is integrated into the database system shown in Figure 4.13. The ESP32-WROOM microcontroller is used by the system to gather data from hardware elements, such as the MLX90614 infrared temperature sensor and a pulse sensor. Laragon, a lightweight local server environment, processes and sends this data to the backend. In order to ensure accurate data storage and retrieval for real-time or historical analysis, the database structure comprises key tables like bp_readings, temp_readings, and patients. These tables contain fields like patient ID (p_id), consultation ID (c_id), readings, timestamps (created_at, updated_at), and time.

With its model-based methodology and sophisticated query builder, the Laravel PHP framework-powered backend increases efficiency by automating CRUD (Create, Read, Update, Delete) operations [23]. This removes a large portion of the manual SQL code writing that is usually needed for tasks like creating queries, processing query results, and setting up database connections. Laravel speeds up development and lowers the chance of errors by eliminating repetitive code structures and minimizing manual handling at every stage. The Arduino IDE enables ESP32-WROOM programming for data processing and transmission, and the system efficiently combines hardware and software components. This demonstrates a smooth transition between secure database storage and real-time data collection, resulting in a stable, scalable, and reliable ecosystem for medical applications.

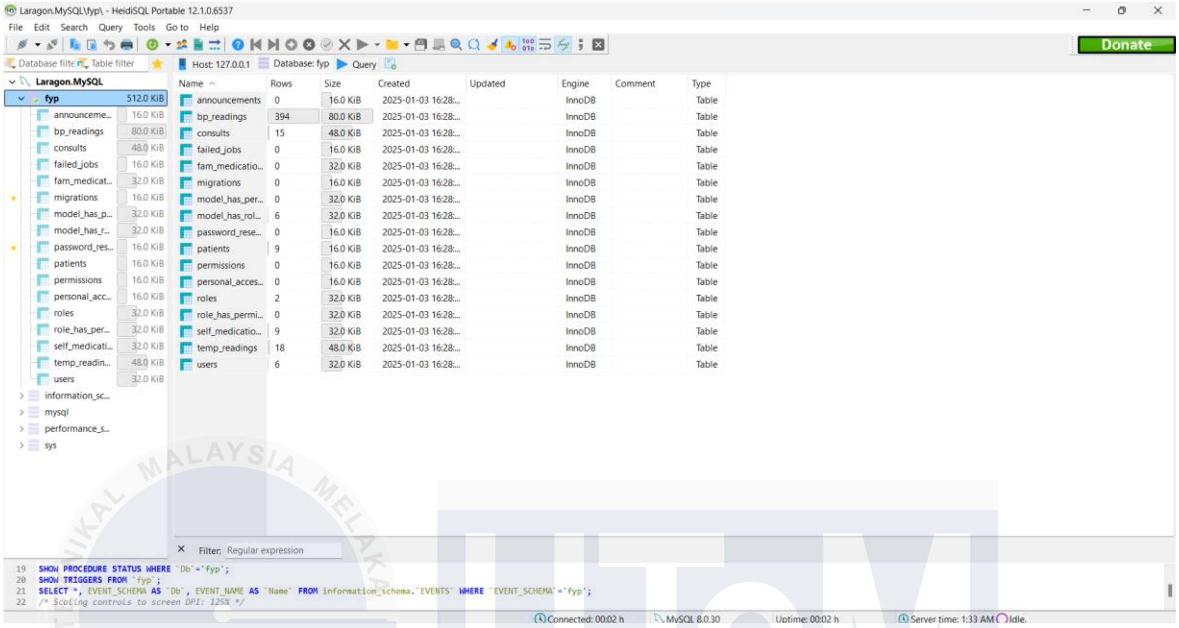
Table 4.4 Survey comparison of Analysis REST API

No.	Analysis	PHP Native	Laravel Framework
1	Rest API Speed	18.3 ms	344.52 ms
2	Code efficiency in creating REST APIs	7 processes	1 process
3	URL structure in REST API	3 paths	2 paths

Table 4.5 Comparison of Similar Database Systems

Feature	Laragon and Laravel Framework	Firebase Real-Time Database	AWS RDS
Database Type	Structured SQL	NoSQL	SQL
Data Retrieval Speed	Fast	Very Fast	Moderate
Integration with ESP32	Direct via ARDUINO IDE	Additional SDKs	Custom API integration
Ease of Deployment	Moderate	Easy	Complex
Scalability	Local Server	Highly scalable	Highly scalable
Timestamps Support	Yes	Built-in	Built-in
Backend Framework	Laravel PHP	SDKs provided	Requires Custom middleware

4.2.2.2 Data Size



Laragon.MySQL	
fyp	512.0 KiB
announceme...	16.0 KiB
bp_readings	80.0 KiB
consults	48.0 KiB
failed_jobs	16.0 KiB
fam_medicat...	32.0 KiB
migrations	16.0 KiB
model_has_p...	32.0 KiB
model_has_r...	32.0 KiB
password_res...	16.0 KiB
patients	16.0 KiB
permissions	16.0 KiB
personal_acc...	16.0 KiB
roles	32.0 KiB
role_has_per...	32.0 KiB
self_medicati...	32.0 KiB
temp_readin...	48.0 KiB
users	32.0 KiB
information_sc...	
mysql	
performance_s...	
sys	

Name	Rows	Size	Created	Updated	Engine	Comment	Type
announcements	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
bp_readings	394	80.0 KiB	2025-01-03 16:28:...		InnoDB		Table
consults	15	48.0 KiB	2025-01-03 16:28:...		InnoDB		Table
failed_jobs	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
fam_medicatio...	0	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
migrations	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
model_has_per...	0	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
model_has_rol...	6	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
password_rese...	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
patients	9	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
permissions	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
personal_acces...	0	16.0 KiB	2025-01-03 16:28:...		InnoDB		Table
roles	2	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
role_has_permi...	0	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
self_medicatio...	9	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table
temp_readings	18	48.0 KiB	2025-01-03 16:28:...		InnoDB		Table
users	6	32.0 KiB	2025-01-03 16:28:...		InnoDB		Table

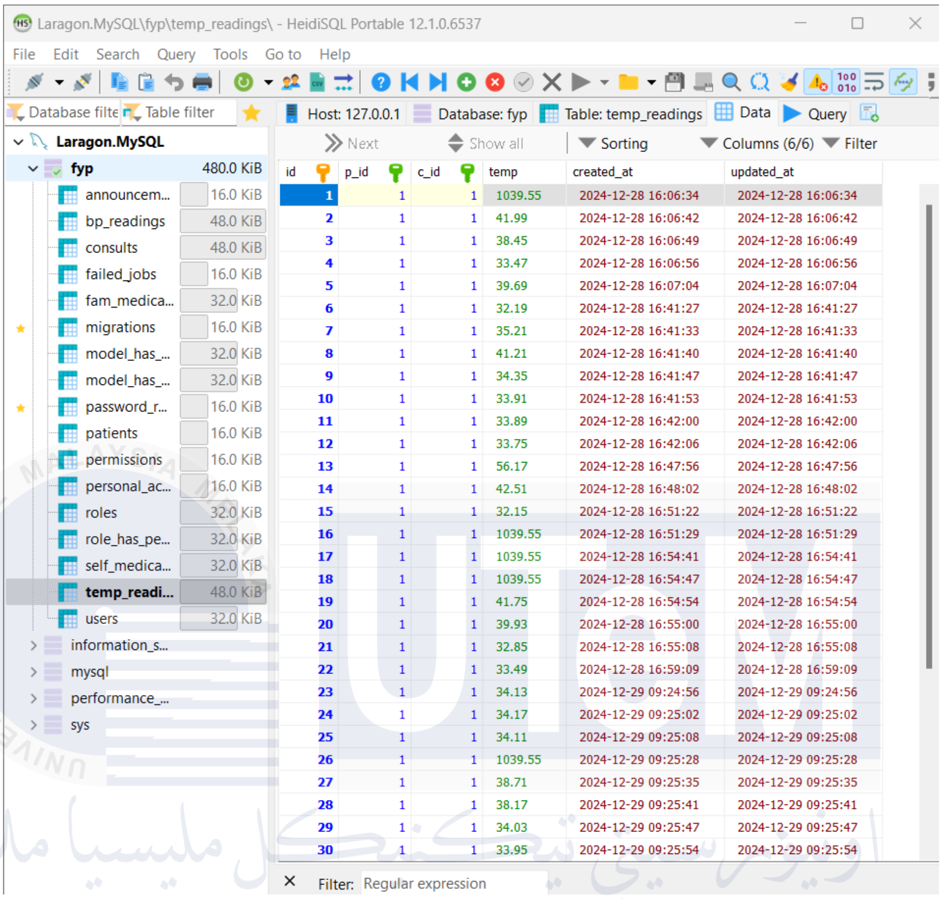
```
SHOW PROCEDURE STATUS WHERE `Db`='fyp';
SHOW TRIGGERS FROM `fyp`;
SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.`EVENTS` WHERE `EVENT_SCHEMA`='fyp';
/* Scaling controls to screen DPI: 125% */
```

Connected: 00:02 h   MySQL 8.0.30   Uptime: 00:02 h   Server time: 1:33 AM   Idle.

Figure 4.14 Data Size

According to the Figure 4.14 that was supplied, the database called "fyp" has multiple tables with notably different data sizes. With 394 KB of data, the largest table, bp_readings, is the one that contains the most records or detailed entries, most likely corresponding to extensive or frequent blood pressure readings. Smaller tables, like announcements (0 KB) and failed_jobs (0 KB), on the other hand, either contain few or no entries or no data at all. Other tables, such as patients (22 KB) and farm_medications (15 KB), indicate moderate data volumes, perhaps because they are used to store information that is less frequent or at the summary level. Reliability and performance are prioritized, especially for transactions, as evidenced by the effective arrangement and usage of InnoDB as the storage engine across all tables. Critical health metrics take up more space in this distribution, which emphasizes the data priorities of the health monitoring system and is consistent with the system's emphasis on tracking and analysing health trends.

4.2.2.3 Database Temperature Patient



Laragon.MySQL\fyp\temp_readings\ - HeidiSQL Portable 12.1.0.6537
File Edit Search Query Tools Go to Help
Database filter
Table filter
Host: 127.0.0.1
Database: fyp
Table: temp_readings
Data
Query
Next
Show all
Sorting
Columns (6/6)
Filter
Filter: Regular expression

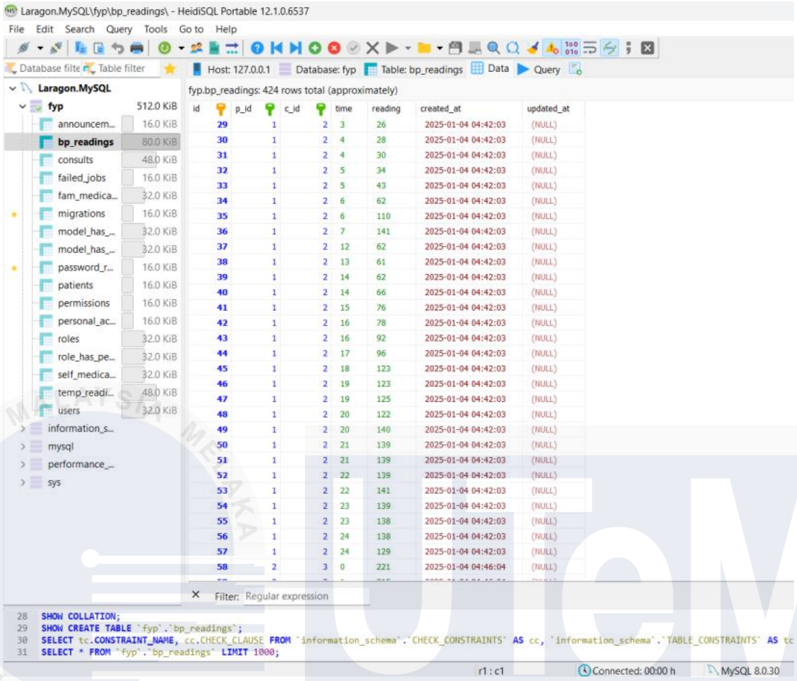
Laragon.MySQL	
fyp	480.0 KiB
announcem...	16.0 KiB
bp_readings	48.0 KiB
consults	48.0 KiB
failed_jobs	16.0 KiB
fam_medica...	32.0 KiB
migrations	16.0 KiB
model_has_...	32.0 KiB
model_has_...	32.0 KiB
password_r...	16.0 KiB
patients	16.0 KiB
permissions	16.0 KiB
personal_ac...	16.0 KiB
roles	32.0 KiB
role_has_pe...	32.0 KiB
self_medica...	32.0 KiB
temp_readi...	48.0 KiB
users	32.0 KiB
information_s...	
mysql	
performance_...	
sys	

