

**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**  
**University M'Hamed BOUGARA – Boumerdes**



**Institute of Electrical and Electronic Engineering**  
**Department of Electronic**

Project Report Presented in Partial Fulfilment of  
the Requirements of the Degree of

**'Master'**  
**In Computer Engineering**

Title:

**Design and implementation of an Embedded  
AIOT-based Home Hospitalization System**

Presented By:

- **Mr. REZRAZI Mohamed Seif Eddine**
- **Mr. Djelti Mohamed Amine**

Supervisor:

**Dr. Touzout W**

## Abstract

In recent years, the rise in chronic diseases among the elderly has emphasized the need for innovative healthcare solutions. The COVID-19 pandemic has further highlighted the challenges faced by hospitals in accommodating a surge of patients while ensuring the safety of healthcare providers. To address these challenges, home hospitalization has emerged as a viable alternative. The Artificial Intelligence of Things (AIoT) based home hospitalization system integrates artificial intelligence, internet of things (IoT), sensors, and mobile/web applications to enable remote monitoring and management of patients' health conditions. The system includes a hardware device with sensors for accurate data collection, a mobile application for patients to access health information and communicate with doctors, a web dashboard for doctors to manage patient data and provide personalized recommendations, and an AI model that analyze patient data and predict health conditions. The system employs a secure and reliable communication protocol for efficient data transmission. The primary objective of this system is to provide convenient and accessible healthcare services, particularly for the elderly and individuals with limited access to hospitals. By offering remote consultations and monitoring, the system reduces the need for physical travel and ensures timely medical attention. The integration of AI and IoT technologies strengthens the system's ability to support doctors in making informed decisions.

## *Acknowledgements*

*In the name of Allah, the Most Gracious, the Most Merciful.*

*We humbly praise and thank Allah, the Almighty, for granting us the wisdom, perseverance, and strength to successfully complete this project. It is through His divine guidance and blessings that we have been able to overcome challenges and reach this moment of accomplishment.*

*We are eternally grateful to our beloved families, whose unwavering love, support, and financial assistance have been the driving force behind our journey. To our parents, whose constant encouragement and belief in our abilities have been the foundation of our success, we offer our heartfelt appreciation.*

*Our sincere gratitude goes to our supervisor, Dr. TOUZOUT Walid, for his invaluable guidance, mentorship, and unwavering support throughout this project.*

*We would like to extend a special thanks to our dear friends, Abdelmadjid and Charaf Eddine, whose impact and contributions have enriched this endeavor. We are also grateful to Lotfi, Tarek, Oumnia, Lamine, and Abdenour for their unwavering support and encouragement along the way.*

*Lastly, we would like to express our gratitude to the members of the Wameedh Scientific Club, both past and present, for their collective efforts and collaborative spirit.*

*May Allah bless each and every individual who has supported us on this journey, and may this project be a source of benefit and inspiration for all.*

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Abbreviations</b>	<b>x</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Background and Motivation . . . . .	11
1.2 Problem Statement . . . . .	12
1.3 Objectives . . . . .	13
1.4 Structure of The Report . . . . .	13
<b>2 Background</b>	<b>14</b>
2.1 Overview of AIoT in Healthcare . . . . .	14
2.2 Home Hospitalization Systems . . . . .	14
2.2.1 Existing Solutions . . . . .	15
2.3 Relevant Technologies and Concepts . . . . .	16
2.3.1 Internet of Things . . . . .	16
2.3.2 ESP32 Microcontroller and Sensor Integration . . . . .	17
2.3.2.1 ESP32 Microcontroller . . . . .	17
2.3.2.2 FreeRTOS . . . . .	17
2.3.2.3 ESP32 NVS . . . . .	17
2.3.2.4 ESP-NETIF . . . . .	17
2.3.2.5 UART Communication . . . . .	18
2.3.2.6 I2C Communication . . . . .	18
2.3.2.7 One-Wire Protocol . . . . .	18
2.3.2.8 Heartbeat Measurement . . . . .	18
2.3.2.9 Blood Pressure Measurement . . . . .	19
2.3.2.10 Glucose Measurement . . . . .	20
2.3.2.11 Body Temperature Measurement . . . . .	20
2.3.3 Protocols for Data Transmission . . . . .	20
2.3.3.1 Transmission Control Protocol . . . . .	20



2.3.3.2	User Datagram Protocol . . . . .	21
2.3.4	Flutter Framework . . . . .	21
2.3.4.1	Asynchronous Programming . . . . .	21
2.3.4.2	State management . . . . .	21
2.3.4.3	Business Logic Component . . . . .	22
2.3.4.4	Clean Architecture with Bloc . . . . .	22
2.3.5	Database Integration with Firebase . . . . .	22
2.3.5.1	Firestore Authentication . . . . .	23
2.3.5.2	Firestore Firestore . . . . .	23
2.3.6	AI Models for Disease Prediction in Healthcare . . . . .	23
<b>3</b>	<b>Design and Implementation</b>	<b>26</b>
3.1	System Architecture and Design . . . . .	26
3.1.1	Communication Setup between Hardware and Mobile Application	28
3.1.1.1	Protocol OpCodes . . . . .	28
3.1.1.2	Protocol Workflow . . . . .	29
3.1.2	Hardware Components and Functionalities . . . . .	30
3.1.2.1	Reverse Engineering Blood Pressure . . . . .	32
3.1.3	Firmware Design and Implementation . . . . .	33
3.1.3.1	Device Interface Server . . . . .	34
3.1.3.2	Network Configuration Setup Process . . . . .	35
3.1.3.3	Measurement Process and Task Execution . . . . .	35
3.1.3.4	Heart Rate Data Acquisition . . . . .	37
3.1.3.5	Body Temperature Data Acquisition . . . . .	37
3.1.4	Mobile Application Design and Features . . . . .	38
3.1.4.1	Authentication and User Management . . . . .	38
3.1.4.2	Connection Setup with Hardware . . . . .	38
3.1.4.3	Sensor Control and Data Visualization . . . . .	39
3.1.4.4	Data Storage with Firestore Firestore . . . . .	40
3.1.4.5	History Viewing and Graphical Representations . . . . .	40
3.1.4.6	Appointment Requests and Chat Functionality . . . . .	41
3.1.5	Web Development for Doctor's Dashboard . . . . .	41
3.1.5.1	Authentication and User Management . . . . .	41
3.1.5.2	Patient Management and Data Visualization . . . . .	42
3.1.5.3	Appointment Requests and Chat Functionality . . . . .	42
3.1.5.4	Notifications and Alert System . . . . .	42
3.2	AI Model Development for Disease Prediction . . . . .	43
3.2.1	Dataset Selection and Preprocessing . . . . .	43
3.2.2	AI Model Architecture . . . . .	44
3.2.2.1	Transformer-Encoder Layer . . . . .	44
3.2.2.2	Dense Layers . . . . .	45
3.2.2.3	Benefits of the Architecture . . . . .	46
3.2.3	AI Model Training and Validation . . . . .	47
3.2.3.1	Data Split . . . . .	47
3.2.3.2	Training Process . . . . .	47

3.2.3.3	Learning Rates . . . . .	47
3.2.3.4	Optimizers . . . . .	48
3.2.3.5	Validation . . . . .	48
3.2.4	Integration with the System . . . . .	48
<b>4</b>	<b>Results and Discussion</b>	<b>50</b>
4.1	Overview . . . . .	50
4.2	Usability Evaluation . . . . .	50
4.3	Hardware Functionality Evaluation . . . . .	52
4.3.1	Power Consumption Testing and Analysis . . . . .	52
4.3.2	Reliability Testing . . . . .	52
4.4	Firmware Functionality Testing and Validation . . . . .	53
4.4.1	Optimizing Stack Size Allocation for Task Efficiency . . . . .	53
4.4.2	A Comparative Analysis of Our Sensor and Other Established Sensors . . . . .	53
4.4.2.1	Heart Rate Sensor Accuracy Evaluation . . . . .	54
4.4.3	Error Handling Testing . . . . .	54
4.5	AI Model Evaluation and Disease Prediction Results . . . . .	55
4.5.1	Training Performance Metrics: Accuracy and Loss Curves . . . . .	55
4.5.2	Confusion Matrix . . . . .	56
4.6	Discussion . . . . .	57
4.6.1	Diabetes Prediction Model . . . . .	57
4.6.2	Effectiveness of the AIoT System in Home Healthcare . . . . .	58
4.6.3	Limitations of the System . . . . .	59
4.6.4	Future Enhancements and Potential Exploration Directions . . . . .	59
	<b>Appendices</b>	<b>66</b>
<b>A</b>	<b>Physical Implementation of our System</b>	<b>67</b>
<b>B</b>	<b>Mobile Application Screenshots</b>	<b>68</b>
<b>C</b>	<b>Web Dashboard Screenshots</b>	<b>71</b>

# List of Figures

2.1	Working principle of PPG sensors. . . . .	19
3.1	System Architecture Diagram . . . . .	27
3.2	Protocol Handshake Workflow . . . . .	30
3.3	Hardware Components Diagram . . . . .	31
3.4	Physical Implementation of the Circuit . . . . .	32
3.5	Blood Pressure Sensor PCB . . . . .	32
3.6	System Initialization Flowchart . . . . .	34
3.7	Flowchart of the Measurement Opcode Handling Process in the Device Interface Server. . . . .	36
3.8	Diagram Illustrating the Interaction between UI, BLoC, and Hardware.	39
3.9	Transformer-Encoder Layer . . . . .	45
3.10	Illustration of the Dense Layers . . . . .	46
4.1	The System Usability Scale (SUS) . . . . .	51
4.2	Heart Rate Sensor Accuracy Evaluation . . . . .	54
4.3	Training Performance Metrics: Accuracy and Loss Curves . . . . .	56
4.4	Confusion Matrix . . . . .	56
A.1	Physical Implementation of our System . . . . .	67
B.1	Sign up process screens . . . . .	68
B.2	Handshake Process Screens . . . . .	69
B.3	Data History Screens . . . . .	69
B.4	Data Measurements Screens . . . . .	70
C.1	Doctor Sign up screen . . . . .	71
C.2	Doctor Dashboard screen . . . . .	72
C.3	Doctor Patients screen . . . . .	72
C.4	Doctor History screen (Graph) . . . . .	73
C.5	Doctor History screen (Table) . . . . .	73

# List of Tables

3.1	Supported OpCodes . . . . .	29
4.1	Patients' SUS score . . . . .	51

# List of Abbreviations

**ADC** Analog-to-Digital Converter.

**AI** Artificial intelligence.

**AIOT** Artificial Intelligence of Things.

**AP** Access Point.

**API** Application Programming Interface.

**BLoC** Business Logic Component.

**DHCP** Dynamic Host Configuration Protocol.

**EEPROM** Electrically Erasable Programmable Read-Only Memory.

**FN** False Negative.

**FP** False Positive.

**GeLU** Gaussian Error Linear Unit.

**I2C** Inter-Integrated Circuit.

**IDF** IoT Development Framework.

**IOT** Internet of Things.

**IR** Infrared.

**LED** Light-Emitting Diode.

**MAP** Mean Arterial Pressure.

**ML** Machine Learning.

**NETIF** Network Interface.

**NVS** Non Volatile Storage.

**OWB** One Wire Bus.

**PCB** Printed Circuit Board.

**PPG** Photoplethysmography.

**ReLU** Rectified Linear Unit.

**RTOS** Real-Time Operating System.

**SCL** Serial Clock.

**SDA** Serial Data.

**SDK** Software Development Kit.

**SSID** Service Set Identifier.

**SUS** System Usability Scale.

**TCP** Transmission Control Protocol.

**TN** True Negative.

**TP** True Positive.

**UART** Universal Asynchronous Receiver / Transmitter.

**UDP** User Datagram Protocol.

**UI** User Interface.

# Chapter 1

## Introduction

The healthcare industry is experiencing a rapid transformation due to technological advancements, and one of the most promising innovations is the integration of Artificial Intelligence of Things (AIoT) systems. AIoT combines Artificial Intelligence (AI) and the Internet of Things (IoT) to create intelligent and interconnected healthcare solutions.

In the context of home hospitalization, AIoT systems have the potential to revolutionize healthcare delivery, allowing patients to receive personalized and efficient care from their homes. The AIoT home hospitalization system is designed to overcome the limitations of traditional healthcare models, which often involve frequent hospital visits, limited accessibility, and delayed interventions.

By leveraging IoT technologies such as sensors and wearable devices, the system enables continuous monitoring of patients' vital signs and health parameters. This real-time data is collected and transmitted to healthcare providers, allowing for timely interventions and proactive healthcare management. The integration of AI algorithms within the AIoT home hospitalization system further enhances its capabilities.

AI models can analyze the collected data, detect patterns, and make predictions regarding the patient's health status. The benefits of the AIoT home hospitalization system include enabling patients to receive high-quality healthcare services while staying in the familiar and comfortable environment of their homes, providing healthcare providers with access to real-time patient data, allowing for remote monitoring and timely interventions, and enhancing the accuracy of disease detection and prediction, enabling early interventions and better health outcomes.

### 1.1 Background and Motivation

The COVID-19 pandemic has affected many countries worldwide, causing thousands of infected cases and deaths. The virus spreads through direct contact, respiratory droplets, and touching surfaces contaminated with the virus[1]. The elderly and people with chronic diseases are at a higher risk of contracting the virus[2]. Hospitals

worldwide have struggled to accommodate the large number of infected people, and the virus has started to spread among medical and paramedical teams, posing a risk to patients staying in hospitals. The world has additionally witnessed a significant increase in the number of elderly people, with the World Health Organization projecting that the number of people aged 60 and over will rise from 900 million to around 2 billion between 2015 and 2050, increasing patient dependency, chronic diseases and disability[3], as well as the large financial burdens borne by the economies of the countries involved.

Given the significant increase in the number of elderly people who often suffer from chronic diseases and require hospitalization, and the rapid spread of the coronavirus, we believe that home hospitalization must be adopted by governments to limit the spread of any new virus of this kind and maintain the health of patients who require a hospital stay.

Home hospitalization is a smart and pioneering model of health care that aims to alleviate the suffering of patients, particularly the elderly, by avoiding the inconvenience of moving to hospital institutions for treatment and allowing them to receive continuous care in the comfort of their homes.

## 1.2 Problem Statement

The project aims to address the challenges faced by healthcare systems, particularly in the context of home hospitalization, which have become more evident in the wake of the COVID-19 pandemic. The spread of the virus has placed an immense strain on hospitals and medical facilities, leading to overcrowding, limited resources, and compromised patient care. Additionally, the growing elderly population and the increasing prevalence of chronic diseases have further intensified the demand for healthcare services. These factors have highlighted the need for innovative solutions that can provide efficient and effective healthcare delivery, particularly in home settings. Therefore, the problem statement for this project is to develop an AIoT home hospitalization system that leverages advanced technologies, including AI, IoT, and mobile applications, to enable remote patient monitoring, real-time data collection, and AI-driven analysis. The system aims to enhance patient care, alleviate the burden on healthcare facilities, and improve the overall quality of healthcare services provided to individuals in their homes.