id	p_id	c_id	temp	created_at	updated_at
1	1	1	1039.55	2024-12-28 16:06:34	2024-12-28 16:06:34
2	1	1	41.99	2024-12-28 16:06:42	2024-12-28 16:06:42
3	1	1	38.45	2024-12-28 16:06:49	2024-12-28 16:06:49
4	1	1	33.47	2024-12-28 16:06:56	2024-12-28 16:06:56
5	1	1	39.69	2024-12-28 16:07:04	2024-12-28 16:07:04
6	1	1	32.19	2024-12-28 16:41:27	2024-12-28 16:41:27
7	1	1	35.21	2024-12-28 16:41:33	2024-12-28 16:41:33
8	1	1	41.21	2024-12-28 16:41:40	2024-12-28 16:41:40
9	1	1	34.35	2024-12-28 16:41:47	2024-12-28 16:41:47
10	1	1	33.91	2024-12-28 16:41:53	2024-12-28 16:41:53
11	1	1	33.89	2024-12-28 16:42:00	2024-12-28 16:42:00
12	1	1	33.75	2024-12-28 16:42:06	2024-12-28 16:42:06
13	1	1	56.17	2024-12-28 16:47:56	2024-12-28 16:47:56
14	1	1	42.51	2024-12-28 16:48:02	2024-12-28 16:48:02
15	1	1	32.15	2024-12-28 16:51:22	2024-12-28 16:51:22
16	1	1	1039.55	2024-12-28 16:51:29	2024-12-28 16:51:29
17	1	1	1039.55	2024-12-28 16:54:41	2024-12-28 16:54:41
18	1	1	1039.55	2024-12-28 16:54:47	2024-12-28 16:54:47
19	1	1	41.75	2024-12-28 16:54:54	2024-12-28 16:54:54
20	1	1	39.93	2024-12-28 16:55:00	2024-12-28 16:55:00
21	1	1	32.85	2024-12-28 16:55:08	2024-12-28 16:55:08
22	1	1	33.49	2024-12-28 16:59:09	2024-12-28 16:59:09
23	1	1	34.13	2024-12-29 09:24:56	2024-12-29 09:24:56
24	1	1	34.17	2024-12-29 09:25:02	2024-12-29 09:25:02
25	1	1	34.11	2024-12-29 09:25:08	2024-12-29 09:25:08
26	1	1	1039.55	2024-12-29 09:25:28	2024-12-29 09:25:28
27	1	1	38.71	2024-12-29 09:25:35	2024-12-29 09:25:35
28	1	1	38.17	2024-12-29 09:25:41	2024-12-29 09:25:41
29	1	1	34.03	2024-12-29 09:25:47	2024-12-29 09:25:47
30	1	1	33.95	2024-12-29 09:25:54	2024-12-29 09:25:54

Figure 4.15 Database Temperature Patient

Effective hardware-software integration is demonstrated by the temperature reading database that show at Figure 4.15, which was constructed with MySQL for management and Laragon as the server environment. Accurate tracking is ensured by storing real-time sensor data in the temp_readings table, which includes fields for temperature, timestamps, patient and consultation IDs. The system effectively manages real-time storage and retrieval, confirming its functionality, even though some reading anomalies point to calibration problems or testing situations. MySQL and Laragon offer a scalable, lightweight solution that makes it possible for healthcare applications to have reliable database operations.

4.2.2.4 Database Blood Pressure Patient



fyp.bp_readings: 424 rows total (approximately)

id	p_id	c_id	time	reading	created_at	updated_at
29	1	2	3	26	2025-01-04 04:42:03	(NULL)
30	1	2	4	28	2025-01-04 04:42:03	(NULL)
31	1	2	4	30	2025-01-04 04:42:03	(NULL)
32	1	2	5	34	2025-01-04 04:42:03	(NULL)
33	1	2	5	43	2025-01-04 04:42:03	(NULL)
34	1	2	6	62	2025-01-04 04:42:03	(NULL)
35	1	2	6	110	2025-01-04 04:42:03	(NULL)
36	1	2	7	141	2025-01-04 04:42:03	(NULL)
37	1	2	12	62	2025-01-04 04:42:03	(NULL)
38	1	2	13	61	2025-01-04 04:42:03	(NULL)
39	1	2	14	62	2025-01-04 04:42:03	(NULL)
40	1	2	14	66	2025-01-04 04:42:03	(NULL)
41	1	2	15	76	2025-01-04 04:42:03	(NULL)
42	1	2	16	78	2025-01-04 04:42:03	(NULL)
43	1	2	16	92	2025-01-04 04:42:03	(NULL)
44	1	2	17	96	2025-01-04 04:42:03	(NULL)
45	1	2	18	123	2025-01-04 04:42:03	(NULL)
46	1	2	19	123	2025-01-04 04:42:03	(NULL)
47	1	2	19	125	2025-01-04 04:42:03	(NULL)
48	1	2	20	122	2025-01-04 04:42:03	(NULL)
49	1	2	20	140	2025-01-04 04:42:03	(NULL)
50	1	2	21	139	2025-01-04 04:42:03	(NULL)
51	1	2	21	139	2025-01-04 04:42:03	(NULL)
52	1	2	22	139	2025-01-04 04:42:03	(NULL)
53	1	2	22	141	2025-01-04 04:42:03	(NULL)
54	1	2	23	139	2025-01-04 04:42:03	(NULL)
55	1	2	23	138	2025-01-04 04:42:03	(NULL)
56	1	2	24	138	2025-01-04 04:42:03	(NULL)
57	1	2	24	129	2025-01-04 04:42:03	(NULL)
58	2	3	0	221	2025-01-04 04:46:04	(NULL)

```
SHOW COLLATION;
SHOW CREATE TABLE `fyp`.`bp_readings`;
SELECT tc.CONSTRAINT_NAME, cc.CHECK_CLAUSE FROM `information_schema`.`CHECK_CONSTRAINTS` AS cc, `information_schema`.`TABLE_CONSTRAINTS` AS tc
SELECT * FROM `fyp`.`bp_readings` LIMIT 1000;
```

Figure 4.16 Database Blood Pressure Patient

The figures 4.16, show a simplified pipeline that uses a hardware device coupled with a web application and a MySQL database run by Laragon to gather, transmit, store, and display blood pressure readings. With timestamps, the bp_readings table effectively saves patient and consultation data, guaranteeing safe storage and easy retrieval. This integration demonstrates Laragon MySQL's dependability in handling real-time healthcare data by showcasing the effective use of communication protocols and intuitive user interfaces to bridge hardware and software. A hardware device provides the readings, which are then sent to a web application and kept in a MySQL database under Laragon's management.

4.2.3 Hardware Configuration Results and Analysis

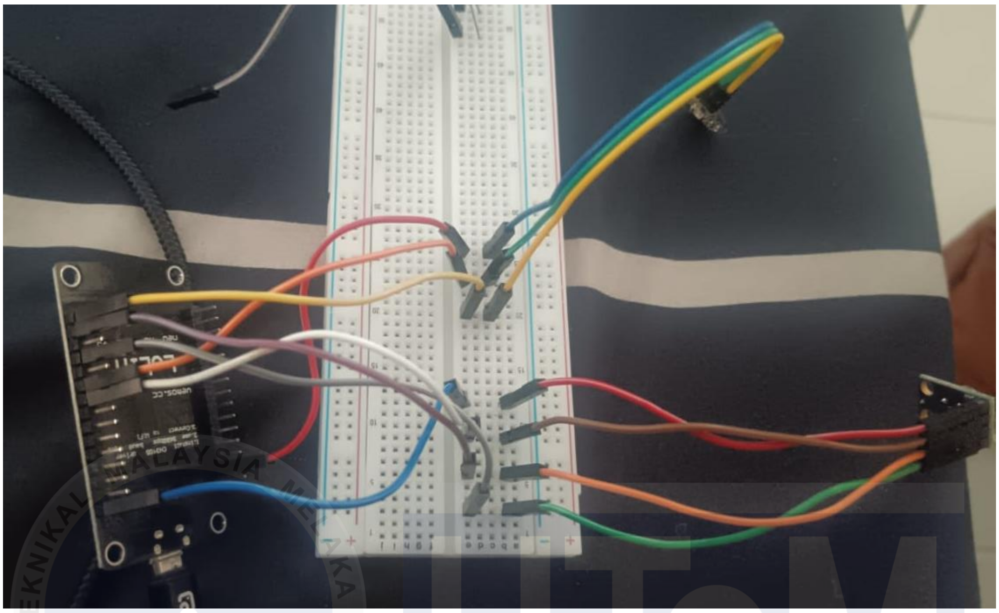


Figure 4.17 Connection of Hardware

An infrared non-contact digital temperature sensor and a heart rate monitor are integrated into the ESP32-WROOM-based real-time physiological monitoring system to gather and process critical data. The heart rate monitor uses the ESP32's ADC for digital processing after detecting electrical signals from heartbeats and converting them into analog voltage levels. It is connected to the ESP32 via an analog pin (such as GPIO34). In order to provide accurate digital temperature readings without physical contact, the infrared temperature sensor simultaneously uses GPIO21 (SDA) and GPIO22 (SCL) to communicate with the ESP32 via the I²C protocol. The ESP32, which is programmed using the Arduino IDE, combines inputs from both sensors for real-time analysis and guarantees synchronized data acquisition. Continuous monitoring is made possible by the wireless transmission of the processed data via Bluetooth or Wi-Fi to distant servers or display devices. Because it can non-invasively measure heart rate and body temperature, this configuration is ideal for health monitoring applications. It is also scalable for Internet of Things applications like fitness tracking and remote patient monitoring.

The connections based shown in the Figure 4.10, with the following points in Table 4.6:

Table 4.6 Connection of Hardware Microcontroller

Metric	**Connection**
Heart Rate Tracker:	- An analog pin (such as GPIO34 or GPIO35) connects the heart rate monitor to the ESP32, which interprets and transforms electrical signals produced by heartbeats into analog voltage levels. - The ESP32's Analog-to-Digital Converter (ADC) processes these analog signals, digitizing the data for sophisticated analysis, such as signal filtering and beats per minute (BPM) computation, guaranteeing precise and trustworthy heart rate monitoring.
Non-Contact Digital Infrared Temperature Sensor:	- The ESP32's GPIO19 and GPIO22 are connected to the temperature sensor's SDA (Data Line) and SCL (Clock Line), which use the I²C protocol for communication. This configuration guarantees dependable and effective data transfer between the microcontroller and the sensor. - The sensor is made to operate without making contact and provides a highly accurate remote measurement of body or surface temperature. The digital data from the sensor is processed by the ESP32, allowing for precise and smooth integration into the physiological monitoring system in real time.

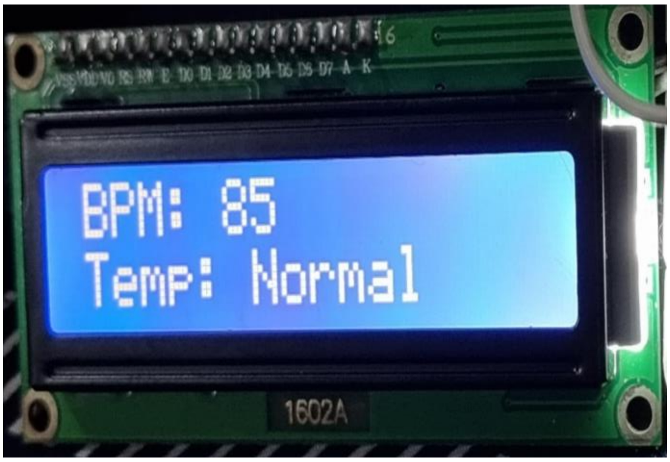
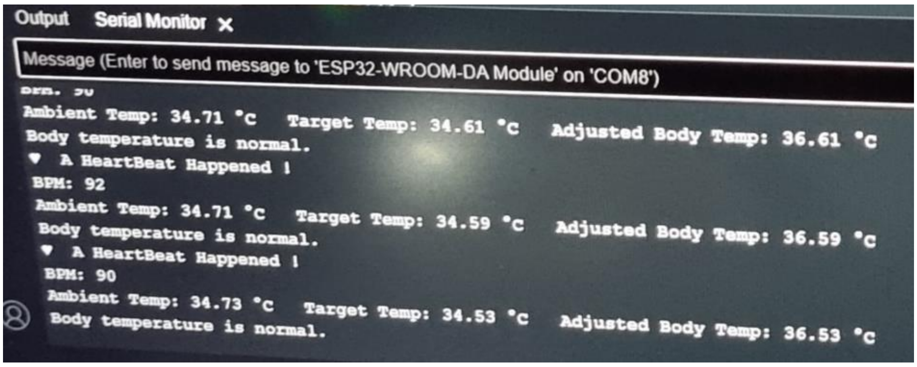


Figure 4.18 LCD Result

Real-time visualization of critical health metrics is made possible by the integration of a 16x2 LCD module, which improves the system's usability and functionality. The first row of the display is designed to show the pulse rate, expressed in beats per minute (BPM), and the second row analyses the temperature status, classifying it as "Normal," "Low," or "High" according to preset thresholds which show at Figure 4.18. This classification makes rapid health evaluations easier. The ESP32 microcontroller interprets data by processing input from the pulse sensor and the MLX90614 infrared temperature sensor. The LCD is then dynamically updated with the processed data. By removing the need for extra hardware, this direct interface guarantees the system's usability and accessibility. Users can effectively monitor vital metrics thanks to the system's clear and concise representation of health parameters. The structured display format facilitates efficient real-time data communication, which enhances health status decision-making.



Output
Serial Monitor ×
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM8')

```
BPM: 90
Ambient Temp: 34.71 °C   Target Temp: 34.61 °C   Adjusted Body Temp: 36.61 °C
Body temperature is normal.
 ♥  A HeartBeat Happened !
 BPM: 92
Ambient Temp: 34.71 °C   Target Temp: 34.59 °C   Adjusted Body Temp: 36.59 °C
Body temperature is normal.
 ♥  A HeartBeat Happened !
 BPM: 90
Ambient Temp: 34.73 °C   Target Temp: 34.53 °C   Adjusted Body Temp: 36.53 °C
Body temperature is normal.
```

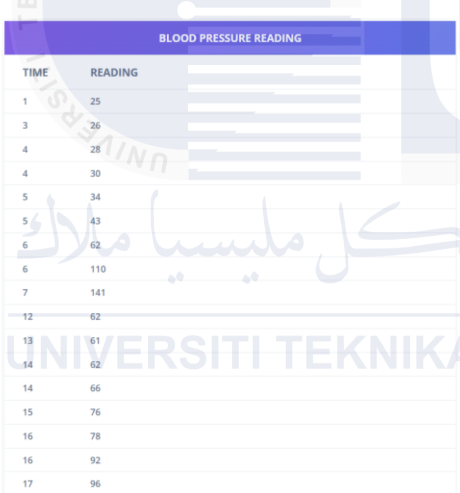
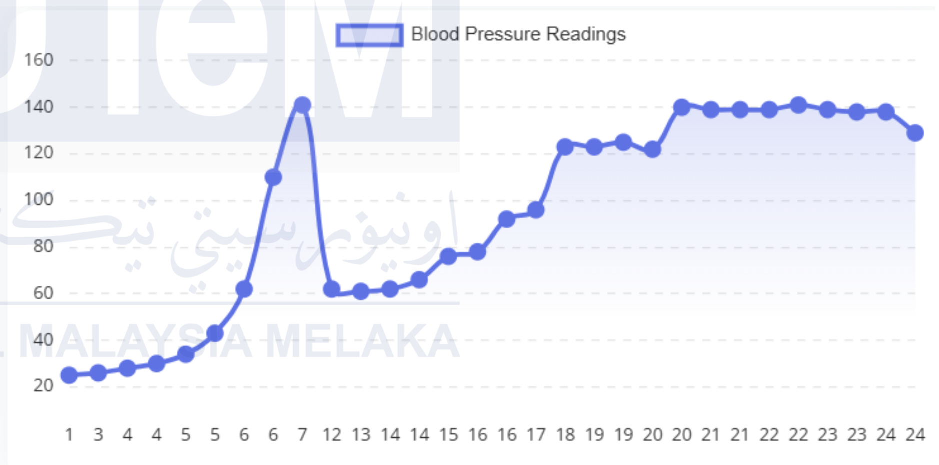
Figure 4.19 Result in Arduino IDE

The Arduino IDE's Serial Monitor output verifies the system's operation through the ESP32-WROOM module's real-time monitoring of body temperature and heart rate data, as seen in Figure 4.19. The output shows the target temperature, ambient temperature, adjusted body temperature, and calculated beats per minute (BPM). Accurate heartbeat detection is shown by the message "♥ A HeartBeat Happened!" and the current heart rate, and the adjusted body temperature is evaluated to determine whether it is within the normal range, as shown by the message "Body temperature is normal." These outcomes show how precisely the sensors and processing algorithms work, guaranteeing accurate, real-time health metrics and proving the system's usefulness in physiological monitoring applications.

4.2.4 Preliminary Results

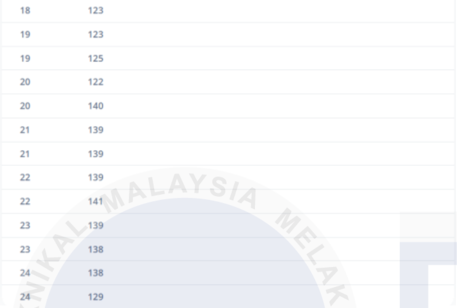
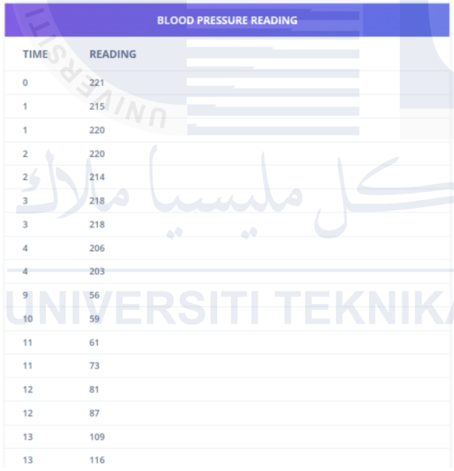
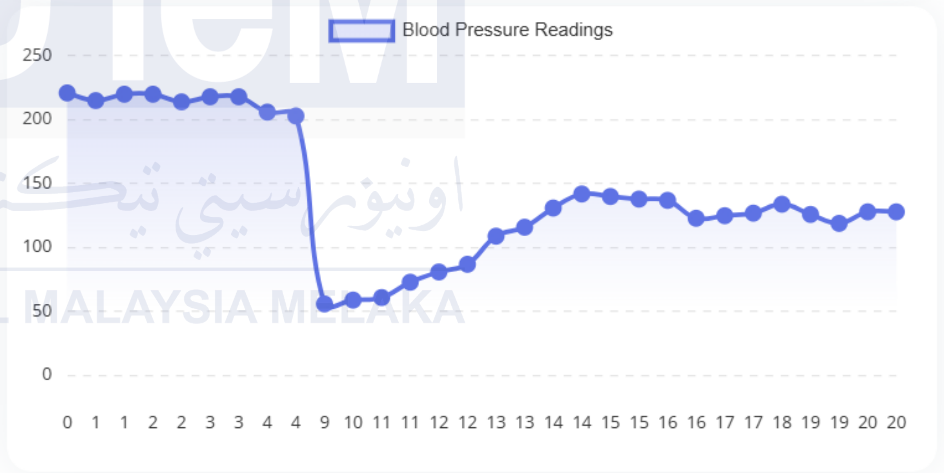
4.2.4.1 Preliminary Results XD-58C Heart Pulse Sensor

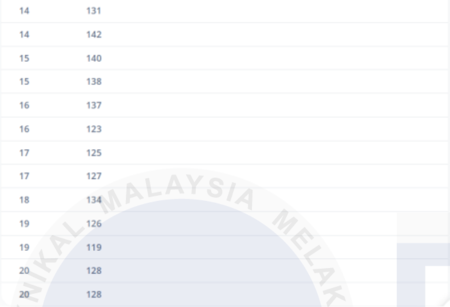
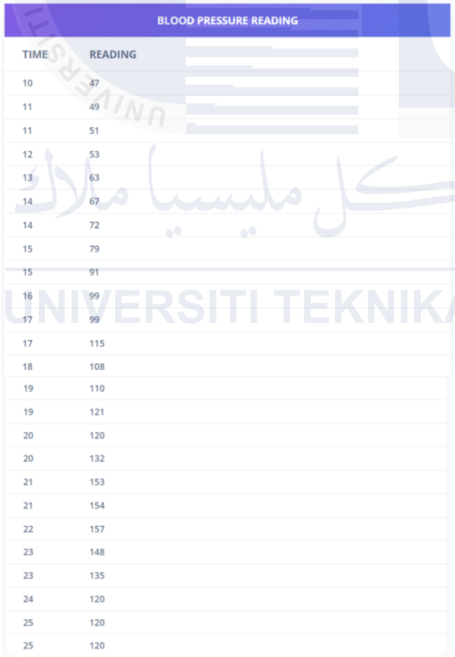
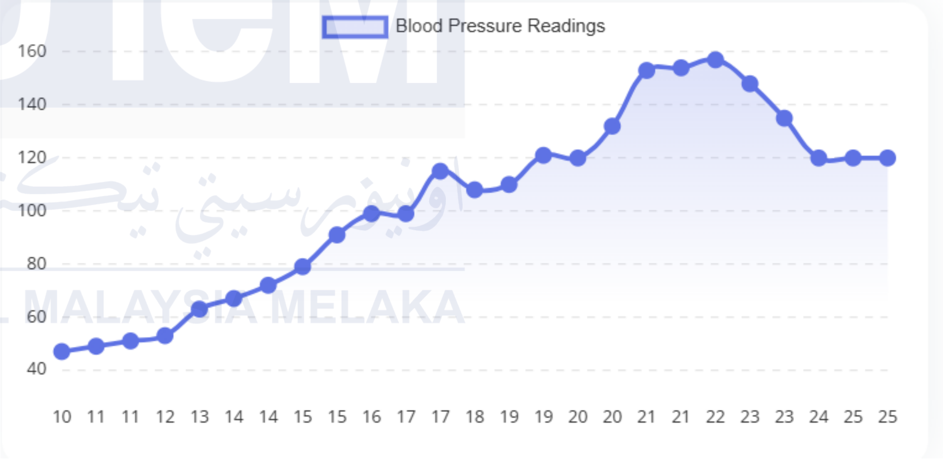
Table 4.7 Preliminary Results XD-58C

Id.	Patient	Result BP	Picture of Chart
1	Shazwan Hazmi		

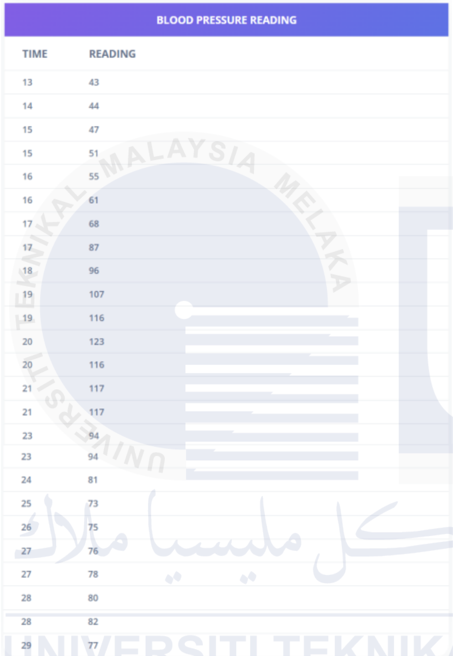
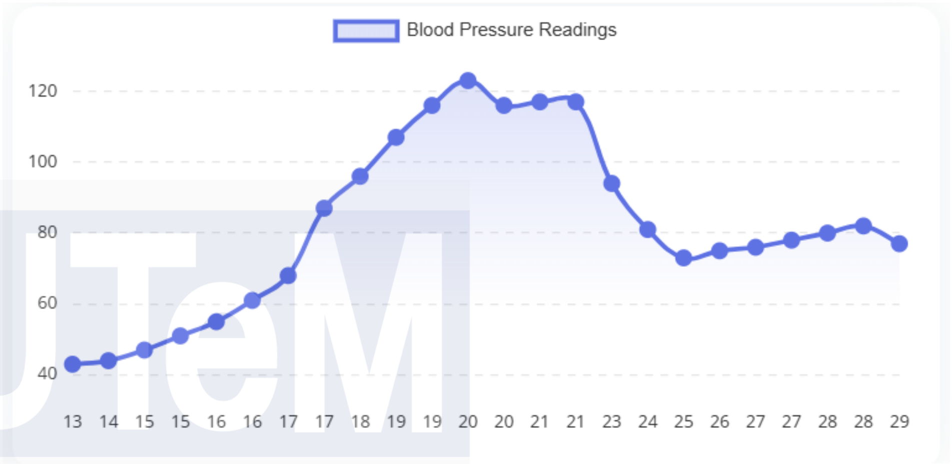
BLOOD PRESSURE READING

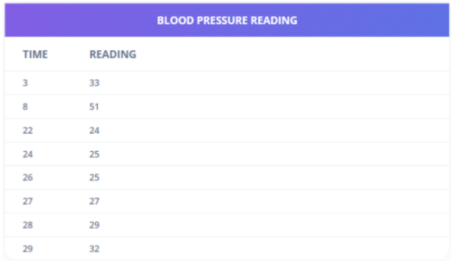
TIME	READING
1	25
3	26
4	28
4	30
5	34
5	43
6	62
6	110
7	141
12	62
13	61
14	62
14	66
15	76
16	78
16	92
17	96