## 1.3 Objectives

The objectives of the work are the following:

- Create a dependable and user-friendly mobile application that allows patients to monitor their health metrics at the comfort of their own homes.
- Create a hardware system comprised of sensors, microcontroller, and communication modules to allow for seamless data transmission between the sensors and the mobile application.
- Establish a secured and efficient UDP based communication protocol to allow for real-time data transmission between the mobile app and the hardware system, providing accurate and timely monitoring of patients' health status.
- Create a web-based dashboard for doctors that allows them to remotely monitor, analyze, and make decisions about their patients' data, measures, and historical trends.
- Develop an AI model capable of analyzing the collected health data and making accurate predictions of cardiac problems for early diagnosis, allowing for timely intervention and proactive healthcare management.
- Evaluate the AIoT home hospitalization system's performance and usability, including the mobile application, hardware integration, communication reliability, doctor's dashboard, AI model accuracy, and user satisfaction.

## 1.4 Structure of The Report

The structure of our report is designed to provide a comprehensive understanding of our home hospitalization system. It is organized into several key sections. The Introduction chapter provides an overview of the objectives and the significance of remote patient monitoring. The Background section delves into the theoretical foundations, covering topics such as AI and machine learning, hardware components utilized in our system, and communication protocols. The Design and Implementation chapter explores the detailed design and development process of our home hospitalization system, including the integration of commercial sensors and the creation of the mobile application and web dashboard. In the Tests and Results chapter, the evaluation process is described, highlighting the conducted tests on hardware functionality, firmware performance, and disease prediction accuracy. The Discussion section critically analyzes the obtained results, considering the limitations and potential for improvement. Finally, the Conclusion chapter summarizes our main findings, addresses the objectives, and proposes future directions. The report concludes with a list of references and any relevant appendices containing supporting materials.

# Chapter 2

## Background

### 2.1 Overview of AIoT in Healthcare

The AIoT, which is a combination of artificial intelligence technologies and the Internet of Things infrastructure, is making healthcare smarter and smarter intending to improve human-machine interaction and enhance data management practices. The AIoT has the potential to revolutionize healthcare by enabling personalized medicine, improving the methods of treatment, doing predictive analysis of disease, monitoring patients, and providing timely detection of health issues. With AIoT, physicians can access near real-time dashboards that give a complete picture of a patient's health metrics, such as heart rate, oxygen saturation, and body temperature, and receive intelligent health analysis reports. These reports are generated using AI that analyzes patient data acquired[4].

The AIoT can also be used to develop smart healthcare systems that can improve healthcare services for the elderly. AIoT-based healthcare services utilized in the medical industry can be classified as electronic health and telecare networks, diagnosis, prevention, rehabilitation, and monitoring devices[5]. AIoT applications can potentially turn a smartphone into a health detection device to assist patients. The AIoT can also be used to identify asymptomatic Covid patients. The AIoT has the potential to improve healthcare outcomes and reduce healthcare costs by leveraging digital technology and machine learning as tools to deliver personalized medicine[6].

### 2.2 Home Hospitalization Systems

Hospital-at-home programs, commonly referred to as home hospitalization systems, allow some patients who require acute care to receive treatment in their homes. This form of care delivery completely replaces acute hospital treatment by providing hospital-level care in the patient's home. Programs for home hospitalization give patients greater convenience because they can get hospital-quality care while relaxing

in their own homes. When compared to conventional hospital stays, this can be very advantageous for the delivery of healthcare. According to research, patients who receive home care had a lower risk of contracting hospital-acquired infections and being readmitted to the hospital.

Programs for home hospitalization have been found to enhance patient outcomes and lower healthcare costs by 30% or more[7] in addition to offering patients convenience.

### 2.2.1 Existing Solutions

Home hospitalization programs are gaining popularity as a cutting-edge method of healthcare. By offering patients medical care and monitoring in the comfort of their own homes, these initiatives aim to lessen the need for protracted hospital visits. A project that is comparable to ours is illustrated by the paper "A Home Hospitalization System Based on the Internet of Things, Fog Computing, and Cloud Computing" [8]. However, it is crucial to take into account and deal with some restrictions that come with this kind of system.

- The possibility of complexity brought on by the existence of several mobile applications is one restriction of the system discussed in the study. These apps may need additional time and work to design and maintain, which might present problems with coordination and user experience.
- The demand for an equipped environment is another restriction to consider. The installation and integration of numerous devices and sensors in the patient's house may be necessary in order to implement an IoT-based home hospitalization system. This could result in logistical difficulties, such as making sure that there is compatible infrastructure available and taking care of any technological requirements.
- The study also emphasizes the involvement of nurses in the home hospitalization system. The investigation of AI models that can offer advice and support in place of or in addition to human nurses represents a potential area for improvement. Integrating AI models can increase the system's scalability and possibly lessen reliance on a small pool of medical specialists.

We can better comprehend the scope of difficulties and potential areas for development in our own AIoT home healthcare system project by being aware of these limits.

## 2.3 Relevant Technologies and Concepts

In this section, we will explore the relevant technologies and concepts that underpin the development of the embedded AIoT-based home healthcare system. These technologies and concepts encompass a wide range of disciplines, including embedded systems, artificial intelligence (AI), internet of things (IoT), mobile and web development. By understanding these fundamental components, we can gain insights into how they synergistically come together to create an innovative solution that revolutionizes remote healthcare monitoring and management.

### 2.3.1 Internet of Things

The Internet of Things (IoT) is a technology that facilitates interaction between real and virtual objects. It is rapidly developing and transforming traditional systems into scalable, adaptable, and more efficient E-learning systems[9]. IoT is a network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity which enables these objects to connect and exchange data[10].

The history of IoT can be traced back to the early days of radio and television, where the concept of connected devices first emerged. In 1999, Kevin Ashton, a British entrepreneur, coined the term "Internet of Things" during his tenure at Procter and Gamble[11], envisioning a future of interconnected objects. Over time, significant milestones have shaped the development of IoT. The introduction of the Arduino board in 2003 simplified IoT device creation, as it has become a popular choice for IoT projects due to its low cost, ease of use, and flexibility[12]. By 2008, the number of connected devices had already surpassed the global population. As we fast forward to 2021, connected devices have now surpassed unconnected ones on a global scale. This trend is expected to continue at an accelerated pace, with an estimated 70 billion connected devices projected by 2025[13]. Notably, Google's acquisition of Nest for \$3.2 billion highlighted IoT's growing importance in the tech industry. In 2022, the World Economic Forum recognized IoT as one of the most impactful technological advancements[14]. These achievements underscore the continuous evolution and profound impact of IoT on our interconnected world.

IoT has several applications in healthcare, and its adoption is increasing in this field. IoT is being used in healthcare to improve the quality of service, energy efficiency, and security of healthcare systems. It is expected that IoT will continue to play a significant role in healthcare in the future.

## 2.3.2 ESP32 Microcontroller and Sensor Integration

### 2.3.2.1 ESP32 Microcontroller

The ESP32[15] is a system-on-a-chip microcontroller with integrated Wi-Fi and Bluetooth capabilities. The ESP32 offers wireless connectivity, and high-speed dual-core processing, and is known for its affordability and low power consumption, making it a highly suitable choice for IoT applications. The official development framework for the ESP32 platform, or ESP-IDF, offers a selection of libraries and tools for creating programs on the hardware.

ESP-IDF[16] is the official development framework for the ESP32 microcontroller, offering comprehensive support and a rich development environment. It provides low-level control over the ESP32 chip, enabling fine-grained manipulation of hardware features. With its extensive libraries and tools, ESP-IDF simplifies the development process for Wi-Fi, Bluetooth, and networking protocols. It is an excellent choice for IoT applications, combining strong support, documentation, and low-level control.

### 2.3.2.2 FreeRTOS

FreeRTOS[17] is an open-source, real-time operating system kernel designed for embedded devices. The Internet of Things (IoT) and embedded systems both make substantial use of the FreeRTOS for its adaptability, dependability, and capacity for effectively managing real-time tasks.

### 2.3.2.3 ESP32 NVS

The ESP32 NVS[18] (Non-Volatile Storage) is a feature of the ESP32 microcontroller that provides a persistent storage solution. It allows you to store and retrieve data that needs to be retained even when the power is turned off or the device is reset. The NVS in ESP32 is implemented using the flash memory of the microcontroller. It provides a key-value pair storage mechanism, where data is stored and accessed using unique keys. The NVS is organized into namespaces, which allow for logical separation of different sets of data.[16]

### 2.3.2.4 ESP-NETIF

ESP-NETIF[19] is a networking interface library provided by Espressif Systems, specifically designed for their ESP32 and ESP8266 platforms. It serves as an abstraction layer that simplifies the process of connecting and configuring various network interfaces, such as Wi-Fi, Ethernet, or Bluetooth, on these devices.

ESP-NETIF provides a consistent API (Application Programming Interface) for managing network interfaces, allowing developers to easily switch between different connectivity options without changing their application code. The library provides additional features such as network event handling and DHCP (Dynamic Host Configuration Protocol) client and server functionalities.

### **2.3.2.5 UART Communication**

Universal Asynchronous Receiver-Transmitter (UART)[20] is a hardware communication interface commonly used for point-to-point serial communication between devices. UART is a standard protocol used to transmit and receive data between two devices, typically a microcontroller or computer, and another device such as a sensor, peripheral, or another microcontroller.

UART data is transmitted in individual bits without a clock signal governing the timing. It uses two communication lines: one for transmitting data (TX) and one for receiving data (RX). The data is sent in packets or frames, where each frame consists of a start bit, data bits (usually 7 or 8 bits), optional parity bit for error detection, and stop bits to indicate the end of a frame.

### **2.3.2.6 I2C Communication**

Inter-Integrated Circuit (I2C)[21], is a widely utilized protocol for connecting processors and microcontrollers with peripheral ICs that operate at lower speeds over short distances. The I2C interface employs a two-wire connection, consisting of the Serial Data Line (SDA) and the Serial Clock Line (SCL).

### **2.3.2.7 One-Wire Protocol**

The One-Wire protocol is a serial communication protocol developed by Dallas Semiconductor (now Maxim Integrated) that enables devices to communicate using a single data line. This protocol, widely used in applications requiring simplicity and low pin count, allows for data transfer, device enumeration, and power delivery over a single wire. One-Wire devices, such as temperature sensors, EEPROMs, and real-time clocks, are designed with minimal hardware requirements, making them cost-effective and efficient solutions.[22]

### **2.3.2.8 Heartbeat Measurement**

Heart rate measurement is critical in monitoring and assessing a person's cardiovascular health. It provides valuable insights into the general functioning of the heart and can aid in detecting anomalies or irregularities. One common technique used for

heart rate measurement is the utilization of photoplethysmography (PPG) sensors. PPG sensors pass low-intensity infrared (IR) light through biological tissues, which is then absorbed by venous and arterial blood, skin pigments, bones, and other biological tissues. PPG sensors can detect variations in blood flow as changes in light intensity because blood more strongly absorbs light than the surrounding tissues. PPG sensors capture the peaks of volumetric changes in arterial blood associated with cardiac activity and calculate the frequency of these peaks to determine the heart rate. Figure 2.1 shows the Working principle of a PPG sensors

The MAX30100 sensor is a PPG module. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. It utilizes the I2C interface for communication with micro-controllers or processors.[23]

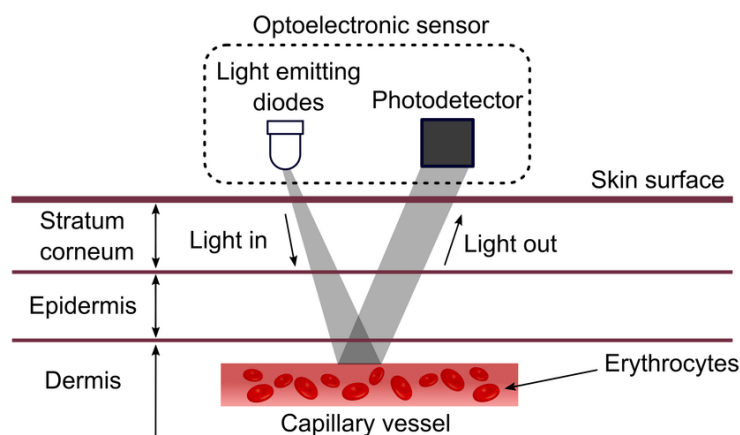


Figure 2.1: Working principle of PPG sensors.

### 2.3.2.9 Blood Pressure Measurement

A popular technique for non-invasively monitoring blood pressure is oscillometric measurement. To begin the procedure, a cuff frequently worn around the upper arm is inflated to a pressure higher than the anticipated systolic pressure. This inflation temporarily interrupts blood flow in the brachial artery. After that, the cuff gradually deflates, allowing blood flow to continue. As the cuff pressure drops, the artery momentarily opens and closes, causing the cuff pressure to fluctuate. An inbuilt pressure sensor in the cuff keeps track of these oscillations. The resulting raw pressure signal is subjected to algorithmic processing to ascertain significant properties, such as the size and timing of the oscillations. The formulas determine the mean arterial pressure (MAP), systolic pressure, and diastolic pressure. In our project, we undertook the reverse engineering of a commercial blood pressure meter as part of our research and development process.

### 2.3.2.10 Glucose Measurement

Glucose measurement involves the use of a glucose meter. A small blood sample, ordinarily obtained from a finger prick, is applied to a disposable test strip inserted into the meter. The test strip contains chemicals and electrodes that react with glucose in the blood, generating an electrical signal. The electrodes detect this signal, which is then converted into a numerical value representing the blood glucose level.

### 2.3.2.11 Body Temperature Measurement

Skin temperature measurement is the process of quantifying the temperature of the skin surface. It is a valuable physiological parameter that provides insights into the thermal regulation of the body and can serve as an indicator of various health conditions.

In our project, we employed the DS18B20[24], a digital thermometer that provides 9-bit to 12-bit Celsius temperature measurements. The sensor communicates using the one-wire protocol, making it easy to interface with microcontrollers and other devices. It has a temperature measurement range of  $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$  with a typical accuracy of  $\pm 0.5^{\circ}\text{C}$  within the range of  $-10^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ . [24]

## 2.3.3 Protocols for Data Transmission

The effective transmission of data is a fundamental part of computer networking, necessitating the use of trustworthy and strong protocols that control how data is transmitted between devices. TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two often used protocols in this area.

TCP and UDP are part of the internet protocol suite and facilitate data transmission over IP networks. While both protocols share similarities in their basic function, they possess distinct characteristics and functionalities that make them suitable for specific applications.

### 2.3.3.1 Transmission Control Protocol

Transmission Control Protocol (TCP) is a widely-used connection-oriented communication protocol, renowned for its reliability and acknowledgment capabilities. It ensures secure and ordered data transmission between devices in computer networks. With features like error checking, flow control, and congestion control, TCP provides a robust and dependable method for transmitting data.



### 2.3.3.2 User Datagram Protocol

UDP is a connectionless communication protocol, meaning it does not establish a dedicated connection before transmitting data. UDP is particularly suitable for initial handshakes and quick exchanges of data where immediate responsiveness is crucial. However, it does not provide the reliability and error correction mechanisms of TCP, making it more susceptible to packet loss or corruption. Therefore, UDP should be carefully chosen based on the specific requirements of the application.

## 2.3.4 Flutter Framework

Flutter is an open-source, event-driven UI software development kit (SDK) created by Google. It is used to develop cross-platform applications including Android, iOS, Linux, macOS, Windows, and the web from a single codebase.

Flutter apps are written in the Dart language and make use of many of the language's more advanced features. Flutter extends Dart's Pub package manager and software repository, which allows users to publish and use custom packages as well as Flutter-specific plugins. Flutter's engine, written primarily in C++, provides low-level rendering support.

### 2.3.4.1 Asynchronous Programming

Asynchronous programming is a type of programming that allows a unit of work to run independently from the primary application thread. This means that the program will run independently of this function, even if it takes a long time to complete, and notifies the main thread when the work is complete.[25]

Flutter, being a reactive framework, fully supports asynchronous programming. It provides built-in features and tools to handle asynchronous operations effectively. Flutter utilizes Dart as its programming language, which has native support for asynchronous programming through its `async/await` syntax.

### 2.3.4.2 State management

State management is a crucial aspect of building robust and efficient Flutter applications as it involves handling and updating the data that drives the user interface[26]. Flutter offers a diverse range of state management solutions, each tailored to specific needs and preferences. By employing suitable state management techniques, developers can build applications that are scalable, maintainable, and performant. Mastery of state management fundamentals is essential for creating robust and responsive Flutter applications. Some of the popular state management solutions in Flutter in-

clude BLoC, Provider, Riverpod, setState, GetIt, MobX, and States\_Rebuilder[27].

### 2.3.4.3 Business Logic Component

Business Logic Component (BLoC) is a design pattern used in Flutter applications to separate the business logic from the UI layer, making it easier to maintain and test the application[28]. BLoC follows the concept of streams and events, where the UI components emit events to the BLoC, and the BLoC processes these events, updates the state, and notifies the UI to reflect the changes.

This pattern promotes code reusability, testability, and maintainability by keeping the UI components focused on rendering the views while the BLoC handles the data flow and business logic. The BLoC pattern is suitable for larger apps with complex state management needs, but may not be necessary for smaller projects.

### 2.3.4.4 Clean Architecture with Bloc

Flutter Clean Architecture is a software architectural pattern that combines the principles of Clean Architecture with the Flutter framework to create scalable and maintainable Flutter applications.

Clean Architecture is a concept introduced by Robert C. Martin that promotes a clear separation of concerns and independence between layers of an application. It consists of three main layers:

- **Presentation Layer:** This layer contains the UI components and is responsible for handling user interactions and displaying data. This layer typically consists of widgets and screens.
- **Domain Layer:** Also known as the business logic layer, this layer contains the core business rules and logic of the application. It should be independent of any specific framework or platform and can be reused across different platforms. This layer often includes use cases, entities, and interfaces (abstract classes).
- **Data Layer:** This layer deals with data retrieval and storage. It can include repositories, data sources, and external services. The data layer is responsible for abstracting the details of data access, allowing the domain layer to remain decoupled from specific data sources.

## 2.3.5 Database Integration with Firebase

In the context of our AIoT healthcare system, the integration of a reliable and efficient data storage solution is of paramount importance.

Databases can be used to store patient information, such as their medical history, current medications, and vital signs, which can be accessed by healthcare providers remotely. It can also be used to monitor patient conditions in real-time, allowing healthcare providers to intervene quickly if necessary[29]. However, databases in an AIoT home hospitalization system must be secure and protect patient privacy, as they may contain sensitive information[30].

To address this need, we have incorporated Firebase, a cloud-based platform that provides various services for building web and mobile applications. Some of the services provided by Firebase include a real-time database, authentication, and Firestore, a NoSQL document database for mobile and web apps.

### **2.3.5.1 Firebase Authentication**

Firebase Authentication is a service that provides a comprehensive set of features and capabilities to streamline the authentication process. It supports a variety of authentication methods, including email/password, phone number, social media platforms (such as Google, Facebook, and Twitter), and more. By employing Firebase's authentication mechanisms, we can ensure that only authorized users can access and interact with the system.

### **2.3.5.2 Firebase Firestore**

Firebase Firestore is a cloud-hosted NoSQL document-based database solution for mobile and web applications. It is a well-known solution that offers a safe, personalized, and dynamic computing environment with guaranteed service quality. Firestore is organized into collections that can be linked to other subcollections. Its queries are noticeably faster and more efficient than those of Firebase Realtime Database. Firestore can be used to optimize data read cost, response size, and time regarding the cloud Firestore database

## **2.3.6 AI Models for Disease Prediction in Healthcare**

Artificial Intelligence (AI) and Machine Learning (ML) are two related fields that are rapidly evolving and have the potential to revolutionize the healthcare industry.

AI refers to the capability of machines to perform tasks that traditionally require human intelligence, such as understanding speech, making decisions, and learning from experience. AI systems can be trained to analyze data, identify patterns, and make predictions based on that data. This technology has applications in various fields, including virtual assistants and autonomous vehicles. AI is a rapidly evolving field in medicine, especially cardiology and brain science, and is revolutionizing risk prediction and stratification, diagnostics, precision medicine, workflows, and

efficiency[31].

ML is a subset of AI that involves training algorithms to make predictions or decisions based on data. ML algorithms can learn from data and improve their performance over time. There are several types of ML algorithms, including supervised learning, unsupervised learning, and reinforcement learning. In the field of healthcare, machine learning techniques have advanced. Through the analysis of medical data, the way healthcare is delivered has changed as an outcome concerning AI and machine learning[32].

Deep Learning (DL) is a subset of ML that solves problems that ML alone cannot. DL uses neural networks to boost computing labor while delivering accurate results. NLP, speech recognition, and facial recognition are just a few of the fantastic uses of DL.

AI and ML has shown potential in predicting diseases through the use of AI models. These models leverage advanced algorithms and machine learning techniques to analyze large datasets and identify patterns, risk factors, and early indicators of various diseases. Here are some specific things that AI and ML can do in healthcare:

- **Improve diagnostic accuracy in a variety of medical fields** such as radiology, pathology, and dermatology. DL algorithms have been used to analyze medical images, such as CT scans and mammograms, with a level of accuracy that is comparable to that of human radiologists. AI algorithms have also been used to analyze medical images, such as biopsy slides, with a level of accuracy that is comparable to that of human pathologists[33].
- **Predict heart disease using machine learning models.** Machine learning models can be used to target early detection and accurate prediction of heart disease, which is indispensable to bring down the mortality rates and to treat the cardiac patients with best clinical decision support[34].
- **transform the approach to medicine and improve the efficiency and effectiveness of clinical trials.** AI and big data are revolutionizing the healthcare industry, particularly in the field of clinical trials. The application of AI, specifically deep learning (DL), has shown promising results in ophthalmic imaging research. As a result, the use of AI in randomized controlled trials (RCTs) is expected to become a reality in the near future[35].

While there is growing interest and acceptance of AI models for disease prediction, there are still challenges and limitations to their application. The accuracy of these models is highly dependent on the quality and quantities of the data used for training, and there is an ongoing need to improve data accessibility and quality. Additionally, there are ethical considerations, including privacy, evolving regulatory issues, and data security concerns.

In conclusion, the use of AI models for disease prediction in healthcare offers great promise for the future. With a committed focus on the quality and accessibility of data, and an understanding of the ethical implications of AI, these models have the potential to improve the accuracy and efficiency of disease prediction and overall quality of patient care.

# Chapter 3

## Design and Implementation

### 3.1 System Architecture and Design

Our system, as shown in Figure 3.1, is designed to enable remote healthcare monitoring and facilitate efficient communication between the patient's home and the healthcare providers.

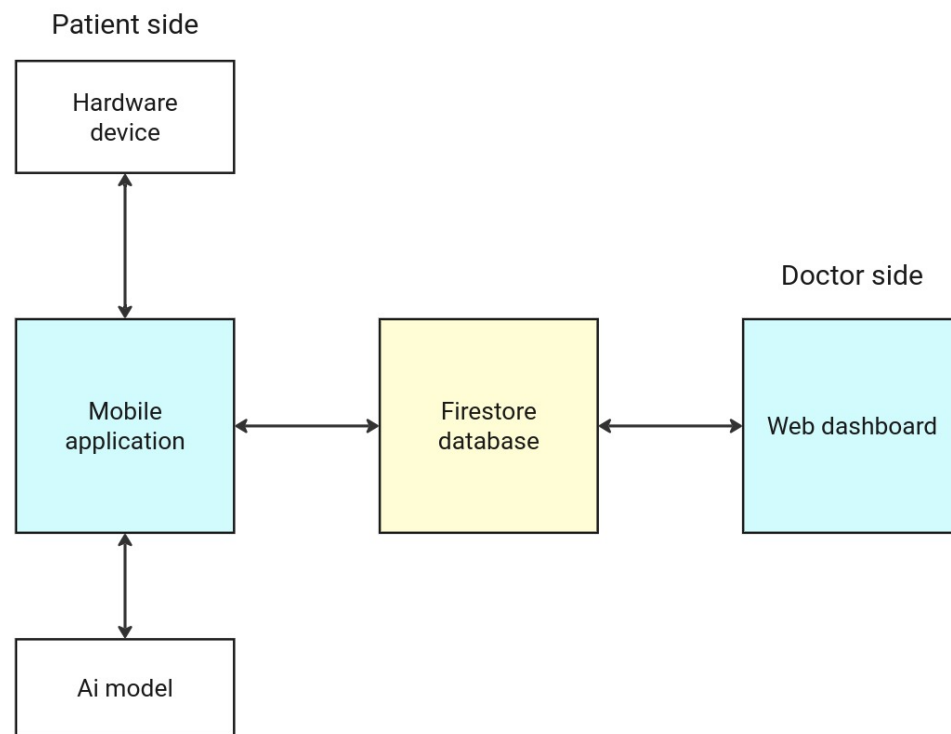
The hardware device, powered by an ESP32 microcontroller, serves as the central hub for data collection. It integrates the MAX30100 sensor for heart rate measurement and the TMP36 sensor for body temperature monitoring, as well as commercially available blood pressure and glucose sensors. These sensors enable precise and accurate measurements, providing valuable health data for monitoring and analysis.

The mobile application offers patients convenient access to their historical data and enables them to initiate measurements directly from the app. With the mobile application, patients can effortlessly review their previous measurements, track their health progress over time, and gain insights into their health trends. Additionally, the app provides a user-friendly interface that allows patients to easily start measurements, such as checking their heart rate, recording their body temperature, or monitoring their blood pressure. The application also enables direct communication between patients and doctors. Through the app's messaging feature, patients can conveniently send messages to their healthcare providers, allowing for seamless communication and timely exchange of information.

We have also designed a UDP-based custom communication protocol that facilitates seamless interaction between the ESP32 device and the mobile application. This protocol ensures efficient and reliable data exchange, allowing the ESP32 to transmit sensor readings and other relevant information to the mobile application. The protocol establishes a standardized format for data transmission, ensuring compatibility and smooth integration between the ESP32 and the mobile application.

After the data is collected, it is securely transmitted to the Firestore database. The Firestore database serves as the centralized storage system for the collected patient data, ensuring its persistence and accessibility. By leveraging Firestore, we ensure the scalability, reliability, and real-time synchronization of the data, enabling seamless retrieval and analysis by healthcare professionals. The data stored in Firestore is then processed, and utilized for generating insights, facilitating personalized healthcare management, and supporting doctors' decision-making.

Additionally, we have designed a web dashboard specifically tailored for doctors, providing convenient access to their patient's data and the insights generated by the AI model. Each doctor is granted secure access to their respective patients' data, allowing them to view and analyze the collected measurements, historical trends, and AI-generated insights.



miro

Figure 3.1: System Architecture Diagram

### 3.1.1 Communication Setup between Hardware and Mobile Application

Establishing a robust and efficient communication protocol is of paramount importance in our AIoT-based home hospitalization system. Recognizing this, we developed a custom communication protocol to facilitate data exchange between the ESP32 microcontroller and the mobile application. This protocol was specifically designed to ensure reliable and efficient transmission of patient data. By implementing this protocol, we achieved a lightweight and efficient communication mechanism, optimized for the limited resources of the ESP32 and the mobile application.

The protocol operates over UDP and utilizes a fixed port number, 2409, for communication. The protocol packet begins with a unique protocol ID, 'E-Health\0', which serves as an identifier for our system. Following the protocol ID, two bytes are allocated for the opcode, indicating the type of message being transmitted. The remaining portion of the packet can accommodate up to 244 bytes of data.

We have designed our communication protocol with a focus on simplicity and scalability. We have ensured that the protocol is easy to develop and extend by adopting a modular approach. This allows for the seamless addition of new opcodes to accommodate future functionalities or system requirements. The protocol's flexible structure enables straightforward opcode definition and integration, facilitating the incorporation of new message types without compromising the integrity of the existing communication framework. By making our protocol adaptable and easy to expand, we have future-proofed our system, providing the foundation for potential enhancements and advancements in the future.

#### 3.1.1.1 Protocol OpCodes

In order to enhance the scalability of our protocol, we have implemented opcode categorization by assigning specific opcode ranges to different types of commands. This approach allows for better organization and management of the protocol functionalities. We have defined three main categories for the opcodes:

- 0x1XXX for system commands
- 0x2XXX for handshake requests and replies
- 0x3XXX for measurements and sensors, this range is divided into 4 sub-ranges:
  - 0x31XX: Blood pressure
  - 0x32XX: Blood glucose level
  - 0x33XX: Body temperature
  - 0x34XX: Heartbeat



Table 3.1 provides an overview of the currently supported opcodes in our protocol along with a brief definition of each opcode. The table serves as a reference for understanding the purpose and functionality of each opcode within the communication framework of our AIoT-based home hospitalization system.

Opcode	Brief definition
0x1001	Network configuration (SSID and password)
0x1999	Reset save network configuration
0x2000	Send handshake request
0x2001	Handshake reply
0x3X00	Send measurement request
0x3X01	Update in the measurement state
0x3X02	Measurement result packet
0x3X02	Measurement failed

Table 3.1: Supported OpCodes

### 3.1.1.2 Protocol Workflow

In the protocol workflow, the mobile application initiates communication by broadcasting a handshake request. The hardware device responds with a handshake reply, establishing a connection and determining the IP address of the Device Interface Server. This enables secure bidirectional data transmission, allowing the mobile application to send commands and the hardware device to transmit real-time health measurements. The protocol ensures efficient and reliable communication, facilitating remote monitoring and timely healthcare interventions.

After the establishment of the connection, the device interface server actively listens to packets sent by the mobile application.

Figure 3.2 provides an example workflow for handshake process.

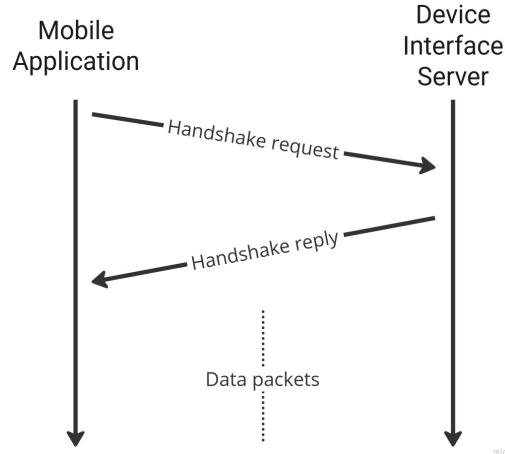


Figure 3.2: Protocol Handshake Workflow

### 3.1.2 Hardware Components and Functionalities

The hardware components of our AIoT-based home hospitalization system play a critical role in capturing and processing patient data for remote monitoring. At the core of our system is the ESP32 microcontroller, which serves as the central processing unit. With its dual-core architecture and built-in Wi-Fi and Bluetooth capabilities, the ESP32 provides the necessary computational power and connectivity for data transmission.