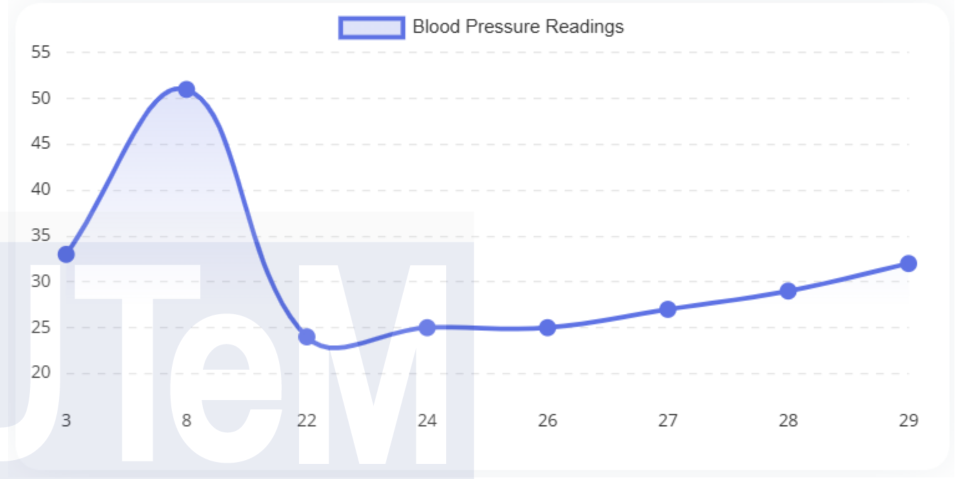
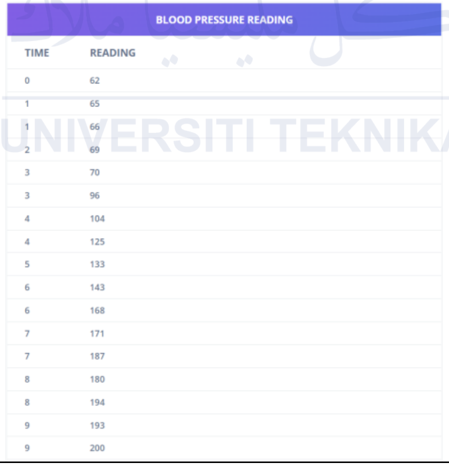
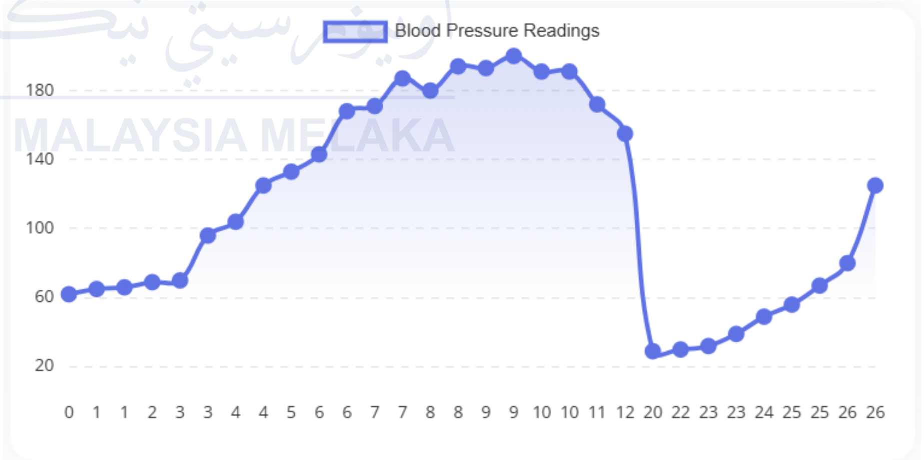
		 18 123 19 123 19 125 20 122 20 140 21 139 21 139 22 139 22 141 23 139 23 138 24 138 24 129	
2	**Darwisy**	 BLOOD PRESSURE READING TIME READING 0 221 1 215 1 220 2 220 2 214 3 218 3 218 4 206 4 203 9 56 10 59 11 61 11 73 12 81 12 87 13 109 13 116	

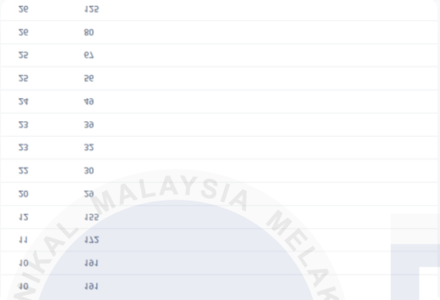
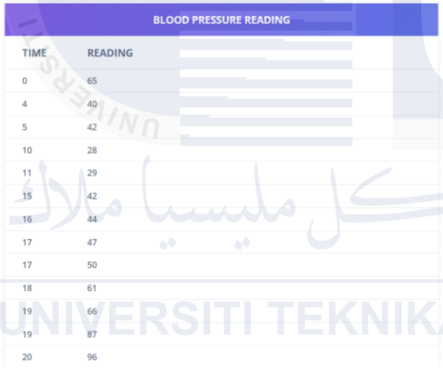
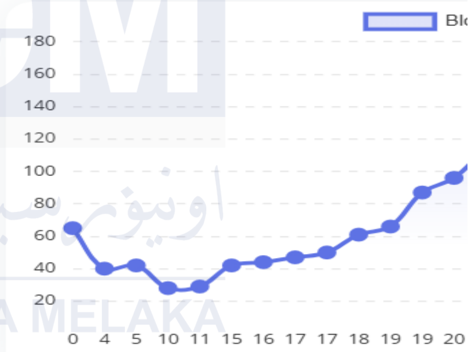
<table>
<tr>
<td></td>
<td></td>
<td>
<table>
<tr><td>14</td><td>131</td></tr>
<tr><td>14</td><td>142</td></tr>
<tr><td>15</td><td>140</td></tr>
<tr><td>15</td><td>138</td></tr>
<tr><td>16</td><td>137</td></tr>
<tr><td>16</td><td>123</td></tr>
<tr><td>17</td><td>125</td></tr>
<tr><td>17</td><td>127</td></tr>
<tr><td>18</td><td>134</td></tr>
<tr><td>19</td><td>126</td></tr>
<tr><td>19</td><td>119</td></tr>
<tr><td>20</td><td>128</td></tr>
<tr><td>20</td><td>128</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td><b>3</b></td>
<td><b>Asyraf</b></td>
<td>
BLOOD PRESSURE READING
<table>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>10</td><td>47</td></tr>
<tr><td>11</td><td>49</td></tr>
<tr><td>11</td><td>51</td></tr>
<tr><td>12</td><td>53</td></tr>
<tr><td>13</td><td>63</td></tr>
<tr><td>14</td><td>67</td></tr>
<tr><td>14</td><td>72</td></tr>
<tr><td>15</td><td>79</td></tr>
<tr><td>15</td><td>91</td></tr>
<tr><td>16</td><td>99</td></tr>
<tr><td>17</td><td>99</td></tr>
<tr><td>17</td><td>115</td></tr>
<tr><td>18</td><td>108</td></tr>
<tr><td>19</td><td>110</td></tr>
<tr><td>19</td><td>121</td></tr>
<tr><td>20</td><td>120</td></tr>
<tr><td>20</td><td>132</td></tr>
<tr><td>21</td><td>153</td></tr>
<tr><td>21</td><td>154</td></tr>
<tr><td>22</td><td>157</td></tr>
<tr><td>23</td><td>148</td></tr>
<tr><td>23</td><td>135</td></tr>
<tr><td>24</td><td>120</td></tr>
<tr><td>25</td><td>120</td></tr>
<tr><td>25</td><td>120</td></tr>
</table>
</td>
<td>

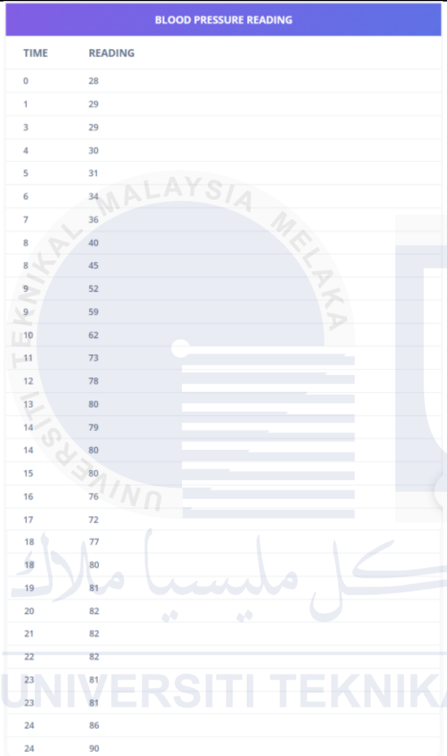
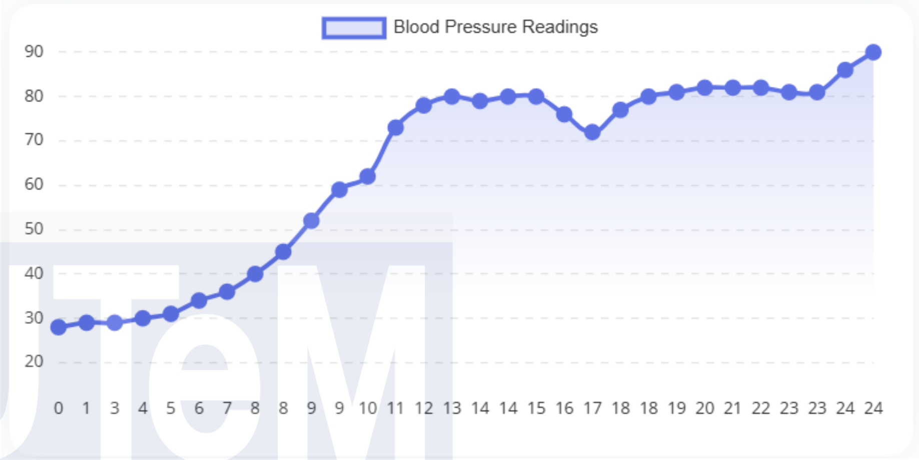
</td>
</tr>
</table>

<table>
<tr>
<td><b>4</b></td>
<td><b>Mizan</b></td>
<td>

<table>
<tr><th colspan="2">BLOOD PRESSURE READING</th></tr>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>13</td><td>43</td></tr>
<tr><td>14</td><td>44</td></tr>
<tr><td>15</td><td>47</td></tr>
<tr><td>15</td><td>51</td></tr>
<tr><td>16</td><td>55</td></tr>
<tr><td>16</td><td>61</td></tr>
<tr><td>17</td><td>68</td></tr>
<tr><td>17</td><td>87</td></tr>
<tr><td>18</td><td>96</td></tr>
<tr><td>19</td><td>107</td></tr>
<tr><td>19</td><td>116</td></tr>
<tr><td>20</td><td>123</td></tr>
<tr><td>20</td><td>116</td></tr>
<tr><td>21</td><td>117</td></tr>
<tr><td>21</td><td>117</td></tr>
<tr><td>23</td><td>94</td></tr>
<tr><td>23</td><td>94</td></tr>
<tr><td>24</td><td>81</td></tr>
<tr><td>25</td><td>73</td></tr>
<tr><td>26</td><td>75</td></tr>
<tr><td>27</td><td>76</td></tr>
<tr><td>27</td><td>78</td></tr>
<tr><td>28</td><td>80</td></tr>
<tr><td>28</td><td>82</td></tr>
<tr><td>29</td><td>77</td></tr>
</table>
</td>
<td>


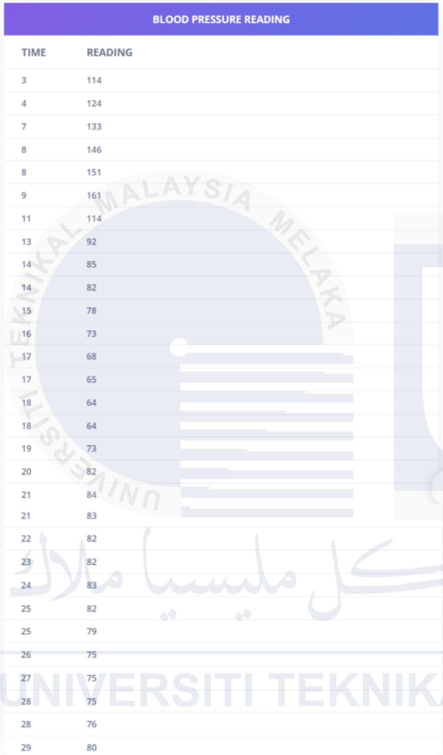
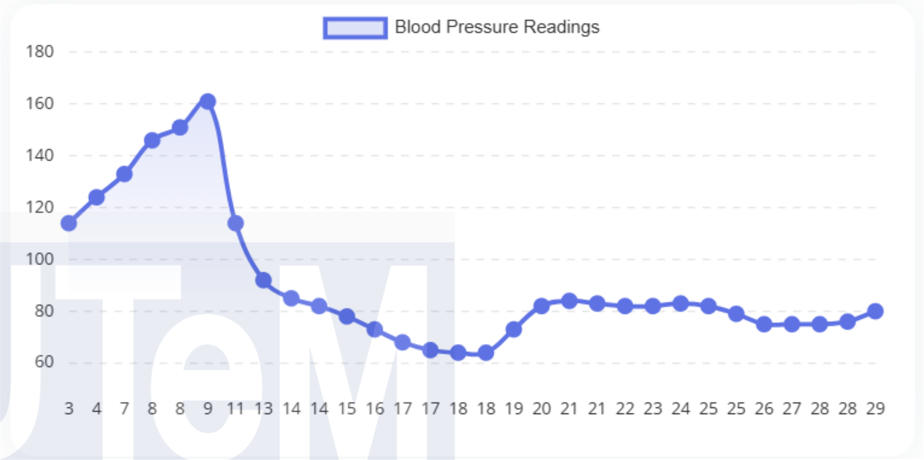
</td>
</tr>
<tr>
<td><b>5</b></td>
<td><b>Fidauddin</b></td>
<td>