The MAX30100 heartbeat sensor, responsible for heart rate monitoring, is connected to the ESP32 microcontroller through the I2C interface, specifically using I2C port 0. This allows for reliable and high-speed data exchange between the sensor and the microcontroller.

The DS18B20 sensor, integrated for temperature measurement, is connected to the ESP32's GPIO pin. This sensor provides temperature readings directly in digital format, utilizing a single wire for communication and power. The ESP32 interfaces with the DS18B20 sensor using the one-wire protocol, enabling accurate temperature measurements for monitoring the patient's body temperature.

Additionally, we have integrated a commercially available blood pressure sensor into our system, specifically connecting it to UART 1 of the ESP32 microcontroller.

In the case of glucose measurement in our system, we encountered a challenge as we couldn't find a suitable prototyping sensor. As a solution, we designed the system to allow the user to perform the glucose measurement using an external glucose monitoring device. The user then manually inputs the glucose measurement results into the mobile application.

In addition to the essential hardware components and sensors, we have included additional features to enhance the functionality and usability of our system. These include a reset button and power LED, which provide convenient control and indication for powering on or resetting the hardware device. Furthermore, we have incorporated an additional LED that serves as an indicator for various system states or notifications, providing visual cues to the user. These additional components contribute to a user-friendly experience and facilitate easy interaction with the home hospitalization system.

Figure 3.3 provides a detailed diagram illustrating the interconnections and arrangements of the hardware components in our AIoT-based home hospitalization system, showcasing the ESP32 microcontroller, MAX30100 heartbeat sensor, TMP36 temperature sensor, blood pressure sensor, power and indicator LEDs, and reset button. Additionally, Figure 3.4 demonstrates the physical implementation of our circuit.

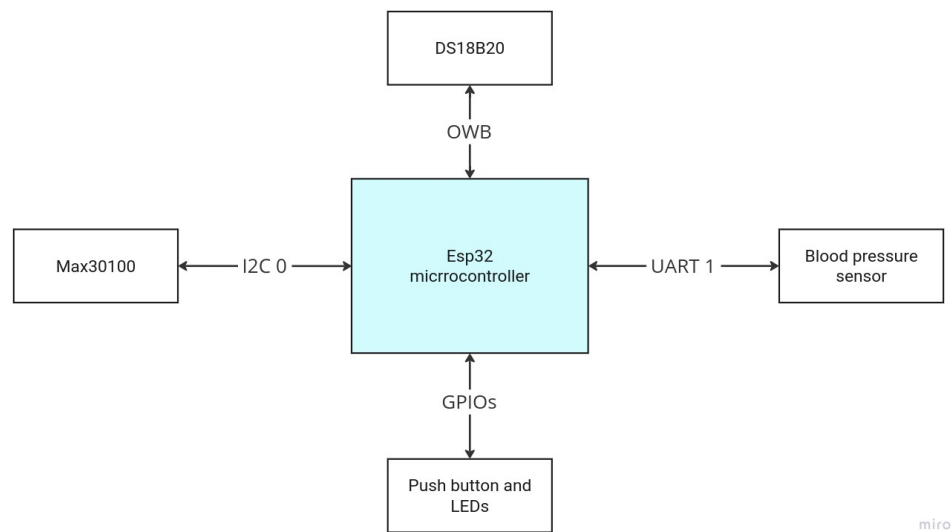
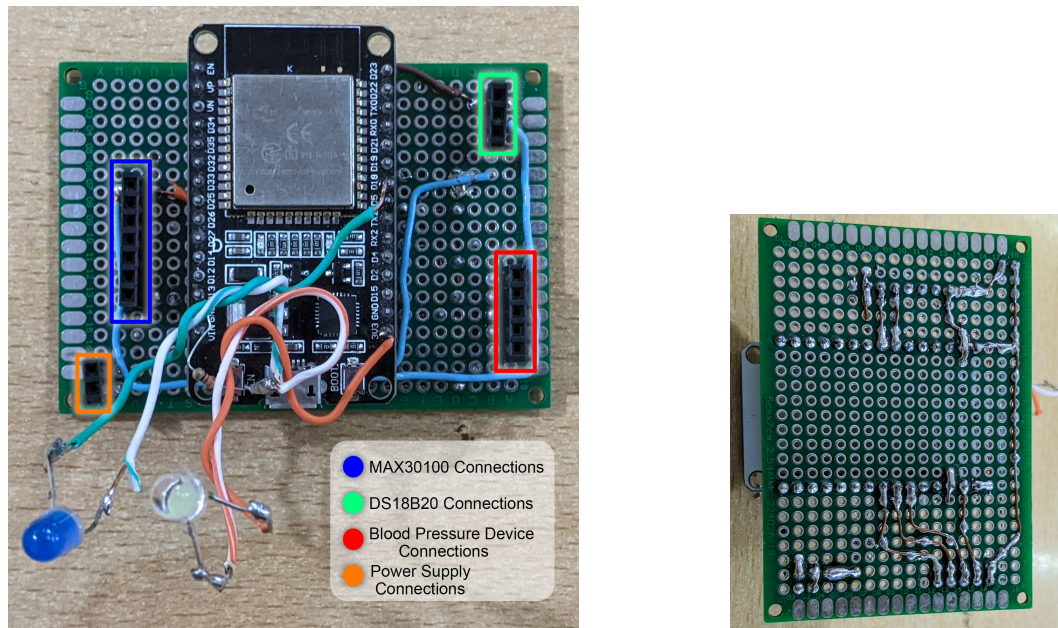


Figure 3.3: Hardware Components Diagram



(a) Top View of the Physical Implementation of the Circuit, showcasing the Pinout.

(b) Back View of the Physical Implementation of the Circuit.

Figure 3.4: Physical Implementation of the Circuit

### 3.1.2.1 Reverse Engineering Blood Pressure

Prototyping healthcare sensors can be a costly and time-consuming process, often resulting in less precise measurements. Recognizing these challenges, we made a strategic decision to leverage commercially available sensors for our AIoT-based home hospitalization system. This approach ensures the reliability and accuracy of the measurements while saving time and resources. Additionally, The accessibility of these sensors in the market makes procurement and replacement easier when needed.

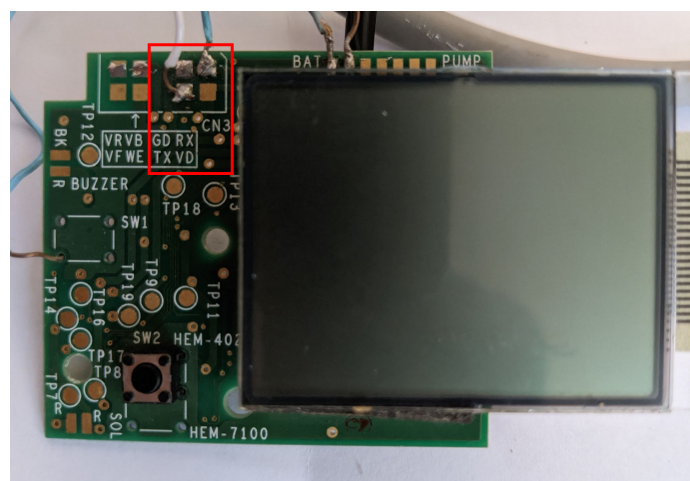


Figure 3.5: Blood Pressure Sensor PCB

During the inspection of the blood pressure sensor's PCB, we discovered familiar UART interface traces labeled RX, TX, and GND as shown in figure 3.5. By analyzing the data communication using a logic analyzer, we determined the UART interface configuration, which included a baud rate of 9600, 1 start bit, no stop bit, and no parity.

The sensor provided updates during the measurement process, including initialization readiness ("CHK"), waiting for input ("WAI"), inflation ("INF"), and deflation ("DEF"). After the measurement, the sensor transmitted "EXH" followed by 13 bytes of data. We found that the first three bytes contained crucial information, indicating measurement success or error. An error is indicated by a result that starts with "f15" and a successful measurement result starts with "f00" and contains encoded data in subsequent, allowing us to decode and extract systolic and diastolic blood pressure values and heart rate.

A successful data reading example was represented by the format "f00420026804E," corresponding to a systolic blood pressure of 132 mmHg, diastolic blood pressure of 77 mmHg, and a heart rate of 78 beats per minute. In this case, the systolic blood pressure was encoded as "4200," which was obtained by converting it to decimal. By multiplying this value by 2, we derived the systolic blood pressure of 132 mmHg. Similarly, the diastolic blood pressure was encoded as "2680." The presence of the digit "8" in the encoding indicated that the diastolic blood pressure is an odd number, so we added 1 to it, resulting in a diastolic blood pressure of 77 mmHg. The heart rate value was directly converted to decimal without further manipulation.

### 3.1.3 Firmware Design and Implementation

In the firmware development, we utilized the FreeRTOS (Real-Time Operating System) framework in conjunction with the ESP-IDF, which allowed us to leverage the powerful features and capabilities of both frameworks. FreeRTOS provided real-time multitasking capabilities and scheduling functionalities, while ESP-IDF offered hardware abstraction, networking support, and system-level features tailored for the ESP32. This combination allowed us to develop robust and efficient firmware that effectively utilized the resources of the ESP32, ensured reliable data processing, and enabled communication with the hardware components and external systems.

As shown in Figure 3.6, first, the firmware checks if there are valid network credentials stored in the Non-Volatile Storage (NVS). If valid credentials are found, the firmware enters the NORMAL\_MODE. the ESP32 initializes the Wi-Fi module as a station (STA) and attempts to connect to the network using the retrieved credentials. However, if no valid network credentials are available, the firmware enters the FIRST\_BOOT state. In FIRST\_BOOT, the ESP32 operates as an access point (AP), allowing clients to connect and configure the network settings.

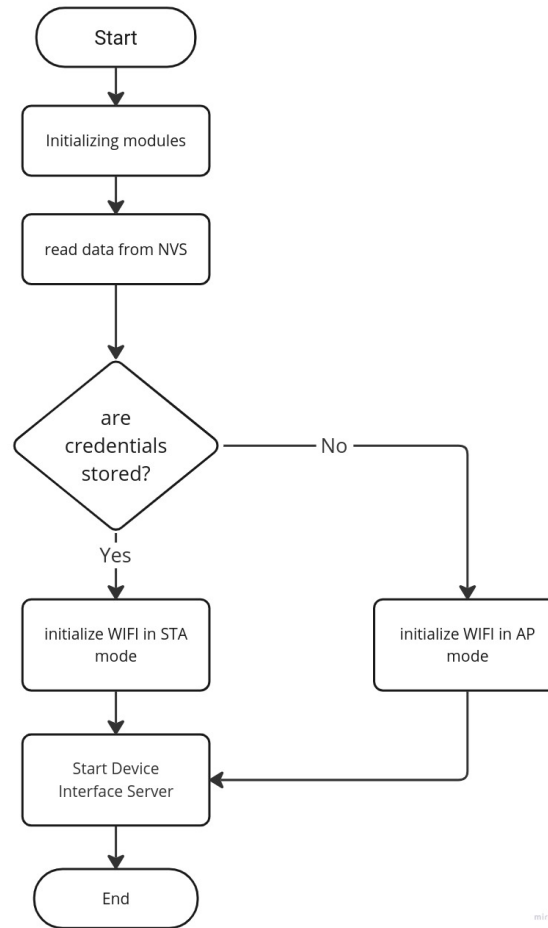


Figure 3.6: System Initialization Flowchart

### 3.1.3.1 Device Interface Server

After booting and initializing the Wi-Fi interface in the correct mode, the firmware enters the device interface server module. This module operates as a closed-loop function, continuously listening for incoming packets. When a packet is received, the firmware first checks the protocol ID to ensure it matches the expected value. If the protocol ID is valid, the firmware proceeds to examine the opcode within the packet. Based on the opcode, the firmware executes the corresponding action or process associated with that opcode. This allows for seamless handling and interpretation of incoming packets, ensuring efficient communication and appropriate responses within the system.

If a measurement opcode is received, the firmware creates a separate task specifically dedicated to the measurement function. As part of the initialization process, two queues are established. The first queue allows for the transmission of the received command from the mobile phone to the measurement task, ensuring that the task is aware of the desired measurement operation. The second queue facilitates the communication of updates about the measurement state from the task back to the mobile

phone. This bidirectional communication enables the patient to actively track and monitor the progress and status of their measurement in real-time, providing valuable feedback and enhancing the overall user experience.

### **3.1.3.2 Network Configuration Setup Process**

When the device initially boots up, the user has to connect to the access point of the ESP32 microcontroller. The user can then send the network SSID and password from the mobile app to configure the device's network settings.

When the network configuration opcode is received, the hardware device switches from the access point mode to the station mode and begins the process of connecting to the specified Wi-Fi network. If the connection to the network fails, the firmware reverts back to AP mode, allowing the user to retry the network configuration process.

However, if the connection is successful, the firmware stores the received network information in the NVS of the ESP32 and operates in station mode. In the next reboots, the system reads the network configuration from the NVS, allowing the device to directly enter station mode and connect to the configured Wi-Fi network without requiring user intervention.

### **3.1.3.3 Measurement Process and Task Execution**

Each measurement task within the firmware follows a similar structure. It consists of a loop that continues until the measurement process is completed. Within this loop, a Finite State Machine (FSM) is implemented to manage the different states and transitions of the measurement process. Whenever a state change occurs, the updated state information is sent to the device interface server via a queue. Subsequently, the device interface server relays this information to the mobile app, ensuring that the patient is kept informed about the current state of their measurement. This continuous feedback loop enables real-time tracking and communication of the measurement progress, enhancing the patient's engagement and understanding of the ongoing process.

Figure 3.7 presents a flowchart illustrating the process by which the device interface server handles measurement opcodes.