<table>
<tr><th colspan="2">BLOOD PRESSURE READING</th></tr>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>3</td><td>33</td></tr>
<tr><td>8</td><td>51</td></tr>
<tr><td>22</td><td>24</td></tr>
<tr><td>24</td><td>25</td></tr>
<tr><td>26</td><td>25</td></tr>
<tr><td>27</td><td>27</td></tr>
<tr><td>28</td><td>29</td></tr>
<tr><td>29</td><td>32</td></tr>
</table>
</td>
<td></td>
</tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>
</td>
</tr>
<tr>
<td><b>6.</b></td>
<td><b>Aishah</b></td>
<td>
BLOOD PRESSURE READING
<table>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>0</td><td>62</td></tr>
<tr><td>1</td><td>65</td></tr>
<tr><td>1</td><td>66</td></tr>
<tr><td>2</td><td>69</td></tr>
<tr><td>3</td><td>70</td></tr>
<tr><td>3</td><td>96</td></tr>
<tr><td>4</td><td>104</td></tr>
<tr><td>4</td><td>125</td></tr>
<tr><td>5</td><td>133</td></tr>
<tr><td>6</td><td>143</td></tr>
<tr><td>6</td><td>168</td></tr>
<tr><td>7</td><td>171</td></tr>
<tr><td>7</td><td>187</td></tr>
<tr><td>8</td><td>180</td></tr>
<tr><td>8</td><td>194</td></tr>
<tr><td>9</td><td>193</td></tr>
<tr><td>9</td><td>200</td></tr>
</table></td>
<td>
</td>
</tr>
</table>

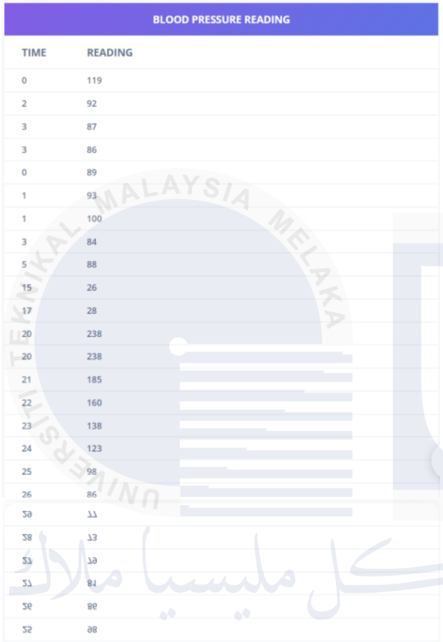
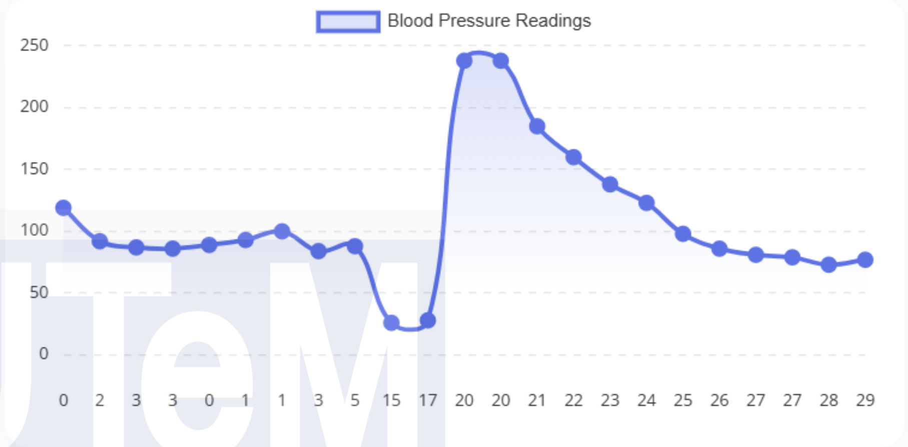
		 26 125 26 80 25 67 25 56 24 46 23 36 23 35 22 30 20 26 15 155 11 115 10 161 10 161	
7.	**Shazwan Hazmi**	 BLOOD PRESSURE READING TIME READING 0 65 4 40 5 42 10 28 11 29 15 42 16 44 17 47 17 50 18 61 19 66 19 87 20 96	

<table>
<tr>
<td><b>8.</b></td>
<td><b>Asyraf</b></td>
<td>
BLOOD PRESSURE READING
<table>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>0</td><td>28</td></tr>
<tr><td>1</td><td>29</td></tr>
<tr><td>3</td><td>29</td></tr>
<tr><td>4</td><td>30</td></tr>
<tr><td>5</td><td>31</td></tr>
<tr><td>6</td><td>34</td></tr>
<tr><td>7</td><td>36</td></tr>
<tr><td>8</td><td>40</td></tr>
<tr><td>8</td><td>45</td></tr>
<tr><td>9</td><td>52</td></tr>
<tr><td>9</td><td>59</td></tr>
<tr><td>10</td><td>62</td></tr>
<tr><td>11</td><td>73</td></tr>
<tr><td>12</td><td>78</td></tr>
<tr><td>13</td><td>80</td></tr>
<tr><td>14</td><td>79</td></tr>
<tr><td>14</td><td>80</td></tr>
<tr><td>15</td><td>80</td></tr>
<tr><td>16</td><td>76</td></tr>
<tr><td>17</td><td>72</td></tr>
<tr><td>18</td><td>77</td></tr>
<tr><td>18</td><td>80</td></tr>
<tr><td>19</td><td>81</td></tr>
<tr><td>20</td><td>82</td></tr>
<tr><td>21</td><td>82</td></tr>
<tr><td>22</td><td>82</td></tr>
<tr><td>23</td><td>81</td></tr>
<tr><td>23</td><td>81</td></tr>
<tr><td>24</td><td>86</td></tr>
<tr><td>24</td><td>90</td></tr>
</table>
</td>
<td>

</td>
</tr>
</table>

<table>
<tr>
<td><b>9.</b></td>
<td><b>Darwisy</b></td>
<td>

BLOOD PRESSURE READING
<table>
<tr><th>TIME</th><th>READING</th></tr>
<tr><td>3</td><td>114</td></tr>
<tr><td>4</td><td>124</td></tr>
<tr><td>7</td><td>133</td></tr>
<tr><td>8</td><td>146</td></tr>
<tr><td>8</td><td>151</td></tr>
<tr><td>9</td><td>161</td></tr>
<tr><td>11</td><td>114</td></tr>
<tr><td>13</td><td>92</td></tr>
<tr><td>14</td><td>85</td></tr>
<tr><td>14</td><td>82</td></tr>
<tr><td>15</td><td>78</td></tr>
<tr><td>16</td><td>73</td></tr>
<tr><td>17</td><td>68</td></tr>
<tr><td>17</td><td>65</td></tr>
<tr><td>18</td><td>64</td></tr>
<tr><td>18</td><td>64</td></tr>
<tr><td>19</td><td>73</td></tr>
<tr><td>20</td><td>82</td></tr>
<tr><td>21</td><td>84</td></tr>
<tr><td>21</td><td>83</td></tr>
<tr><td>22</td><td>82</td></tr>
<tr><td>23</td><td>82</td></tr>
<tr><td>24</td><td>83</td></tr>
<tr><td>25</td><td>82</td></tr>
<tr><td>25</td><td>79</td></tr>
<tr><td>26</td><td>75</td></tr>
<tr><td>27</td><td>75</td></tr>
<tr><td>28</td><td>75</td></tr>
<tr><td>28</td><td>76</td></tr>
<tr><td>29</td><td>80</td></tr>
</table>
</td>
<td>


</td>
</tr>
</table>

10.	**Amir**		

BLOOD PRESSURE READING

TIME	READING
0	119
2	92
3	87
3	86
0	89
1	93
1	100
3	84
5	88
15	26
17	28
20	238
20	238
21	185
22	160
23	138
24	123
25	98
26	86
29	77
28	73
27	79
27	81
26	86
25	98

4.2.4.2 Data Analysis for XD-58C Heart Pulse Sensor

This section will calculate the data for the blood pressure reading value in units of seconds and demonstrate how the heart pulse has been reading. Ten readings are obtained from patients varying in age and the medications they wish to discuss with their physician. Using device hardware that can detect the reading, the nurse will take the patient's blood pressure. It will be necessary to make some adjustments to the heart rate sensor or install a capacitor to lessen any detection interruptions if the value is not accurate.

The table below show how to calculate BPM convert into BPS:

BPS = $\frac{BPM}{60}$

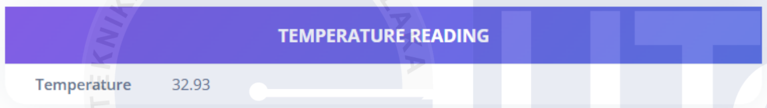
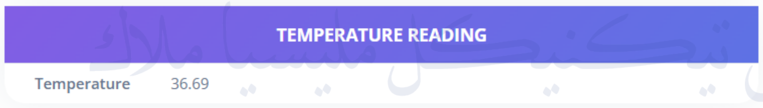
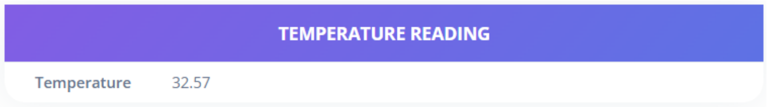
Example:

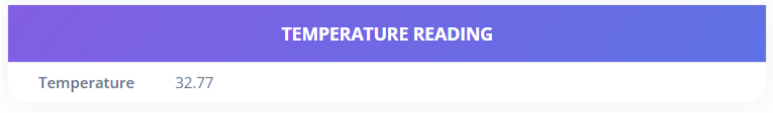
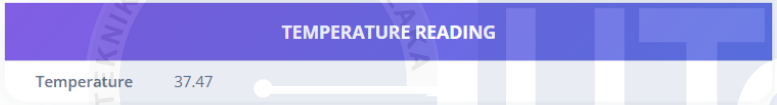
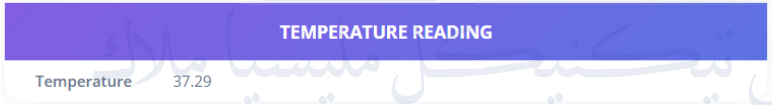
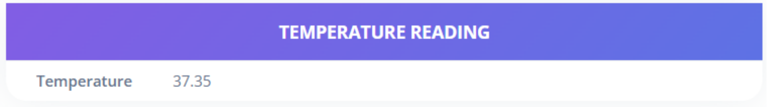
Time	BPM	BPS
10	47	0.733
11	49	0.817
11	51	0.850
12	53	0.883
13	63	1.050
14	72	1.200
14	79	1.317
15	91	1.517
15	90	1.500
16	89	1.483
17	88	1.467
17	88	1.467
18	78	1.300

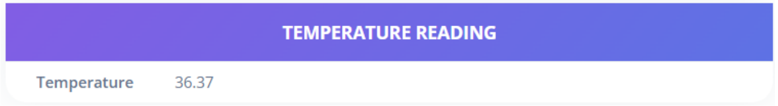
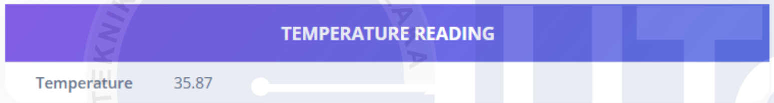
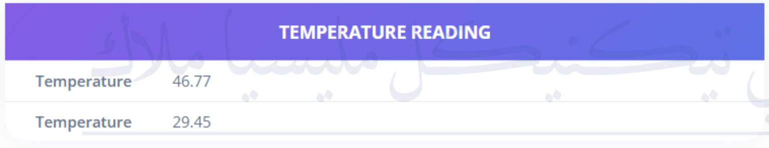
Beats per minute (BPM) data collected over a 10-second period was converted to beats per second (BPS) for a more thorough analysis. The results ranged from 0.733 to 1.517, reflecting small, normal variations over time. While the trend line showed that heart rate stayed within normal physiological ranges, lower BPS readings could indicate bradycardia, which is linked to electrolyte imbalances, hypothyroidism, or problems with heart conduction, while higher BPS readings might indicate conditions like tachycardia, which is linked to stress, fever, or cardiac arrhythmias. According to these findings, the system consistently captures heart rate data in real time with accuracy but not all get the right value of accuracy. Allowing for the precise monitoring required to identify minute variations in cardiovascular activity and offer trustworthy health insights for early intervention.