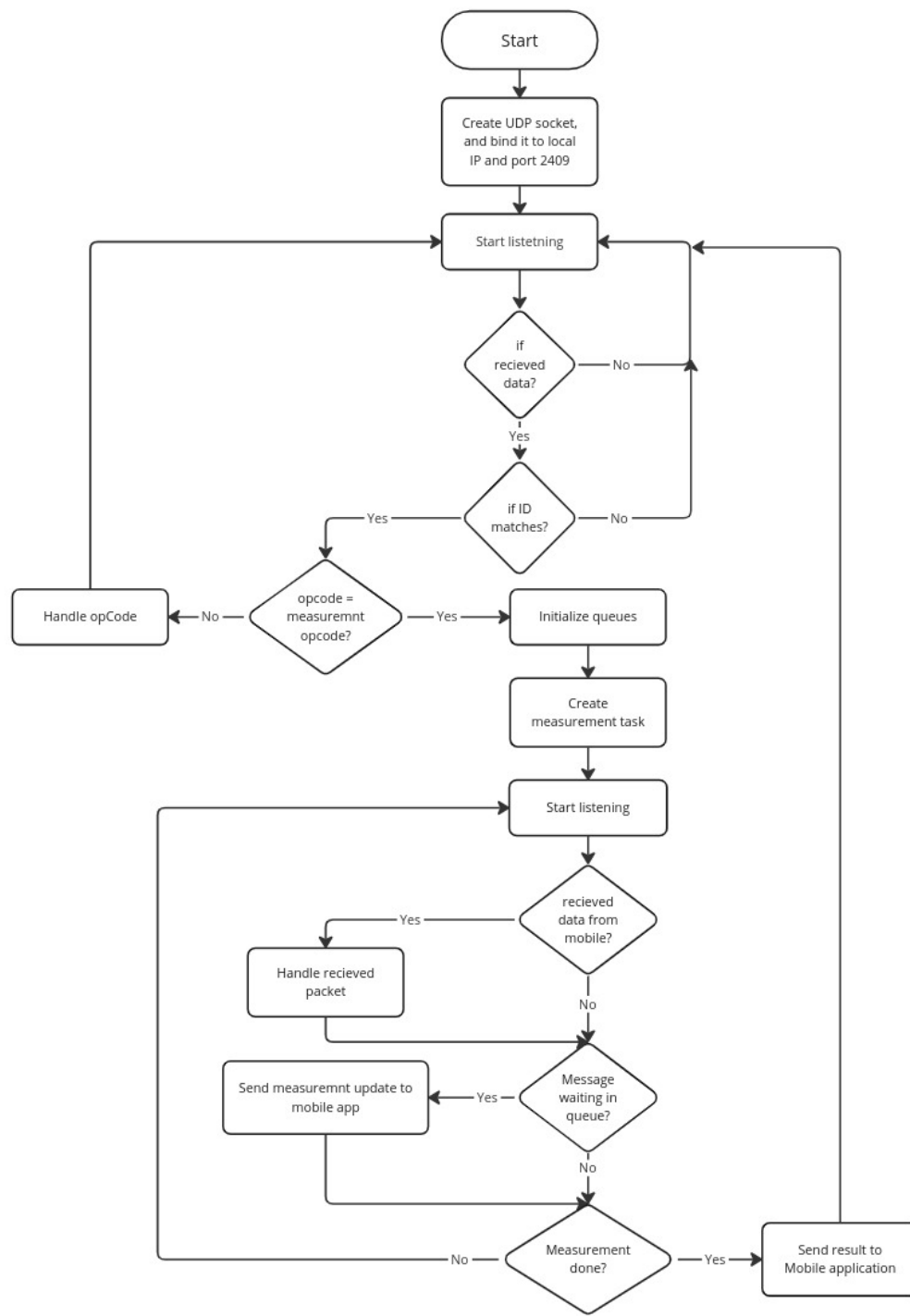


Figure 3.7: Flowchart of the Measurement Opcode Handling Process in the Device Interface Server.



#### 3.1.3.4 Heart Rate Data Acquisition

To ensure optimal performance and accurate measurements, we carefully configure the sensor according to our requirements. The configuration includes:

- Sampling rate: 1000Hz
- LED pulse width: 1600us
- ADC range: 16bit
- LED current level: 11mA

The parameters of the MAX30100 sensor play a crucial role in the accuracy and reliability of the measurements as well as the comfort of the patient. The sampling rate determines how frequently the sensor captures data points, affecting the temporal resolution of the measurements. Higher sampling rates can provide more detailed information but may consume more power. LED brightness affects the intensity of the emitted light, which can influence the sensor's ability to capture accurate signals. Pulse width determines the duration of the LED pulse, impacting the depth of tissue penetration.

During the heart rate data acquisition process, we begin by waiting for the patient to place their finger on the sensor. Once a heart pulse is detected, we initiate the recording process. To ensure accurate and reliable data, we employ a rolling average with an outlier rejection filter with a window size of 5 on data recorded over 15 seconds. This filter helps us handle any outliers or irregularities in the collected data, ensuring that we obtain accurate and reliable heart rate measurements.

#### 3.1.3.5 Body Temperature Data Acquisition

For body temperature measurement in our system, we have employed the oral temperature method. This approach involves utilizing a sensor that is placed in the mouth to capture temperature readings. The process begins by allowing the sensor to calibrate to the mouth temperature for approximately 10 seconds, ensuring accurate measurements. Following the calibration, we capture the maximum temperature reading within the next 5 seconds, providing a reliable indication of the oral body temperature.

### 3.1.4 Mobile Application Design and Features

The mobile application, built using the Flutter framework, serves as a fundamental element of the system. It enables users to monitor and manage their health conditions. This chapter provides an in-depth explanation of the mobile application's design and key features, with a particular emphasis on various elements such as authentication, connection setup, sensor control, data visualization, data storage, history viewing, appointment requests, chat functionality, and notifications.

#### 3.1.4.1 Authentication and User Management

To provide secure access, the mobile application includes powerful authentication and user management features. Firebase Authentication is used to handle user authentication after registration where the user is required to register and create an account using their email address, and enter their personal information, including their name, height, weight, sex, blood pressure, phone number, and their corresponding doctor information. This last information allows for better communication between the user and their healthcare provider. The personal information allow the mobile application to calculate relevant health metrics, such as BMI (Body Mass Index), and offer appropriate health objectives and guidelines.

The user profile management feature allows users to update and modify their personal information as needed, ensuring that the mobile application keeps up-to-date records of the user's health-related details. Additionally, users have the option to configure their privacy settings, specifying the level of data sharing and visibility to healthcare providers or other authorized individuals.

The inclusion of sophisticated authentication and user management functions in the mobile application enables safe access and individualized user experiences, which are critical for maintaining user confidence and engagement.

The FigureB.1 in the Appendix section Shows a screenshot for this feature.

#### 3.1.4.2 Connection Setup with Hardware

To establish a connection with the hardware device, the mobile application initiates the handshake by broadcasting a handshake request to the network. Upon receiving a handshake request, the device interface server responds by sending a handshake reply to the IP address of the requesting device. This handshake is necessary to retrieve the IP address of the device interface server and establish a connection. Additionally, it serves as a means to send the initialization state of the sensors, including information about whether each sensor has successfully initialized or encountered any failures during the initialization process. Once the handshake is completed, the IP address is stored for future reference, eliminating the need for repeated handshakes in subsequent sessions.

To ensure seamless tracking of received packets from the hardware device without impacting the user interface rendering, we have implemented the Flutter BLoC architecture. which separates the business logic and the user interface, providing an efficient and organized approach to handling data flow.

When a measurement event is received by the BLoC, it triggers a handler function that contains a loop. This loop continuously listens to the hardware device for updates and emits corresponding states based on the received data. The BLoC effectively manages the state of the application, allowing for real-time updates and rendering of relevant information to the user interface.

The FigureB.2 in the Appendix section Shows a screenshot for this feature.

By utilizing the Flutter BLoC pattern, we ensure efficient communication and synchronization between the mobile application and the hardware device. This architecture enables seamless tracking of measurements and provides a responsive user experience, as the UI remains unaffected by the continuous data processing happening in the background. To provide a visual representation of the interaction between the user interface, the BLoC, and the hardware, Figure 3.8 illustrates the flow of events and asynchronous communication within the system.

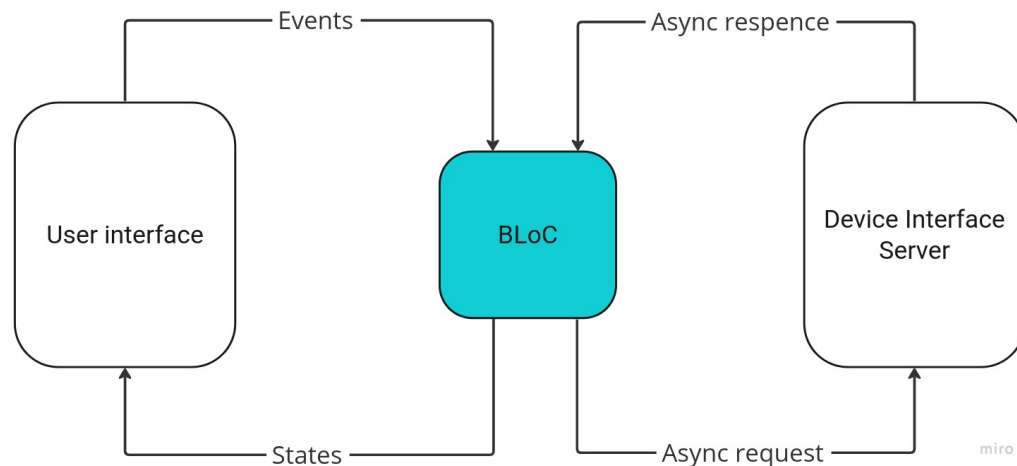


Figure 3.8: Diagram Illustrating the Interaction between UI, BLoC, and Hardware.

### 3.1.4.3 Sensor Control and Data Visualization

The mobile application provides users with comprehensive control over the sensors integrated into the hardware devices. It interacts with the sensors through the established connection, enabling users to initiate sensor measurements.

The mobile application provides users with a dedicated screen for sensor control and data visualization. This screen allows users to select the desired sensor they want to utilize and provides control options such as turning the sensor on or off.

For instance, if the user wants to utilize a heart rate sensor, they can choose it from the available options on the screen. Once selected, the application provides interactive hints and instructions to guide the user through the sensor usage process. These hints may include prompts such as "Please place your finger on the sensor" or "The sensor is currently measuring your heart rate."

Once the sensor has captured the necessary readings, the application displays the collected data on the screen in a clear and understandable format for the user.

The FigureB.4 in the Appendix section Shows a screenshot for this feature.

#### **3.1.4.4 Data Storage with Firebase Firestore**

Data storage in the mobile application relies on the Firebase Firestore database, which provides a structured and scalable solution. The process begins with user registration, where Firebase creates an account and assigns a unique user ID. This user ID serves as the basis for creating a collection within the Firestore database, specifically under the "Patients" document. In this collection, the user's personal information, including their entered details during registration, password, and email, are stored.

Additionally, another collection is created using the same user ID, but this time under the "Measurement" document. Within this collection, four empty arrays are generated: glucose, temperature, blood pressure, and heart rate. These arrays serve as placeholders to store the user's vital sign measurements.

In scenarios where the mobile application operates in offline mode, such as when the user measures their vital signs without an internet connection, the data is temporarily stored locally. Firebase provides seamless offline support through its package, automatically pushing the locally stored data to the database once a connection is reestablished. This ensures that vital sign measurements are reliably synchronized and stored in the Firestore database for future access and analysis.

Furthermore, authorized doctors have access to the database, allowing them to securely analyze user data. This functionality empowers doctors to review the data, perform analyses, and provide personalized recommendations based on the user's health information.

#### **3.1.4.5 History Viewing and Graphical Representations**

The mobile application provides users with a history viewing feature, enabling them to access and review their previous sensor readings and health data. By retrieving historical data from the Firestore database, users can conveniently access and analyze their health information in chronological order.

To enhance data analysis and interpretation, the application offers customizable graphs and tables. Users have the flexibility to select specific time intervals, health

parameters, and visualization formats, allowing them to generate graphical representations of their historical health data.

The FigureB.3 in the Appendix section Shows a screenshot for this feature.

#### **3.1.4.6 Appointment Requests and Chat Functionality**

The mobile application enables users to request and manage appointments with the doctor. Users can schedule appointments, view upcoming appointments, and receive notifications regarding appointment confirmations and reminders. Additionally, the mobile application facilitates communication between users and healthcare providers through a chat functionality. Users can engage in real-time messaging, seek clarifications, and receive guidance from healthcare providers conveniently within the mobile application.

The chat functionality enhances the overall user experience and promotes effective communication and collaboration in the home healthcare setting.

### **3.1.5 Web Development for Doctor's Dashboard**

The web development aspect of the AIoT-based home healthcare system includes the creation of a comprehensive doctor's dashboard. This section focuses on the key features and functionalities implemented in the web dashboard, providing doctors with efficient management and analysis tools.

One notable advantage of the web development process is the utilization of the Flutter framework, which allows for code sharing between different platforms. By leveraging Flutter's cross-platform capabilities, we can maximize code reusability and reduce development efforts. In this case, we were able to use the same codebase as the mobile application to build the web dashboard. This approach not only ensures consistency in functionality and user experience but also streamlines the development process by eliminating the need to write separate code for the web platform.

#### **3.1.5.1 Authentication and User Management**

Similar to the mobile application, the web dashboard incorporates a robust authentication and user management system.

During registration, doctors create an account using their email address and provide their personal information including their name, hospital name and address, specialty, mobile number, and gender. This ensures secure identification within the system. Upon successful registration, doctors are assigned a unique identifier generated by Firebase Authentication. This identifier is used to create a collection in the Firestore database, storing the doctor's personal information. In this collection an

"IDs" list is created to store patient IDs associated with the doctor. This allows the doctor to access patient information, data measurements, and history.

By utilizing patient IDs, doctors can efficiently retrieve and analyze patient data, gaining insights into health conditions and providing personalized recommendations. This structured organization of data ensures secure and scalable access to patient information.

The FigureC.1 in the Appendix section Shows a screenshot for this feature.

### **3.1.5.2 Patient Management and Data Visualization**

The web dashboard provides doctors with a comprehensive patient management interface. Doctors can view a list of their registered patients and access individual patient profiles. Within the patient profiles, doctors can review personal data, medical history, and vital sign measurements.

The measurements are presented in various visual formats, such as charts and tables, enabling doctors to analyze trends and patterns in the patient's health data.

The FiguresC.3, C.4 and C.5 in the Appendix section Shows a screenshots for this features.

### **3.1.5.3 Appointment Requests and Chat Functionality**

The web dashboard facilitates appointment management between doctors and patients. Doctors can request appointments with patients when they identify potential issues in the patient's data.

Additionally, doctors can engage in real-time communication with patients through a chat functionality. This enables doctors to seek clarifications, provide guidance, and establish effective communication channels with patients.

### **3.1.5.4 Notifications and Alert System**

The web dashboard incorporates a notification system to keep doctors informed about important events and updates. Doctors receive notifications when patients send messages, new patients are added to their supervision, or when appointment requests are made by patients. This ensures timely communication and allows doctors to respond promptly to patient needs.

An alert system is also integrated into the web dashboard. When the AI model detects anomalies or potential health issues in the patient's data, doctors receive immediate alerts. This helps doctors identify critical situations and take appropriate actions to ensure patient well-being.

## 3.2 AI Model Development for Disease Prediction

In our system, we have developed an AI model specifically for diabetes prediction. This section focuses on the development process and key features of the AI model.

The objective of the AI model is to accurately predict the likelihood of an individual developing diabetes based on various input parameters such as age, gender, body mass index (BMI), blood pressure, and glucose levels. The model utilizes machine learning algorithms to analyze and identify patterns within the data, enabling it to make accurate predictions.

### 3.2.1 Dataset Selection and Preprocessing

For the development of the AI model for diabetes prediction, we utilized the Pima Indians Diabetes Dataset. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset contains valuable information about a group of Pima Indian people, including their medical attributes and whether they have been diagnosed with diabetes or not. Here are the details of the Pima Indians dataset:

- Number of Instances: 768
- Number of Attributes: 8 plus the class variable

Each instance in the dataset represents a unique individual and is associated with the following attributes:

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-hour serum insulin ( $\mu$ U/ml)
- BMI: Body mass index ( $\text{weight in kg}/(\text{height in m})^2$ )
- Diabetes Pedigree Function: Diabetes pedigree function
- Age: Age in years

Additionally, the dataset includes a class variable:

- Outcome: This binary variable indicates whether an individual has been diagnosed with diabetes or not. A value of 0 represents no diabetes, while a value of 1 indicates the presence of diabetes.

The Pima Indians dataset provides a diverse set of attributes that can potentially contribute to diabetes prediction. For our model development, we excluded three attributes (Diabetes Pedigree Function, Insulin, and SkinThickness) due to the unavailability of data. These attributes were omitted from the input features during training and prediction phases.

## 3.2.2 AI Model Architecture

The architecture of our AI model for diabetes prediction is designed to effectively process the input features and make accurate predictions based on the available data. This section provides an overview of the architectural components and their functionalities.

### 3.2.2.1 Transformer-Encoder Layer

The core component of our model is the Transformer-Encoder layer. This layer is responsible for processing the input features, capturing the underlying patterns and relationships in the data, and inferring relationships between different data attributes during the forward pass. The Transformer-Encoder architecture has been widely used in natural language processing tasks, but it can also be adapted for numerical data processing[36].

The Transformer-Encoder layer consists of multiple self-attention heads and feed-forward neural networks. Each self-attention head attends to different parts of the input features, enabling the model to capture both local and global dependencies. This allows the model to extract meaningful representations from the input data[36].

Additionally, the Transformer-Encoder layer uses layer normalization, which is a technique for normalizing the activities of the neurons in a layer. Layer normalization significantly reduces the training time in feed-forward neural networks and is very effective at stabilizing the hidden state dynamics in recurrent networks[36].

In our model, we used a Transformer-Encoder layer with a model size of 20 and a single head. The smaller model size helps to reduce the computational complexity while still maintaining sufficient capacity to capture relevant information from the input features.

Figure 3.9 shows the Transformer-Encoder Layer.



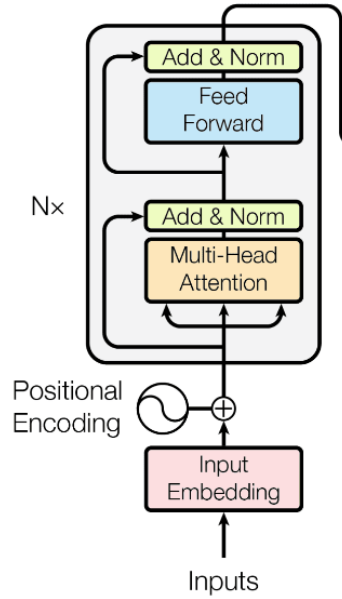


Figure 3.9: Transformer-Encoder Layer

### 3.2.2.2 Dense Layers

Following the Transformer-Encoder layer, we incorporated two dense layers. These layers serve to further process the output of the Transformer-Encoder and reduce it to a single float item, which represents the predicted outcome.

The inclusion of dense layers serves as a means of controlling the dimensionality of the system. It allows us to provide the Transformer-Encoder with a sufficiently long sequence and then reduce the output back to a classification unit specific to our use case, which involves binary classification. To prevent overfitting to the small dataset, dropout regularization is applied after each layer.

The GeLU (Gaussian Error Linear Unit)[37] activation function is applied at the end of the Transformer-Encoder, while ReLU (Rectified Linear Unit)[38] activation is applied after each dense layer. These activation functions introduce non-linearities into the model, enabling it to capture intricate relationships between the input features and the target variable. This capability enhances the model's predictive abilities and improves its accuracy in making diabetes predictions.

The first dense layer expands the input into the Transformer-Encoder layer model size of 20, enabling the model to learn more complex representations. The second dense layer then reduces the output of the Transformer-Encoder dimension to a single float item, which represents the predicted outcome of the diabetes diagnosis. The raw output float is then normalized into a float ranging from 0 to 1 using a Sigmoid function.

Figure 3.10 shows the two dense Layers.

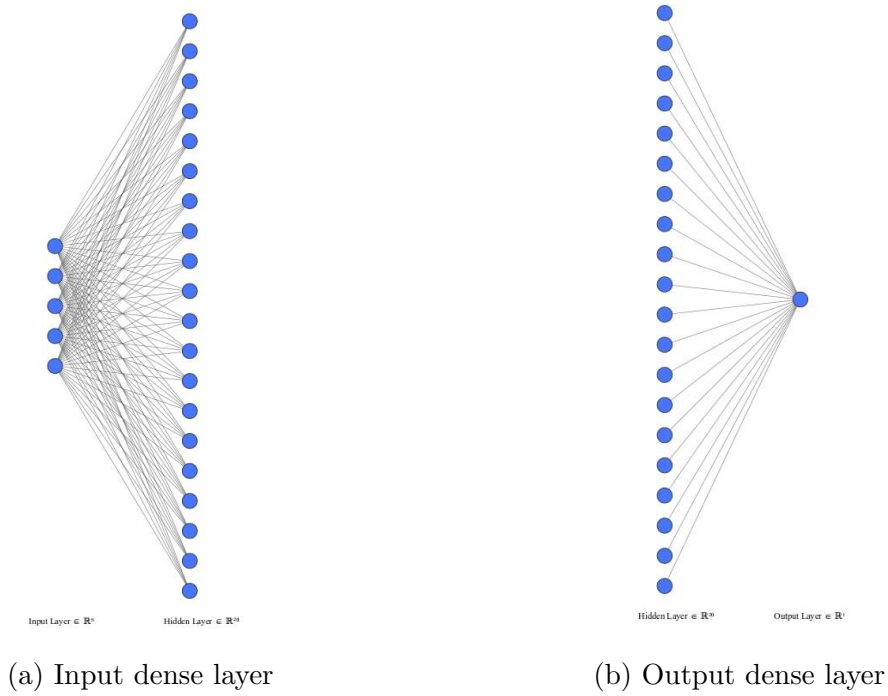


Figure 3.10: Illustration of the Dense Layers

### 3.2.2.3 Benefits of the Architecture

The chosen architecture offers several benefits for our diabetes prediction model:

- **Capturing Complex Patterns:** The Transformer-Encoder layer effectively captures complex patterns and dependencies in the input features. Its self-attention mechanism allows the model to focus on relevant information and learn meaningful representations.
- **Non-linearity and Feature Extraction:** The dense layers introduce non-linearities and enable the model to extract higher-level features from the encoded representations. This helps the model learn more abstract representations and improves its ability to make accurate predictions.
- **Scalability and Efficiency:** The selected architecture strikes a balance between model complexity and computational efficiency. The smaller model size reduces computational requirements while still maintaining sufficient capacity for effective prediction.

Overall, the architecture of our AI model for diabetes prediction combines the power of the Transformer-Encoder layer and dense layers to capture complex patterns and make accurate predictions. The combination of these components enhances the model's ability to analyze the input features and provide valuable insights for diabetes diagnosis.

### 3.2.3 AI Model Training and Validation

In the development of our AI model for diabetes prediction, a crucial step involves training and validation. This section outlines the process and methodologies employed during the training phase to optimize the model's performance and ensure its ability to generalize well on unseen data.

#### 3.2.3.1 Data Split

To facilitate training and validation, we initially divided the Pima Indians dataset into two portions: a training set and a validation set. The training set comprised 80% of the dataset, and the remaining 20% constituted the validation set. The small dataset size, doesn't allow for larger splits, so we were very careful with regularization and optimizer choice.

#### 3.2.3.2 Training Process

During the training phase, we employed specific techniques to iteratively update the model's parameters and optimize its predictive capabilities. The following steps were taken:

- **Epochs:** We conducted training over 5 epochs. An epoch represents a complete pass through the training data, allowing the model to learn from the dataset multiple times. Training over multiple epochs enables the model to refine its internal representations and capture complex patterns present in the data.
- **Batch Size:** A batch size of 1 was utilized, meaning the model updates its parameters after processing each individual instance in the training set. This approach ensures that each data point contributes to the training process and allows for efficient parameter updates.

#### 3.2.3.3 Learning Rates

The learning rate is a crucial hyperparameter that determines the step size during parameter updates. Optimal learning rates are essential for effective convergence and model performance. In our AI model, we used different learning rates for different components:

- **Transformer-Encoder Layer:** A learning rate of  $3.5e-5$  was applied for the Transformer-Encoder layer. Additionally, a warmup phase was incorporated for the learning rate schedule. The warmup phase gradually increases the learning rate during the initial training steps, allowing the model to stabilize and converge more effectively.
- **Dense Layers:** The Dense layers used a learning rate of 0.001, also with a warmup stage. This learning rate was selected to facilitate efficient parameter updates in the Dense layers and ensure convergence.

### 3.2.3.4 Optimizers

The different layers require different optimizers with different optimization processes. We employed specific optimizers for each component:

- **Transformer-Encoder Layer:** is a post-LN design, which is proven to be unstable during training. To solve this problem, a warmup stage is applied during training, the learning rate is increased linearly as the model becomes more of extracting the necessary features and their inferred relationships (fits the data more). We utilized the AdamW optimizer, which combines the advantages of the Adam optimizer and weight decay techniques. AdamW provides more stable and reliable optimization, ensuring that the Transformer-Encoder layer learns the underlying patterns and relationships in the data effectively.
- **Dense Layers:** The Dense layers were optimized using the Adam optimizer, a widely used and efficient optimization algorithm for deep neural networks. The Adam optimizer adapts the learning rate for each parameter individually, enabling effective convergence and training of the Dense layers.

### 3.2.3.5 Validation

After each epoch during training, we evaluated the model's performance on the validation set to assess its generalization capabilities. By using a separate dataset, we could measure the model's ability to make accurate predictions on unseen data and detect any potential overfitting.

The training and validation phase of our AI model development played a crucial role in optimizing the model's performance and ensuring its ability to generalize well. Through careful selection of hyperparameters, including epochs, batch size, learning rates, and optimizers, we facilitated effective training and convergence. The validation process allowed us to assess the model's predictive capabilities on unseen data and make any necessary adjustments to enhance its performance.

## 3.2.4 Integration with the System

To deploy the AI model, we set up a server that would host the model and handle incoming requests. This was done with Google Cloud Platform (GCP). For the backend itself, we used FastAPI, a high-performance backend framework for quickly building RESTful APIs with Python. Since the model implementation was also in Python, it was pretty straightforward to integrate with the backend.

The entirety of the backend is a single REST route that serves as the interface for the AI model. This route is responsible for receiving input parameters in form of JSON (JavaScript Object Notation), parsing them and starting a forward pass of the AI model with the input parameters. In our case, when a patient takes a

measurement, our mobile application initiate a POST HTTP request with a JSON body that contains blood pressure, glucose, BMI, age and number of pregnancies of any patient. The AI model processes it and generates a prediction. The prediction is then relayed back to the mobile app, which stores it in the database for future reference. Additionally, the app notifies the attending doctor of the new prediction, ensuring timely access to the patient's health status.

# Chapter 4

## Results and Discussion

### 4.1 Overview

To evaluate the home hospitalization system proposed in this report, We conducted a series of rigorous tests to evaluate the performance and functionality of our AIoT-based home hospitalization system. The tests encompassed multiple aspects to ensure a comprehensive assessment of the system's capabilities.

### 4.2 Usability Evaluation

To evaluate the proposed home hospitalization system, we have used the System Usability Scale (SUS), as this scale provides a fast and reliable tool for measuring ease of use and allows the evaluation of a variety of services and products, including mobile devices, mobile applications, and websites [39]. To detect most usability problems, it is acceptable to evaluate with five users[40]; in our case, ten doctors were selected to conduct usability testing.

To evaluate the proposed home hospitalization system, the system model was explained and a screenshots from the mobile application and the web dashboard was giving for the doctors, after which they were asked to complete the SUS questionnaire.

The questionnaire is composed of 10 questions to assess system usability, where each of the questions is classified based on the amount of agreement, from one (Strongly Disagree) to five (Strongly Agree), as shown in Figure4.1 . After patients and doctors have finished answering questions, the SUS scores are calculated as shown in Figure4.1.

(a) SUS questionnaire		Strongly Disagree			Strongly Agree		
		1	2	3	4	5	
1	I think that I would like to use this system frequently.						
2	I found the system unnecessarily complex.						
3	I thought the system was easy to use.						
4	I think that I would need the support of a technical person to be able to use this system.						
5	I found the various functions in this system were well integrated.						
6	I thought there was too much inconsistency in this system.						
7	I would imagine that most people would learn to use this system very quickly.						
8	I found the system very cumbersome to use.						
9	I felt very confident using the system.						
10	I needed to learn a lot of things before I could get going with this system.						

(b) Calculation of the SUS Score		(c) SUS Score		
<ul style="list-style-type: none"> <li>For each of the odd-numbered questions, subtract 1 from the score.</li> <li>For each of the even-numbered questions, subtract their value from 5.</li> <li>Take these new values which you have found, and add up the total score. Then multiply this by 2.5.</li> <li>Then we get a general value of usability on a scale between 100 (excellent usability) and 0 (ease of use-free).</li> </ul>		SUS Score	Grade	Adjective Rating
		>80.8	A	Excellent
		68-80.8	B	Good
		68	C	Ok
		51-68	D	Poor

Figure 4.1: The System Usability Scale (SUS)

Doctors	1	2	3	4	5	6	7	8	9	10
Question 1	5	5	5	4	5	5	5	5	5	5
Question 2	2	3	2	1	1	1	1	1	1	1
Question 3	4	2	4	5	5	5	5	5	4	5
Question 4	3	2	3	1	1	5	3	2	1	1
Question 5	3	5	5	4	5	5	5	5	5	5
Question 6	2	3	1	2	1	1	2	1	1	1
Question 7	4	1	3	4	5	5	2	5	3	5
Question 8	2	3	1	1	2	1	2	1	2	2
Question 9	3	2	5	3	4	5	4	4	4	5
Question 10	4	2	3	1	1	1	4	2	1	1
SUS score	65	55	80	85	95	90	72.5	92,5	87,5	97,5
Average SUS score	82									

Table 4.1: Patients' SUS score

Table 4.1 displays the values of the SUS questionnaires provided by each Doctor, the SUS value per Doctor, and the average SUS calculated for all Doctors.

The System Usability Scale evaluation was conducted for doctors, and the obtained evaluation score was 82. According to the established criteria, a score above 80.8 is required to achieve a degree A, which represents excellent usability. Therefore, based on the SUS evaluation results, it can be concluded that our system was highly accepted and well-received by doctors.

## 4.3 Hardware Functionality Evaluation

Ensuring the hardware functionality of our system is of paramount importance as it directly impacts the overall performance and reliability of the system. Conducting a comprehensive evaluation of the hardware functionality is crucial to identify and address any potential issues or shortcomings. By conducting thorough testing, we aim to achieve an error-free hardware system, as rectifying hardware failures can be both time-consuming and costly. This section presents the evaluation process and results of the hardware functionality, focusing on power consumption, performance, and error-handling aspects.

### 4.3.1 Power Consumption Testing and Analysis

In Power Consumption Testing and Analysis, our system was primarily designed to operate using a stable power supply rather than relying on batteries. However, understanding the power consumption details is still important for assessing the system's efficiency. To measure power consumption, we utilized a power measurement unit to monitor the current drawn during various system functionalities. The results revealed that the system exhibited a peak current of less than 1A at 5V, corresponding to approximately 5W of power. This power requirement can be easily fulfilled by commonly available power supplies, ensuring reliable and consistent operation of the system.