4.2.4.3 Preliminary Results MLX90614 Non-Contact Infrared Temperature Sensor

Table 4.8 Preliminary Results **MLX90614**

Id.	Patient	Result Temperature	Calculation
1	**Shazwan Hazmi**	 TEMPERATURE READING Temperature 32.93	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{30.0 + 32.93}{2}$ = 31.47℃
2	**Darwisy**	 TEMPERATURE READING Temperature 36.69	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{33.45 + 36.69}{2}$ = 35.07℃
3	**Asyraf**	 TEMPERATURE READING Temperature 32.57	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{30.22 + 32.57}{2}$ = 31.40℃
4	**Mizan**		

		 TEMPERATURE READING Temperature 32.77	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{30.0 + 32.77}{2}$ = 31.40°C
5	**Fidauddin**	 TEMPERATURE READING Temperature 37.47	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{33.34 + 37.47}{2}$ = 35.41°C
6.	**Asyraf**	 TEMPERATURE READING Temperature 37.29	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{33.40 + 37.29}{2}$ = 35.34°C
7.	**Shazwan**	 TEMPERATURE READING Temperature 37.35	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{34.00 + 37.35}{2}$ = 35.68°C
8.	**Amir**		

		 TEMPERATURE READING Temperature 36.37	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{35.00 + 36.37}{2}$ = 35.69°C
9.	**Aishah**	 TEMPERATURE READING Temperature 35.87	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{32.48 + 35.87}{2}$ = 34.18°C
10.	**Darwisy**	 TEMPERATURE READING Temperature 46.77 Temperature 29.45	Body Temperature = $\frac{Object\ Temperature + Ambient\ Temperature}{2}$ Body Temperature = $\frac{31.55 + 46.77}{2}$ = 39.16°C

4.2.4.4 Data Analysis for MLX90614 Non-Contact Infrared Temperature Sensor

Data analysis was done using readings from patients with a variety of medical conditions in order to evaluate the accuracy and reliability of the temperature sensor in determining body temperature in Celsius. These included differences in COVID-19 status, symptom severity, and length of illness. Without the need for calibration adjustments or extra hardware, the sensor continuously recorded precise readings, proving its sturdy construction and dependability in practical situations. In addition to tracking temperature, the sensor offered insightful information about clinical patterns, such as establishing links between the highest temperatures ever recorded and diseases like COVID-19 or influenza. Additionally, it monitored blood pressure (BP) readings, establishing normal ranges for both metrics. Lowered BP is frequently linked to fatigue or dehydration. By allowing medical practitioners to promptly spot alarming patterns, like a rising fever accompanied by unusual blood pressure readings, this enhanced monitoring capability further supports the device's value in clinical settings.

The sensor's reliability for healthcare applications, where precise data is essential for diagnosis and treatment decisions, is demonstrated by its consistent and accurate readings that are free of significant errors. Its real-time monitoring features facilitate easy integration with more extensive health systems and provide useful information to enhance patient care and guide evidence-based procedures. The sensor's ability to function without the need for external adjustments demonstrates its suitability for clinical use, particularly in identifying temperature and blood pressure abnormalities that may signal the start of disease or complications. In addition to directing future developments in sensor technology to improve the accuracy and effectiveness of health monitoring devices, these findings support the sensor's function as a dependable tool in healthcare settings.

4.2.5 Circuit Wiring

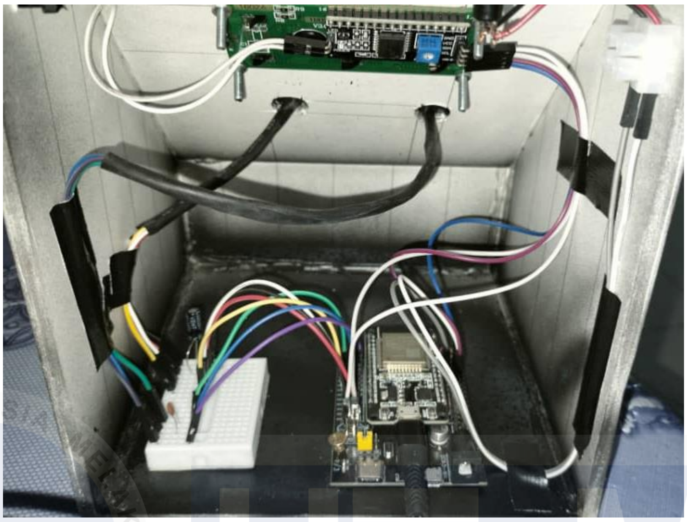


Figure 4.20 Circuit Wiring

As shown in Figure 4.20, the goal of this project is to create a sophisticated health monitoring system using an ESP32 microcontroller, LCD display, push button, non-contact infrared temperature sensor, and heart rate sensor. Capacitors and ceramic components, which are essential for reducing noise and stabilizing sensor readings, are incorporated into the system to guarantee accurate and trustworthy health metrics.

Table 4.9 System Functional Analysis

Heart Rate Monitoring	The heart rate sensor provides vital information about the user's cardiovascular health by precisely detecting pulse signals. This feature is essential for evaluating physiological performance in real time.
Temperature Measurement	Without making physical contact, the non-contact infrared temperature sensor is used to measure body temperature

	hygienically. This method guarantees a more hygienic and convenient experience.

4.2.5.1 Prototype Hardware

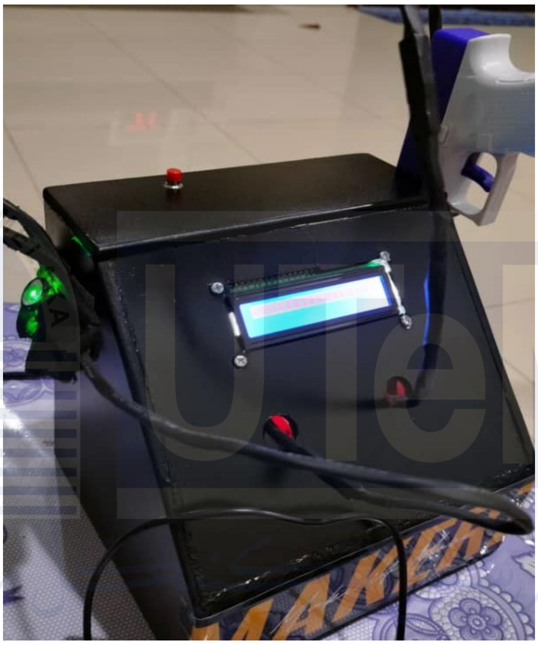


Figure 4.21 Prototype Hardware

An integrated LCD display provides health metrics in an easy-to-read format, improving the user's comprehension of their physiological state, and a push button lets the user interact with the system and initiate measurements as needed. These characteristics, which are shown in Figure 4.21, demonstrate how effectively the gadget can provide useful health information. Additionally, the design incorporates ceramic components and capacitors, which are essential for stabilizing signal output and lowering electrical noise. The overall efficacy and dependability of the system are enhanced by this optimization, which guarantees that the sensor readings stay precise and trustworthy under a variety of circumstances.

4.2.5.2 Model

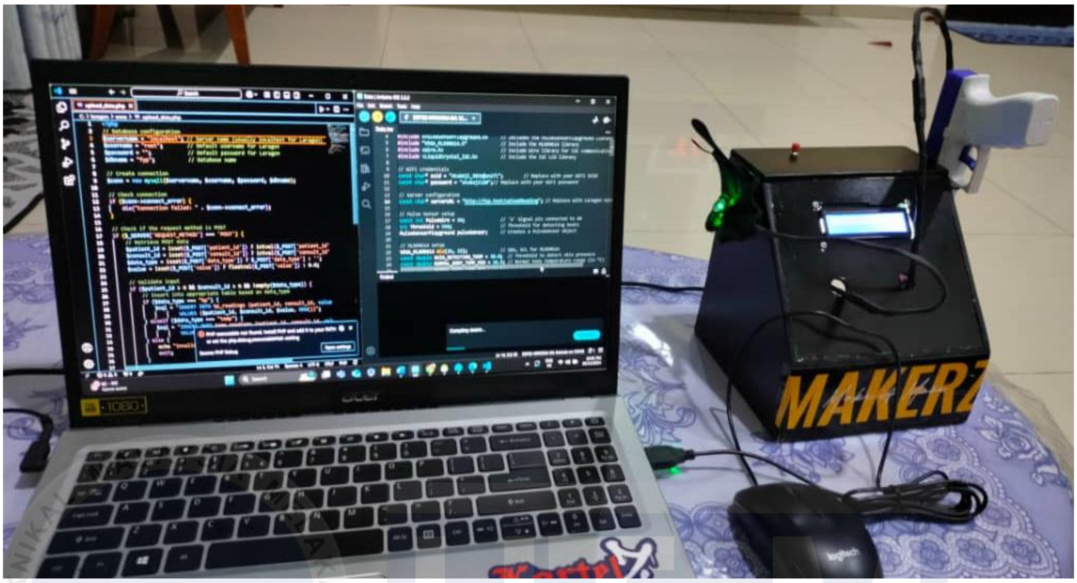


Figure 4.22 Combination Hardware and Software

The function of Laragon in overseeing the backend software, specifically the database that houses recorded health data for analysis, is depicted in Figure 4.22. A thorough assessment of individual health patterns is supported by the database's ability to identify trends in users' health metrics over time. Visual Studio is used for the system's development and debugging, which guarantees smooth hardware and software integration. Also, the real-time data updates and remote accessibility are made possible by the ESP32's Wi-Fi capability, which makes it easier to synchronize data with the database or distant platforms. This health monitoring system offers a data-driven approach to managing personal health by combining real-time measurements, reliable data storage, and user-friendly features. By analysing past and current health data, this promotes proactive care, raises health awareness, and aids in making well-informed decisions.

4.3 Summary

The project "Development of Real-time Patient Health Tracker Website Using Microcontroller and Laragon" is presented in this chapter along with its discussions and findings. The project's goal is to create a comprehensive system that can monitor patients' vital signs in real time and display the information on a user-friendly website. The system sends data to a web server set up with Laragon using microcontrollers to collect data from a variety of patient-affixed sensors, such as blood pressure, temperature, and heart rate monitors. Patients can register and log in, view their health metrics, and connect with licensed doctors using the website's user-friendly interface. Doctors can also track symptoms, diagnose patients, and view real-time patient health metrics with this system. Also, the project's outcomes are shown in this chapter, which includes the landing page, login, registration, about me, patients, readings, reading information, and symptom pages of the website. The simulation results are also covered in this chapter, along with how to connect a heart rate monitor to an ESP32-WROOM, code for Arduino IDE, and connect to Database Laragon.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In the conclusion, utilizing Laragon and a microcontroller to create a real-time patient health tracker website has shown to be an effective strategy for overcoming the shortcomings of conventional healthcare systems. The system has proven to be capable of gathering, sending, storing, and visualizing patient health data in real-time, which allows for timely intervention and customized care. The system can reliably and efficiently gather and process sensor data thanks to the usage of microcontroller technology. The management of patient health data has been made easier by the integration of Laragon, assuring safe and effective storage, retrieval, and analysis. Additionally, the system's interfaces have been made to be user-friendly for both healthcare providers and patients. The system's capacity to deliver efficient and timely care has been further improved by the use of alerts and notifications. All things considered, the real-time patient health tracker website has proven that it has the ability to completely transform remote health monitoring by enabling patients and medical professionals to access vital health information at anytime, anyplace. Due to its affordability, scalability, and dependability, the system holds great promise for broad industry adoption and impact.

5.2 Future Recommendations

Integration of additional sensors:

To give a more complete picture of the patient's health, the system can be enhanced by integrating additional sensors, such as blood pressure, blood glucose, and oxygen saturation sensors. Additionally, the most expensive sensors will provide an accurate reading compared to the least expensive ones.

Better data analysis:

To identify abnormalities and anticipate possible health problems, the system can be enhanced by integrating cutting-edge data analysis techniques, such as machine learning algorithms.

Integration with wearable technology:

To provide real-time health monitoring and alerts, the system can be enhanced by integrating with wearable technology, such as fitness trackers and smartwatches.

Integration with electronic health records:

To give healthcare providers a more comprehensive picture of their patients' medical histories and to help them make better decisions, the system can be enhanced by integrating with electronic health records.

Enhanced security:

To safeguard patient information and maintain privacy, the system can be enhanced by adding cutting-edge security features like encryption and multi-factor authentication.

5.3 Future Works

Potential future works for the real-time patient health tracker website using PHPMyAdmin and a microcontroller:

Integration with other healthcare systems:

Link the system to HIS and EHRs to get a complete picture of the health of your patients.

Algorithms for machine learning:

Use these to identify trends and anticipate possible health problems.

Create a mobile application:

To facilitate effortless access to the system from smartphones.

Integration with wearable technology:

For real-time health monitoring, establish connections with fitness trackers and smartwatches.

Platform for telemedicine:

Enlarge it to incorporate online medical advice and consultations.

Enable patients:

To speak with the system in their own language through natural language processing.

Chatbot:

Provide a chatbot that offers medical guidance and assistance.

REFERENCES

[1] P. Bora, P. Kanakaraja, B. Chiranjeevi, M. Jyothi Sri Sai, and A. Jeswanth, “Smart real time health monitoring system using Arduino and Raspberry Pi,” in *Materials Today: Proceedings*, Elsevier Ltd, 2020, pp. 3855–3859. doi: 10.1016/j.matpr.2021.02.290.

[2] A. Unawane, S. Jadhav, S. Jagtap, and U. G. Students, “E PATIENT MONITORING SYSTEM USING ARDUINO,” 1498. [Online]. Available: www.irjmets.com

[3] R. Sarkar *et al.*, “CarePro: A Complete Arduino and Android-based Elderly Care Health and Security Monitoring System,” in *2021 International Conference on Computational Performance Evaluation, ComPE 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 97–102. doi: 10.1109/ComPE53109.2021.9752318.