### 4.3.2 Reliability Testing

Reliability testing is a crucial aspect of the Hardware Functionality Evaluation, as it aims to assess the robustness and durability of the hardware components under various conditions. This testing ensures that the system can consistently perform its intended functions without failures or malfunctions.

We subjected our system to extended periods of operation and performed frequent measurements to assess its reliability. This involved running the system continuously for extended durations, simulating real-world usage scenarios where the system is expected to operate without interruptions.

During this time, we closely monitored the performance of the hardware components, including sensors, microcontrollers, and communication modules, to ensure they functioned reliably and consistently.



## 4.4 Firmware Functionality Testing and Validation

In the Firmware Functionality Testing and Validation, we focused on assessing the performance, functionality, and reliability of the firmware that drives our system. This involved various tests and validations to ensure that the firmware operates as intended and meets the desired functional requirements.

### 4.4.1 Optimizing Stack Size Allocation for Task Efficiency

During the Firmware Functionality Testing and Validation phase, we conducted a specialized test to determine the optimal stack size for each task in our application. This test involved running each task individually with a larger-than-usual stack size while monitoring the stack usage.

By running the tasks individually, we isolated the specific task and executed it with a generous stack size to ensure that it had ample memory space for its operations. Throughout the execution, we continuously measured the stack usage, keeping track of the maximum stack space utilized by the task.

Analyzing the maximum stack usage provided us with valuable insights into the peak memory requirements of each task. We then adjusted the stack size accordingly to strike a balance between memory allocation and efficiency. The goal was to assign a stack size that was neither too small, risking stack overflow, nor unnecessarily large, wasting memory resources.

By fine-tuning the stack sizes based on the results of this test, we optimized the memory utilization within our firmware. Each task was allocated an appropriate stack size that accommodated its specific needs without compromising the overall system's stability or performance.

### 4.4.2 A Comparative Analysis of Our Sensor and Other Established Sensors

While our system incorporates commercially available blood pressure and glucose sensors, the accuracy and reliability of our heart rate sensor and temperature sensors require thorough evaluation. In this section, we aim to assess the data accuracy of these sensors by comparing their readings to established sensors commonly used in the healthcare industry. By conducting these comparative analyses, we can gain insights into the performance of our sensors and ensure the delivery of reliable and precise measurements for comprehensive patient monitoring.

#### 4.4.2.1 Heart Rate Sensor Accuracy Evaluation

After conducting a comparison between the readings of our heart rate sensor and a commercially available one, we observed that our sensor consistently provided higher readings as shown in Figure 4.2a. Further analysis revealed that this discrepancy was attributed to the initial high values caused by finger vibrations on the sensor. To mitigate this issue, we implemented a rolling average with an outlier rejection filter, utilizing a window size of 5, on the recorded data collected over a 15-second interval. The results of this refinement approach yielded heart rate measurements that closely aligned with the commercial sensor readings, with a maximum difference of 1 bpm. These improved results are presented in Figure 4.2b.

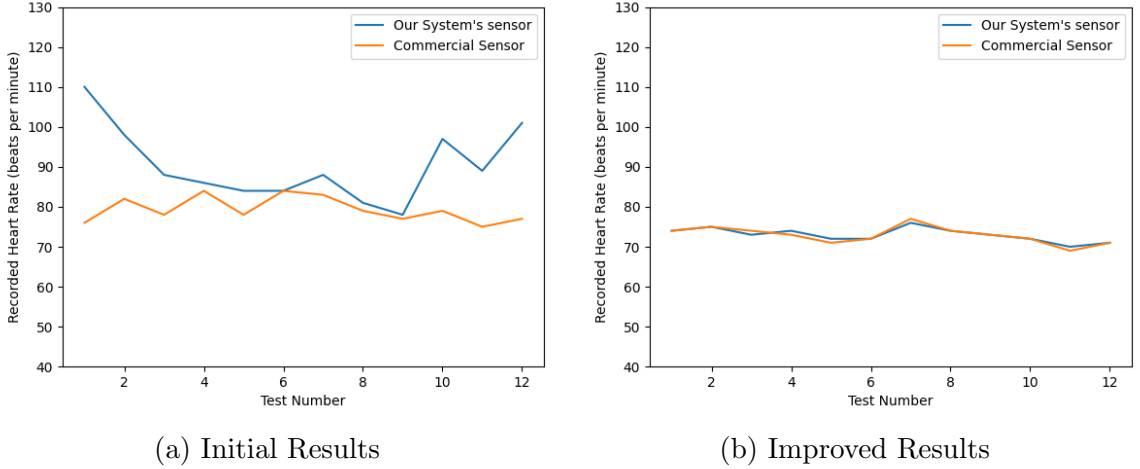


Figure 4.2: Heart Rate Sensor Accuracy Evaluation

#### 4.4.3 Error Handling Testing

In our firmware evaluation, one of our primary goals was to design an error-free system capable of recovering from potential errors. To achieve this, we implemented several strategies to handle different scenarios. First, we utilized the error handling capabilities provided by the espidf lwip framework. This framework includes a handler for network events, such as disconnecting from Wi-Fi, allowing our system to gracefully recover from a loss of Wi-Fi connection.

Furthermore, since our hardware components are encapsulated as black boxes, we devised a mechanism to communicate the state of each sensor during the handshake process. By transmitting the sensor states, we can notify the user about the availability and readiness of the sensors, ensuring a more reliable and informed experience.

Additionally, to address the possibility of connection loss during the measurement process, we implemented customized timeouts for each measurement task. These timeouts provide a means to handle interruptions or failures gracefully and ensure that the system can recover or prompt the user to take appropriate actions.

By incorporating these error-handling strategies into our firmware design, we aimed to enhance the overall robustness and reliability of our system, enabling smooth operation even in challenging situations.

## 4.5 AI Model Evaluation and Disease Prediction Results

In this section, we present the evaluation results of our AI model for disease prediction. We assess the performance of the model using various metrics, including accuracy, loss, confusion matrix, F1 score, and recall.

### 4.5.1 Training Performance Metrics: Accuracy and Loss Curves

In evaluating the AI model for disease prediction, we analyzed two key performance metrics: the accuracy per iterations curve and the loss per iterations curve. These curves provide valuable insights into the model's training progress and its ability to make accurate predictions.

The accuracy per iterations curve depicts how the model's accuracy evolves as it iteratively learns from the training data, showcasing improvements or plateaus in prediction accuracy over time. A rising accuracy curve indicates that the model is learning and making better predictions as it receives more training.

On the other hand, the loss per iterations curve represents the model's training progress by illustrating the decrease in loss function values, indicating how effectively the model minimizes the discrepancy between predicted and actual outcomes. A decreasing loss curve indicates that the model is learning and adjusting its parameters to make more accurate predictions.

Figure4.3 shows the accuracy and the loss per iterations curves.

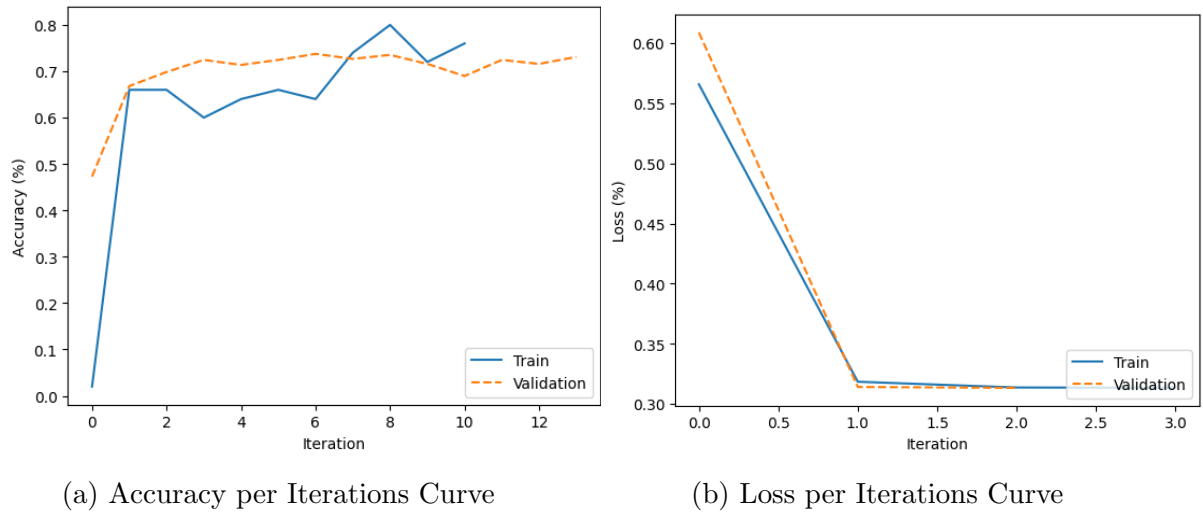


Figure 4.3: Training Performance Metrics: Accuracy and Loss Curves

### 4.5.2 Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's performance by showing the number of true positive, true negative, false positive, and false negative predictions. It enables us to assess the model's ability to correctly classify instances into different disease categories. From the confusion matrix, we can calculate various performance metrics such as precision, recall, and F1 score, which provide a comprehensive evaluation of the model's predictive capabilities.

Figure 4.4 depicts the confusion matrix obtained from our model's predictions.

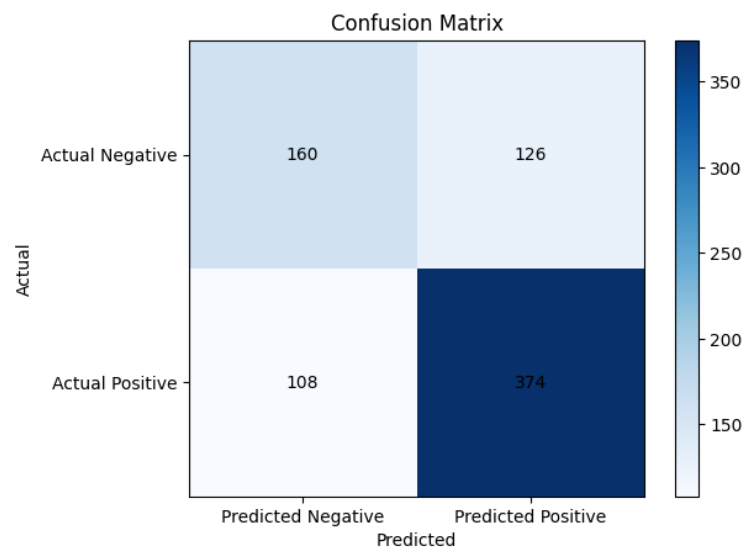


Figure 4.4: Confusion Matrix

- **Precision** is the proportion of true positive predictions (TP) out of the total predicted positives (TP + FP). It measures the accuracy of positive predictions. The formula for precision is as follows:  $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$  which gives us a value of 0.7480. This indicates that out of all the positive predictions made by the model, 74.80% were accurate.
- **Recall**, also known as sensitivity or true positive rate, is the proportion of true positive predictions (TP) out of the total actual positives (TP + FN). It measures the model's ability to correctly identify positive instances. The formula for the recall is as follows:  $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$ . Which gives us a value of  $0.7759 = 77.59\%$
- The **F1 Score** is the harmonic mean of precision and recall. It provides a balanced measure between precision and recall. It takes into account both the true positive rate and the false positive rate, making it a robust metric for assessing the model's predictive power. In our case, we got a value of  $0.7620 = 76.20\%$

## 4.6 Discussion

### 4.6.1 Diabetes Prediction Model

The accuracy metric with a value of 70%, suggests that our model is able to accurately classify diabetes cases around 70% of the time. However, it's important to note that accuracy alone may not be sufficient for assessing the model's performance, especially in cases of imbalanced datasets.

The precision metric, at 74.80%, signifies the percentage of correctly identified diabetic cases out of all the predicted positive instances. A higher precision indicates a lower rate of false positives, which is desirable for ensuring accurate predictions for individuals who are actually diabetic.

The recall metric stands at 77.59%. This metric represents the percentage of actual positive instances (diabetic cases) that our model correctly identifies. It highlights the ability of our model to capture and correctly classify individuals who have diabetes, reducing the rate of false negatives.

The F1 score is calculated as 76.20% in our case. It provides a balanced evaluation of both precision and recall, taking into account both false positives and false negatives. It is a useful metric for assessing the overall effectiveness of our model's predictions.

The achieved results can be attributed to several factors. Firstly, the relatively small size of our dataset may limit the model's ability to capture the full complexity and variability of diabetes patterns. Additionally, the imbalanced nature of the dataset, with a greater number of non-diabetic individuals, may affect the model's performance. These factors can impact the model's ability to generalize well to new and unseen data.

### **4.6.2 Effectiveness of the AIoT System in Home Healthcare**

We have conducted comprehensive evaluations and obtained notable results that shed light on the system's performance and impact. Specifically, we have examined various aspects such as disease prediction accuracy, remote patient monitoring capabilities, and user experience. By analyzing these findings, we can gain valuable insights into the effectiveness of our system and its potential to revolutionize home healthcare delivery.

Initially, the reliable connectivity and real-time monitoring capabilities of our system have proven instrumental in remote patient monitoring. By taking advantage of IoT devices and sensors, we were able to collect vital health data such as blood pressure, heart rate, and body temperature. This data was transmitted securely to the healthcare providers, enabling them to monitor patients' health conditions remotely and make informed decisions regarding their care.

The user-friendly interface of our mobile application and web dashboard provides easy access to health data, allowing patients to have a comprehensive understanding of their health status.

Additionally, the integration of AI and IoT technologies in our system has significantly improved the accuracy and efficiency of disease prediction. By employing a transformer neural network model, we achieved an accuracy of 70% in predicting diabetes indicating that our model can effectively identify individuals at risk of diabetes, enabling early intervention and timely medical support.

However, it is important to be aware of the limitations of our study. The small size and imbalanced nature of our dataset, particularly in the case of non-sick individuals, may have influenced the performance of our AI models. Future studies with larger and more diverse datasets would be beneficial to further validate and enhance the accuracy of our disease prediction models.

Overall, our AIoT system has demonstrated its effectiveness in home healthcare by improving disease prediction accuracy, enabling remote patient monitoring, and promoting patient engagement. These findings highlight the potential of AI and IoT technologies in transforming traditional healthcare delivery, providing more personalized and proactive care to individuals in the comfort of their homes.

### 4.6.3 Limitations of the System

Despite the promising capabilities of our AIoT-based home healthcare system, it is essential to acknowledge and address its inherent limitations. These limitations encompass various aspects of the system, including hardware components, mobile application functionality, the doctor's dashboard interface, and the underlying AI model.

First, one notable limitation in our system's hardware is the time required for body temperature measurement using the DS18B20 sensor. While the measurement process takes approximately 15 seconds, which may be considered lengthy for the patient, it is within the acceptable operating range for the DS18B20 sensor to obtain accurate oral temperature readings. However, this duration can be perceived as a drawback in terms of patient comfort and convenience during the measurement process.

In addition, the impact of crowded network environments on UDP-based data transmission. In networks with high traffic and congestion, UDP packets may be more susceptible to loss or interference. This can result in increased latency and the possibility of incomplete or inconsistent data transfer. Consideration should be given to the limitations of UDP in crowded networks and the exploration of alternative protocols or additional measures to ensure reliable data transmission.