[4] P. N. Banu and P. J. MahaVinu Shree, “MULTI-PARAMETER SMART HEALTH MONITORING SYSTEM USING ARDUINO-UNO,” 1684. [Online]. Available: www.irjmets.com

[5] T. Chaikijurajai, L. J. Laffin, and W. H. Wilson Tang, “Artificial intelligence and hypertension: Recent advances and future outlook,” Nov. 01, 2020, *Oxford University Press*. doi: 10.1093/ajh/hpaa102.

[6] R. H. Rangaswamy and B. Larago, “Design and Implementation of Remote Health Monitoring System Using IoT,” 2022. [Online]. Available: https://journals.ddu.edu.et/index.php/HJET

[7] M. M. Islam, A. Rahaman, and M. R. Islam, “Development of Smart Healthcare Monitoring System in IoT Environment,” *SN Comput Sci*, vol. 1, no. 3, May 2020, doi: 10.1007/s42979-020-00195-y.

[8] Vaigai College of Engineering and Institute of Electrical and Electronics Engineers, *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020) : 13-15 May, 2020*.

[9] “REAL-TIME LOCATION TRACKER FOR HEALTH PATIENT IN CRISIS VADDURI AKSHITA, APARNA POLAVARAM, PUJITHA PUDOTA,” *International Research Journal of Engineering and Technology*, 2022, [Online]. Available: www.irjet.net

[10] P. Mell and T. Grance, “The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.”

[11] S. Goyal, “Volume 1.No3 Software as a Service, Platform as a Service, Infrastructure as a Service − A Review,” 2013, [Online]. Available: http://www.ijcsns.com

[12] T. Khan and A. Chakrabarty, “Development of Application based Health Monitoring System using GSM module Anika Tasniem 12101081 Nura Jamil 12101007.”

[13] M. W. Alam, T. Sultana, and M. S. Alam, “A heartbeat and temperature measuring system for remote health monitoring using wireless body area network,” *International Journal of Bio-Science and Bio-Technology*, vol. 8, no. 1, pp. 171–190, Feb. 2016, doi: 10.14257/ijbsbt.2016.8.1.16.

[14] H. Hassan, A. Kamal Taqi, H. J. Hassan, and K. Hadi, “Implementation of Wireless Body Area Network Based Patient Monitoring System,” vol. 8, no. 4, 2018, [Online]. Available: www.iiste.org

[15] H. Narayan and & Rijhi, “A HEARTBEAT DETECTION METHOD BASED ON IOT AND MONITORING SYSTEM USING ARDUINO UNO AND THING-SPEAK.” [Online]. Available: www.tjprc.org

[16] J. Abedalrahim Jamil Alsayaydeh, M. Faizal bin Yusof, M. Zulhakim Bin Abdul Halim, M. Noorazlan Shah Zainudin, and S. Gazali Herawan, “Patient Health Monitoring System Development using ESP8266 and Arduino with IoT Platform.” [Online]. Available: www.ijacsa.thesai.org

[17] M. V Dole and V. V Yerigeri, “ESP8266 BASED HEALTH MONITORING SYSTEM USING ARDUINO,” *International Research Journal of Engineering and Technology*, 2020, [Online]. Available: www.irjet.net

[18] H. Y. Ren, “Design and implementation of web based on Laravel framework,” 2015.

[19] L. De Nardis, G. Caso, and M. G. Di Benedetto, “ThingsLocate: A ThingSpeak-Based Indoor Positioning Platform for Academic Research on Location-Aware Internet of Things,” *Technologies (Basel)*, vol. 7, no. 3, Sep. 2019, doi: 10.3390/technologies7030050.

[20] Sharmad Pasha, “Thingspeak Based Sensing and Monitoring System for IoT with Matlab Analysis.” [Online]. Available: www.ijntr.org

[21] A. Perdana, N. A. Farhana, P. Harliana, and I. M. Karo Karo, “Web-Based Application Development using PHP-Native Framework on Agent of Change Integrity Zone Information System,” *sinkron*, vol. 8, no. 4, pp. 2458–2468, Oct. 2024, doi: 10.33395/sinkron.v8i4.14118.

[22] J. Woo, E. Kim, T. E. Sung, J. Lee, K. Shin, and J. Lee, “Developing an improved risk-adjusted net present value technology valuation model for the biopharmaceutical industry,” *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 5, no. 3, Sep. 2019, doi: 10.3390/joitmc5030045.

[23] Y. Ariyanto, M. Farhan, F. Rachmad, and D. Puspitasari, “Issue 2 Year 2024 Pages 66-73 Jurnal Manajemen Teknologi dan Informatika,” *Matrix: Jurnal Manajemen Teknologi dan Informatika*, vol. 14, pp. 66–73, 2024, doi: 10.31940/matrix.v14i2.66-73.

APPENDICES

Appendix A Gantt Chart of PSM 1

No	Project Activity	Week														
		1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Final Year Project briefing by JK PSM	■							MID SEM BREAK							
2	Topic confirmation and discussion with supervisor	■														
3	Meeting with supervisor for Introduction (Week 1)	■														
4	Study on project background and writing on Chapter 1	■	■	■	■											
5	Meeting with SV for chapter 1 correction and chapter 2 briefing					■										
6	Make a Research and Writing on chapter 2					■	■	■		■						
7	Meeting with SV for chapter 2 corrections and chapter 3 briefing										■	■				
8	Make a Research and Writing on chapter 3 with doing Full Report											■	■	■		
9	Report Draft Submission														■	
10	PSM Presentation															■

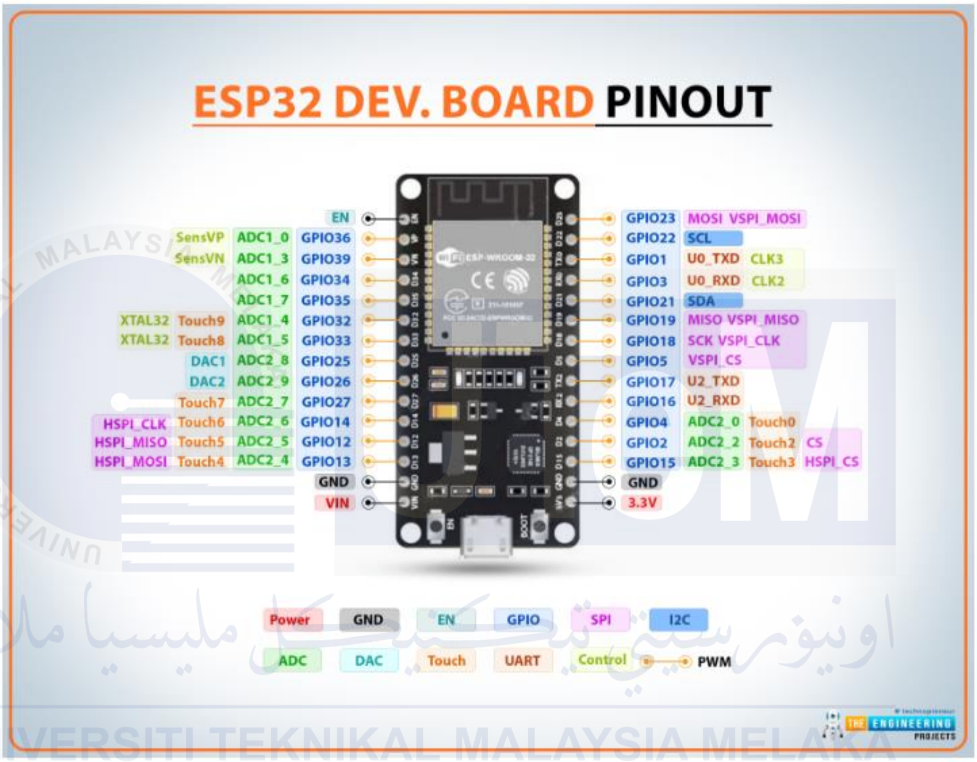
APPENDICES

Appendix B Gantt Chart of PSM 2

No	Project Activity	Week														
		1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Starting Final Year Project for PSM2	■							MID SEM BREAK							
2	Final Year Project briefing by JK PSM	■	■													
3	Meeting with supervisor for Progress (Week 3)			■												
4	PSM Claim Submission (Week 5)			■	■											
5	Progress Work 2 Evaluation (ePSM) (Week 10)					■	■	■		■	■	■				
6	Meeting with SV for Show the results of Chapter 4 and 5									■	■	■				
7	Student BDP2 Report Submission (Week 11)												■			
8	BDP2 Presentation & Evaluation (ePSM) (Week 12)													■		
9	PSM2 Report Evaluation (ePSM) & IIDEX2025 Student Shortlist (Week 13)														■	
10	Final Thesis Submission (eThesis) (Week 14)															■

APPENDICES

Appendix Hardware ESP32



APPENDICES

Coding for Arduino IDE

```
#define USE_ARDUINO_INTERRUPTS true    // Set-up low-level interrupts for most accurate BPM math
#include <PulseSensorPlayground.h>     // Includes the PulseSensorPlayground Library
#include "VEGA_MLX90614.h"             // Include the MLX90614 library
#include <Wire.h>                      // Include Wire library for I2C communication
#include <LiquidCrystal_I2C.h>         // Include the I2C LCD library


// Pulse Sensor setup
const int PulseWire = 34;              // 'S' Signal pin connected to A0
int Threshold = 550;                   // Determine which Signal to "count as a beat" and which to ignore
PulseSensorPlayground pulseSensor;     // Creates a PulseSensor object


// MLX90614 setup
VEGA_MLX90614 mlx(26, 25);             // SDA, SCL for MLX90614
const double SKIN_DETECTION_TEMP = 30.0;  // Threshold to detect skin presence
const double NORMAL_BODY_TEMP_MIN = 36.5; // Normal body temperature range (in °C)
const double NORMAL_BODY_TEMP_MAX = 37.5;


// LCD setup with custom SDA and SCL pins
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);    // Address 0x27, 16 columns, 2 rows


// Button setup
const int buttonPin = 21;              // Button connected to digital pin 21
int lastButtonState = HIGH;            // Variable to store the previous button state


// Patient and Consult ID setup
int patientID = 1;                     // Start with patient ID 1
int consultID = 1;                     // Start with consult ID 1


// State flag for skin detection
bool skinDetected = false;


void setup() {
  Serial.begin(9600);          // Start serial communication
  delay(2000);                 // Allow time for the sensor to stabilize


  // Initialize I2C with custom SDA and SCL pins
  Wire.begin(19, 22);          // SDA = 19, SCL = 22


  // LCD initialization
  lcd.init();                  // Initialize the LCD
  lcd.backlight();             // Turn on the backlight
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");
  delay(2000);                    // Wait for initialization message to display
  lcd.clear();

  // Button setup
  pinMode(buttonPin, INPUT_PULLUP);  // Set button pin as input with internal pull-up resistor

  // Pulse Sensor initialization
  pulseSensor.analogInput(PulseWire);
  pulseSensor.setThreshold(Threshold);

  if (pulseSensor.begin()) {
    Serial.println("PulseSensor object created!");
    lcd.setCursor(0, 0);
    lcd.print("PulseSensor OK!");
  } else {
    Serial.println("PulseSensor failed to initialize.");
    lcd.setCursor(0, 0);
    lcd.print("Pulse FAIL");
  }

  // MLX90614 initialization
  Serial.println("+-----[ MLX90614 Temp Sensor ]-----+");
  lcd.setCursor(0, 1);

  lcd.print("Temp Sensor OK!");

  delay(2000);

  lcd.clear();

}


void loop() {

  // Check button press to switch patient and consult IDs

  int buttonState = digitalRead(buttonPin); // Read the button state

  if (buttonState == LOW && lastButtonState == HIGH) { // Detect button press

    patientID++;                // Increment patient ID

    consultID = patientID;      // Synchronize consult ID with patient ID

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("ID: ");

    lcd.print(patientID);       // Display the current patient ID

    lcd.setCursor(0, 1);

    lcd.print("Consult: ");

    lcd.print(consultID);       // Display the consult ID

    skinDetected = false;       // Reset the skin detection flag for the new patient

    delay(500);                 // Debounce delay

  }

  lastButtonState = buttonState;   // Update the last button state
  // Read MLX90614 temperatures
  double ambientTemp = mlx.mlx90614ReadAmbientTempC();  // Read ambient
temperature
  double targetTemp = mlx.mlx90614ReadTargetTempC();    // Read target temperature

  // Skin detection and display temperature only once
  if (!skinDetected && targetTemp >= SKIN_DETECTION_TEMP) {
    skinDetected = true; // Set flag to true to indicate detection

    double adjustedTargetTemp = targetTemp + 2.0; // Adjust to approximate body
temperature
    Serial.print("Skin detected for Patient ID: ");
    Serial.println(patientID);

    // Display patient ID, consult ID, and temperature on LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ID: ");
    lcd.print(patientID);
    lcd.print(" C: ");
    lcd.print(consultID);

    lcd.setCursor(0, 1);
    lcd.print("Temp: ");
    lcd.print(adjustedTargetTemp, 1); // Display temperature with 1 decimal place


    // Log temperature range

    if (adjustedTargetTemp >= NORMAL_BODY_TEMP_MIN &&

adjustedTargetTemp <= NORMAL_BODY_TEMP_MAX) {

      Serial.println("Body temperature is normal.");

    } else if (adjustedTargetTemp < NORMAL_BODY_TEMP_MIN) {

      Serial.println("Body temperature is low. Possible hypothermia or

undercooling.");

    } else {

      Serial.println("Body temperature is high. Possible fever or hyperthermia.");

    }

  }


  // Pulse Sensor processing

  int myBPM = pulseSensor.getBeatsPerMinute(); // Calculates BPM


  if (pulseSensor.sawStartOfBeat()) {         // Check if a beat happened

    Serial.println("♥  A HeartBeat Happened ! ");

    Serial.print("BP: ");

    Serial.println(myBPM);                    // Print the BPM value


    lcd.setCursor(0, 0);                      // Display BPM on the LCD

    lcd.print("BP: ");
    lcd.print(myBPM);

    lcd.print("   ");                    // Clear remaining characters

  }



  delay(2000); // Delay to avoid flickering updates on the LCD

}
```

Coding connection ARDUINO IDE to LARAGON

```
#include <WiFi.h>

#include <HTTPClient.h>

#include <PulseSensorPlayground.h>

#include "VEGA_MLX90614.h"

#include <Wire.h>

#include <LiquidCrystal_PCF8574.h>

#include <ArduinoJson.h>



// Wi-Fi credentials

const char* ssid = "Realme1";

const char* password = "040601sophia";



// API endpoints

const char* apiGetID = "http://192.168.243.68/api/getID";

const char* apiUploadTP = "http://192.168.243.68/api/uploadTP";

const char* apiUploadBP = "http://192.168.243.68/api/uploadBP";
```

```
// Function prototypes

void sendTempToServer(double temp);

void sendBPToServer(int bp[], int second[], int size);

void fetchTemperature();

void startBPRecording();

void recordBPData();

void fetchIDs();

void handleButtonPress(int pressCount);



// Pulse Sensor setup

const int PulseWire = 34; // 'S' Signal pin connected to A0

int Threshold = 550;

PulseSensorPlayground pulseSensor;



// MLX90614 setup

VEGA_MLX90614 mlx(26, 25);

const double SKIN_DETECTION_TEMP = 30.0;



// LCD setup

LiquidCrystal_PCF8574 lcd(0x27);



// Button setup

const int buttonPin = 21;
```

```
int lastButtonState = HIGH;

// Patient and Consult IDs
int patientID = 1;
int consultID = 1;
int buttonPressCount = 0;

// BP Recording setup
const int RECORD_DURATION = 30; // Duration in seconds
int bpData[30];
int seconds[30];
int bpIndex = 0;
unsigned long startTime = 0;
bool isRecordingBP = false;

void setup() {
    Serial.begin(9600);
    Wire.begin(19, 22); // SDA, SCL for I2C

    lcd.begin(16, 2);
    lcd.setBacklight(255);
    lcd.setCursor(0, 0);
    lcd.print("Connecting WiFi");
```

```
// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
}
Serial.println("\nWiFi connected");
lcd.setCursor(0, 1);
lcd.print("WiFi Connected");

delay(2000);
lcd.clear();

// Pulse sensor setup
pulseSensor.analogInput(PulseWire);
pulseSensor.setThreshold(Threshold);
if (pulseSensor.begin()) {
   Serial.println("PulseSensor initialized!");
} else {
   Serial.println("PulseSensor failed to initialize!");
}

// Button setup
pinMode(buttonPin, INPUT_PULLUP);
```

```
    // Fetch initial IDs
    fetchIDs();
}

void loop() {
    int buttonState = digitalRead(buttonPin);
    if (buttonState == LOW && lastButtonState == HIGH) {
        buttonPressCount++;
        handleButtonPress(buttonPressCount);
        delay(500); // Debounce delay
    }
    lastButtonState = buttonState;

    // Handle BP recording
    if (isRecordingBP) {
        recordBPData();
    }

    delay(500);
}

void handleButtonPress(int pressCount) {
    switch (pressCount % 4) {
```

```
case 1:
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Fetching Temp");
  fetchTemperature();
  break;

case 2:
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Recording BP");
  startBPRecording();
  break;

case 3:
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Fetching IDs");
  fetchIDs();
  break;

case 0:
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
            lcd.print("Next Patient");

            lcd.setCursor(0, 1);

            lcd.print("ID: " + String(patientID));

            break;

    }

}


void fetchIDs() {

    if (WiFi.status() == WL_CONNECTED) {

        HTTPClient http;

        http.begin(apiGetID);

        int httpResponseCode = http.GET();


        if (httpResponseCode > 0) {

            String response = http.getString();

            Serial.println("Response: " + response);


            DynamicJsonDocument doc(512);

            DeserializationError error = deserializeJson(doc, response);


            if (error) {

                Serial.print("JSON Deserialization failed: ");

                Serial.println(error.f_str());

            } else {
```

```
        patientID = doc["p_id"];
        consultID = doc["c_id"];

        Serial.println("Patient ID: " + String(patientID));
        Serial.println("Consult ID: " + String(consultID));

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("p_id: " + String(patientID));
        lcd.setCursor(0, 1);
        lcd.print("c_id: " + String(consultID));
      }
    } else {
      Serial.println("HTTP Error: " + String(httpResponseCode));
    }
    http.end();
  } else {
    Serial.println("Wi-Fi not connected");
  }
}

void fetchTemperature() {
  double targetTemp = mlx.mlx90614ReadTargetTempC();
  Serial.println("Raw Temperature: " + String(targetTemp)); // Debugging log
  if (targetTemp >= SKIN_DETECTION_TEMP) {
    double adjustedTemp = targetTemp + 2.0;
    sendTempToServer(adjustedTemp);
    Serial.println("Adjusted Temperature: " + String(adjustedTemp));
  } else {
    Serial.println("Temperature below threshold: " +
String(SKIN_DETECTION_TEMP));
    sendTempToServer(targetTemp); // Uncomment for testing purposes
  }
}

void startBPRecording() {
  isRecordingBP = true;
  startTime = millis();
  bpIndex = 0;
  Serial.println("Started BP recording");
}


void recordBPData() {
  int bpm = pulseSensor.getBeatsPerMinute();
  unsigned long currentTime = millis();
  int elapsedSeconds = (currentTime - startTime) / 1000;
  if (elapsedSeconds < RECORD_DURATION) {
    if (pulseSensor.sawStartOfBeat() && bpIndex < RECORD_DURATION) {
      bpData[bpIndex] = bpm;
      seconds[bpIndex] = elapsedSeconds;
      bpIndex++;

      Serial.println("BPM: " + String(bpm) + ", Time: " + String(elapsedSeconds) +
"s");
    }
  } else {
    isRecordingBP = false;
    sendBPToServer(bpData, seconds, bpIndex);
  }
}

void sendTempToServer(double temp) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(apiUploadTP);
    http.addHeader("Content-Type", "application/json");

    String payload = String("{\"c_id\":\"") + consultID + "\",\"p_id\":\"" + patientID +
"\",\"temp\":\"" + temp + "\"}";
    http.addHeader("X-HTTP-Method-Override", "PATCH");
    int httpResponseCode = http.POST(payload);


    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println("Response: " + response);
    } else {
      Serial.println("HTTP Error Code: " + String(httpResponseCode));
    }
    http.end();
  } else {
    Serial.println("Wi-Fi not connected");
  }
}


void sendBPToServer(int bp[], int second[], int size) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(apiUploadBP);
    http.addHeader("Content-Type", "application/json");
    http.addHeader("X-HTTP-Method-Override", "PATCH");


    DynamicJsonDocument doc(1024);
    JsonArray timeArray = doc.createNestedArray("time");
    JsonArray bpArray = doc.createNestedArray("reading");
  for (int i = 0; i < size; i++) {
    timeArray.add(String(second[i]));
    bpArray.add(String(bp[i]));
  }

  doc["c_id"] = String(consultID);
  doc["p_id"] = String(patientID);

  String payload;
  serializeJson(doc, payload);

  int httpResponseCode = http.POST(payload);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Response: " + response);
  } else {
    Serial.println("HTTP Error Code: " + String(httpResponseCode));
  }
  http.end();
} else {
  Serial.println("Wi-Fi not connected");
}
```

```
}
```

Code PHP Script

```
<?php

namespace App\Http\Controllers;

use App\Models\BpReading;
use App\Models\Consult;
use App\Models\TempReading;
use Illuminate\Http\Request;

class ApiController extends Controller
{
    //
    public function getID()
    {

        try {
            // Fetch the record by ID and select the specified columns
        $data = Consult::where('status', 'upload')
            ->orderBy('created_at', 'asc')
            ->select('id', 'p_id')
            ->first();

        if (!$data) {
            // Return a 404 response if the record is not found
            return response()->json(['error' => 'No consultation found'], 404);
        }

        // Return the selected columns
        return response()->json([
            'c_id' => $data['id'],
            'p_id' => $data['p_id']
        ], 200);
    } catch (\Exception $e) {
        // Handle any unexpected errors
        return response()->json([
            'error' => 'Failed to retrieve the record',
            'message' => $e->getMessage(),
        ], 500);
    }
}
public function reading(Request $request)
```

```
{
    try {
        $cred = $request->input(
            'c_id',  // validation for 'consult ID'
            'p_id',  // validation for 'consult ID'
        );

        $temperature = $request->validate([
            'temp' => 'required|string',  // validation for 'temperature'
        ]);

        $validated = $request->validate([
            'time' => 'required|array',             // 'time' must be an array
            'reading' => 'required|array',          // 'reading' must be an array
            'time.*' => 'required|string',          // Each 'time' must be a string
            'reading.*' => 'required|string',       // Each 'reading' must be a string
        ]);

        // Create a new item with the validated data
        $consult = TempReading::create([
            'c_id' => $cred['c_id'],
            'p_id' => $cred['p_id'],
            'temp' => $temperature['temp'],
        ]);
        // Ensure the arrays have the same length
        if (count($validated['time']) !== count($validated['reading'])) {
            return response()->json([
                'error' => 'Time or reading is missing.',
            ], 400); // Bad Request
        }
        // Create readings from time and reading arrays
        $readings = array_map(function ($time, $reading) use ($cred) {
            return [
                'c_id' => $cred['c_id'],
                'p_id' => $cred['p_id'],
                'time' => $time,
                'reading' => $reading,
                'created_at' => now(),
            ];
        }, $validated['time'], $validated['reading']);


        $bp = BpReading::insert($readings);
        // Return the generated ID as a response
        return response()->json(['message' => 'Successfully Uploaded'], 201);
    } catch (\Exception $e) {
        // Return a failure response if something goes wrong
        return response()->json([
            'error' => 'Failed to upload!',
            'message' => $e->getMessage(),
        ], 500); // Internal Server Error
    }
}

public function uploadTP(Request $request)
{
    try {
        $temperature = $request->validate([
            'c_id' => 'required|string',  // validation for 'consult ID'
            'p_id' => 'required|string',  // validation for 'consult ID'
            'temp' => 'required|string',  // validation for 'temperature'
        ]);
        // Create a new item with the validated data
        $consult = TempReading::create([
            'c_id' => $temperature['c_id'],
            'p_id' => $temperature['p_id'],
            'temp' => $temperature['temp'],
        ]);

        // Return the generated ID as a response
        return response()->json(['id' => $consult->id], 201);
    } catch (\Exception $e) {
        // Return a failure response if something goes wrong
        return response()->json([
            'error' => 'Failed to create item',
            'message' => $e->getMessage(),
        ], 500); // Internal Server Error
    }
}

public function uploadBP(Request $request)
{
    try {
        $validated = $request->validate([
            'c_id' => 'required|string',           // Validation for 'consult ID'
            'p_id' => 'required|string',           // Validation for 'patient ID'
            'time' => 'required|array',            // 'time' must be an array
            'reading' => 'required|array',          // 'reading' must be an array
            'time.*' => 'required|string',          // Each 'time' must be a string
            'reading.*' => 'required|string',       // Each 'reading' must be a string
        ]);
        // Ensure the arrays have the same length
        if (count($validated['time']) !== count($validated['reading'])) {
            return response()->json([
                'error' => 'Time or reading is missing.',
            ], 400); // Bad Request
        }


        // Create readings from time and reading arrays
        $readings = array_map(function ($time, $reading) use ($validated) {
            return [
                'c_id' => $validated['c_id'],
                'p_id' => $validated['p_id'],
                'time' => $time,
                'reading' => $reading,
                'created_at' => now(),
            ];
        }, $validated['time'], $validated['reading']);
        // Assuming you have a 'Reading' model for storing the readings
        $bp = BpReading::insert($readings);
        Consult::whereKey($validated['c_id'])->update(['status' => 'new']);
        // Return the generated ID as a response
        return response()->json(['message' => 'Data successfully added!'], 201);
    } catch (\Exception $e) {
        // Return a failure response if something goes wrong
        return response()->json([
            'error' => 'Failed to create item',
            'message' => $e->getMessage(),
        ], 500); // Internal Server Error
    }
```

```
   }

}
```