The predictive models used in our system are based on available data and algorithms. Their accuracy is limited by the quality and representativeness of the training data. The AI model may have limitations in predicting rare or complex cases due to imbalances or biases in the training data.

Overall, while our AIoT-based home hospitalization system demonstrates promising capabilities, it is important to recognize and address these limitations in the hardware, mobile application, doctors' dashboard, and AI model to enhance the system's performance, accuracy, and reliability in real-world scenarios.

### 4.6.4 Future Enhancements and Potential Exploration Directions

In our pursuit of continuous improvement and innovation, we have identified several key areas for future enhancements in our AIoT home healthcare system. In the hardware domain, one of our primary objectives is to transition from consumer-grade sensors to industrial-grade and reliable sensors. This upgrade will ensure more accurate and robust data collection, contributing to higher precision in disease detection and monitoring. Additionally, we plan to expand the range of sensors integrated into our system, allowing for comprehensive health monitoring that encompasses a

broader spectrum of vital signs and physiological parameters.

To enhance the communication aspect of our system, we recognize the advantages of utilizing TCP (Transmission Control Protocol) over UDP (User Datagram Protocol) in terms of reliability and data integrity. Consequently, we have designed our firmware and application to be adaptable, making it straightforward to switch from UDP to TCP communication protocols as necessary. This flexibility ensures a more stable and resilient communication framework, enabling seamless data transmission between the IoT devices and the central monitoring system.

We plan to transition from Firebase services to our custom backend infrastructure, enabling greater control over data handling, scalability, and security. The clean architecture of our mobile app and web dashboard facilitates easy integration with alternative backend solutions, ensuring flexibility and adaptability to future advancements and industry standards. Additionally, we aim to incorporate industrial-grade and reliable sensors, expand sensor capabilities, and consolidate all components into a single-block PCB design. To enhance communication reliability, we have designed our firmware and application to smoothly switch from UDP to TCP. Furthermore, our future plans include implementing additional AI predictions for various diseases, and expanding the system's diagnostic capabilities.

Furthermore, expanding the scope of our AI predictions beyond diabetes is a crucial future objective. By leveraging advanced machine learning algorithms and datasets, we can develop prediction models for a wider range of diseases. This expansion will empower our system to provide comprehensive health insights and early detection capabilities for various medical conditions, thereby improving patient care and enabling timely interventions.

Through these targeted improvements and ambitious future directions, we aim to continually enhance the performance, reliability, and versatility of our AIoT home healthcare system, ultimately fostering better patient outcomes and empowering individuals to proactively manage their health.



# Conclusion

In conclusion, our AIoT-based home hospitalization system presents a promising solution for remote healthcare monitoring and disease prediction. By taking advantage of IoT devices, sensors, and advanced machine learning techniques, we have developed a comprehensive system that smoothly integrates data collection, analysis, and visualization. The system's user-friendly interface and real-time monitoring capabilities enable patients to actively engage in their healthcare journey and make informed decisions. Our evaluation and testing results have demonstrated the system's effectiveness and reliability, with high accuracy in disease prediction and robust error-handling capabilities.

Moving forward, we envision further enhancements and potential research directions. These include the adoption of industrial-grade and reliable sensors, the addition of new sensor modalities, the consolidation of components into a single-block PCB design, and the development of a custom backend infrastructure for improved data management and scalability. Furthermore, the flexibility of our clean architecture enables easy integration of alternative communication protocols, such as TCP, for enhanced reliability. Expanding the AI prediction capabilities to encompass a wider range of diseases will unlock new possibilities in proactive healthcare management.

Overall, our AIoT-based home hospitalization system lays the foundation for providing individuals with personalized, remote healthcare solutions. It bridges the gap between medical professionals and patients, enabling efficient monitoring, early intervention, and improved overall well-being. With continued research and development, this system has the potential to revolutionize the way healthcare is delivered, making it more accessible, efficient, and patient-centered.

# Bibliography

- [1] W. Guan, Z. Ni, Y. Hu, W. Liang, and et al. Clinical characteristics of coronavirus disease 2019 in china. *New England Journal of Medicine*, 2020.
- [2] L. Wang, W. He, X. Yu, D. Hu, and et al. Coronavirus disease 2019 in elderly patients: characteristics and prognostic factors based on 4-week follow-up. *Journal of Infection*, 2020.
- [3] P. García-Sánchez, J. González, A.M. Mora, and A. Prieto. Deploying intelligent e-health services in a mobile gateway. *Expert Systems with Applications*, 40(4):1231–1239, 2013.
- [4] Abigail Saltmarsh. The “aiot” and its role in the future of healthcare, Jan 2023.
- [5] Anil Pise, Byungun Yoon, and Saurabh Singh. Enabling ambient intelligence of things (aiot) healthcare system architectures. *Computer Communications*, 198:186–194, 2023.
- [6] Ai in healthcare: Trends we saw in 2022. *HealthSnap, Inc.*, Mar 2023.
- [7] Sarah Klein. Hospital at home programs improve outcomes, lower costs but face resistance.
- [8] Hamed Ben Hassen, Nadia Ayari, and Belgacem Hamdi. A home hospitalization system based on the internet of things, fog computing and cloud computing. *Informatics in Medicine Unlocked*, 20:100368, 2020.
- [9] Majid Bayani Abbasy and Enrique Vilchez Quesada. Predictable influence of iot (internet of things) in the higher education. *International Journal of Information and Education Technology*, 7:914–920, 2017.
- [10] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions, 2012.
- [11] Sambhavna Gupta, Nishant Mudgal, and Rishabh Mehta. Analytical study of iot as emerging need of the modern era. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 233–235, 2016.

- [12] Kristjan Prosen. Using arduino development board for computer aided measurement in school. 2016.
- [13] Kundenzeitung · Ausgabe. 50 years of history and a bright future into the iot age with experience. 2022.
- [14] Kiran Maraiya and M. M. Tripathi. A survey of iot and its cutting-edge applications. *Asian Journal of Science, Technology and Society*, 2022.
- [15] Espressif Systems. ESP32 Documentation. <https://www.espressif.com/en/products/socs/esp32>, 2023.
- [16] Espressif Systems. ESP32 IoT Development Framework Documentation. <https://docs.espressif.com/projects/esp-idf/>, 2023.
- [17] FreeRTOS. FreeRTOS Website. <https://www.freertos.org>, 2023.
- [18] Espressif Systems. ESP Non-volatile Storage Library official documentation. [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs\\_flash.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/nvs_flash.html), 2023.
- [19] Espressif Systems. ESP-NETIF official documentation. [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp\\_netif.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_netif.html), 2023.
- [20] Texas Instruments. *Universal Asynchronous Receiver/Transmitter (UART)*, 2010. Accessed: 2023-05-29.
- [21] Texas Instruments. *Understanding the I2C Bus*, 2015. Accessed: 2023-05-29.
- [22] Analog Devices. 1-Wire Communication Through Software. <https://www.analog.com/en/technical-articles/1wire-communication-through-software.html>, 2023.
- [23] Analog Devices. *Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health*, 2014. Accessed: 2023-05-29.
- [24] Analog Devices. *DS18B20, Programmable Resolution 1-Wire Digital Thermometer*, 2015. Accessed: 2023-05-29.
- [25] Linkedin. Mastering Asynchronous Programming in Flutter with Dart's async, await, and future Keywords. <https://www.linkedin.com/pulse/mastering-asynchronous-programming-flutter-darts-async-michael>, 2023.
- [26] Sebastian Faust. Cologne University of Applied Sciences. Using google's flutter framework for the development of a large-scale reference application. 2020.
- [27] Flutter Documentation. State management - flutter documentation, 2023.

- [28] GitConnected. Building efficient flutter apps with the bloc pattern: Implementation, advantages, and best practices, 2023.
- [29] YoungJin Lee, JongO Han, Hyekyung Lee, and SooHyun Lim. The development and operation of a home management system during the covid-19 pandemic: Experience of the local government gyeonggi-do in korea. *J Korean Med Sci*, 36(19):e134, May 2021.
- [30] Mohammad Wazid, Ashok Kumar Das, and Youngho Park. Blockchain-envisioned secure authentication approach in aiot: Applications, challenges, and future research. *Wireless Communications and Mobile Computing*, 2021.
- [31] Sakshi Panditrao Golhar and Shubhada Sudhir Kekapure. Artificial intelligence in healthcare - a review. *International Journal of Scientific Research in Science and Technology*, 2022.
- [32] Anand Prakash Dube and Raghav Yadav. Analyzing the effectiveness of ml agents in enhancing the predictive model in decision making for medical practitioners in the healthcare industry: A structural equation model analysis. *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pages 1017–1021, 2022.
- [33] Rudra Tiwari. The use of ai and machine learning in healthcare and its potential to improve patient outcomes. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, 2023.
- [34] Santhosh Gupta Dogiparthi, Jayanthi M. K., and Ajith Ananthakrishna Pillai. A comprehensive survey on heart disease prediction using machine intelligence. *International Journal of Medical Research and Health Sciences*, 10:60–68, 2021.
- [35] C. S. Lee and A. Y. Lee. How artificial intelligence can transform randomized controlled trials. *Translational Vision Science & Technology*, 9(2):9, 2020.
- [36] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- [37] Minhyeok Lee. Gelu activation function in deep learning: A comprehensive mathematical analysis and performance, 2023.
- [38] Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Smooth maximum unit: Smooth activation function for deep networks using smoothing maximum technique. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 784–793, 2022.
- [39] James R. Lewis. The system usability scale: Past, present, and future. *International Journal of Human-Computer Interaction*, 34(7):577–590, 2018.

- 
- [40] Zaihasriah Zahidi, Yan Peng Lim, and Peter Charles Woods. Understanding the user experience (ux) factors that influence user satisfaction in digital culture heritage online collections for non-expert users. In *2014 Science and Information Conference*, pages 57–63, 2014.

# Appendices

# Appendix A

## Physical Implementation of our System

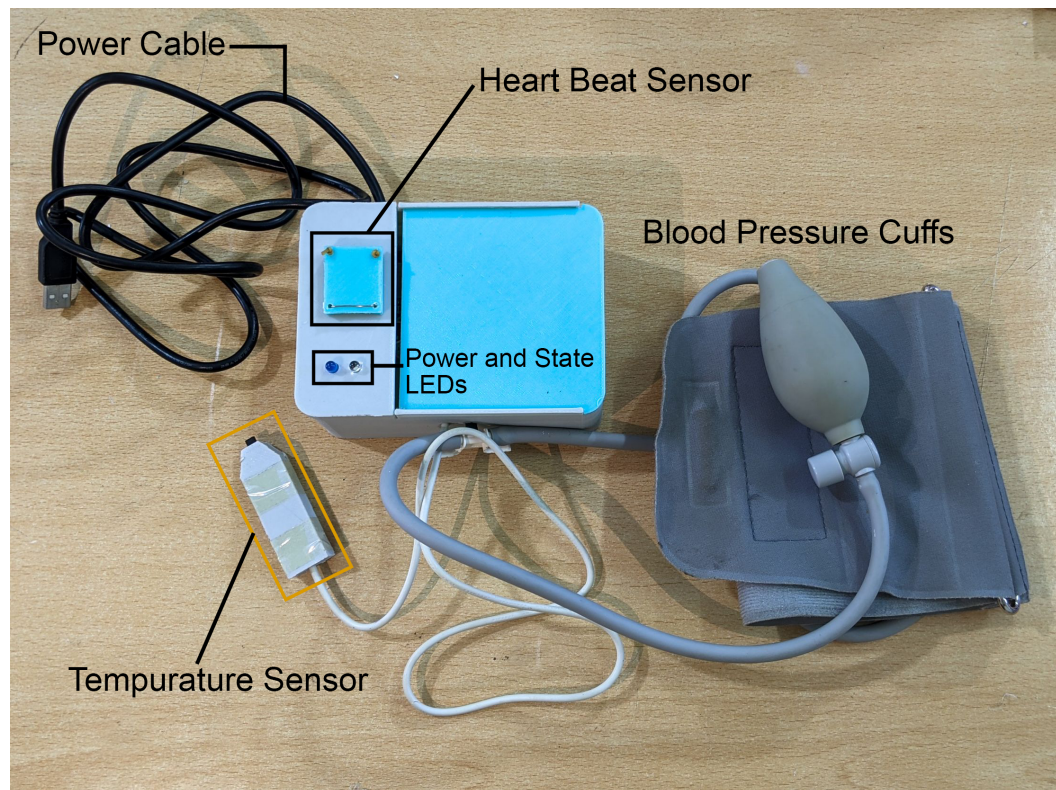


Figure A.1: Physical Implementation of our System

# Appendix B

## Mobile Application Screenshots

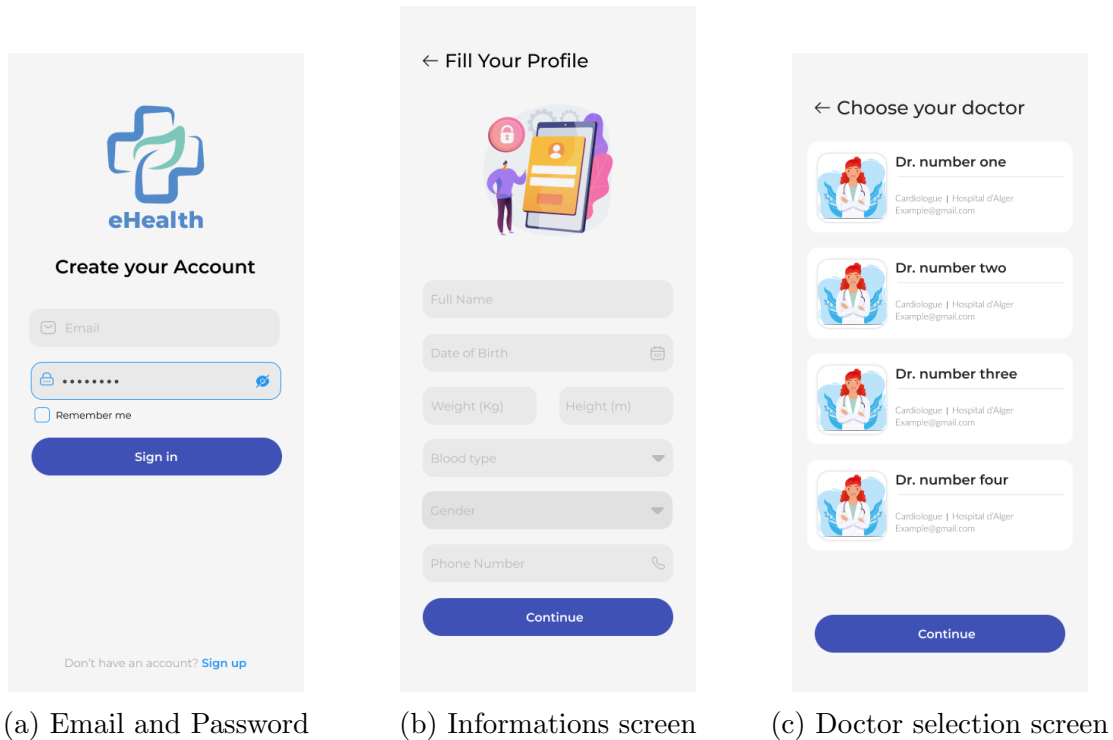
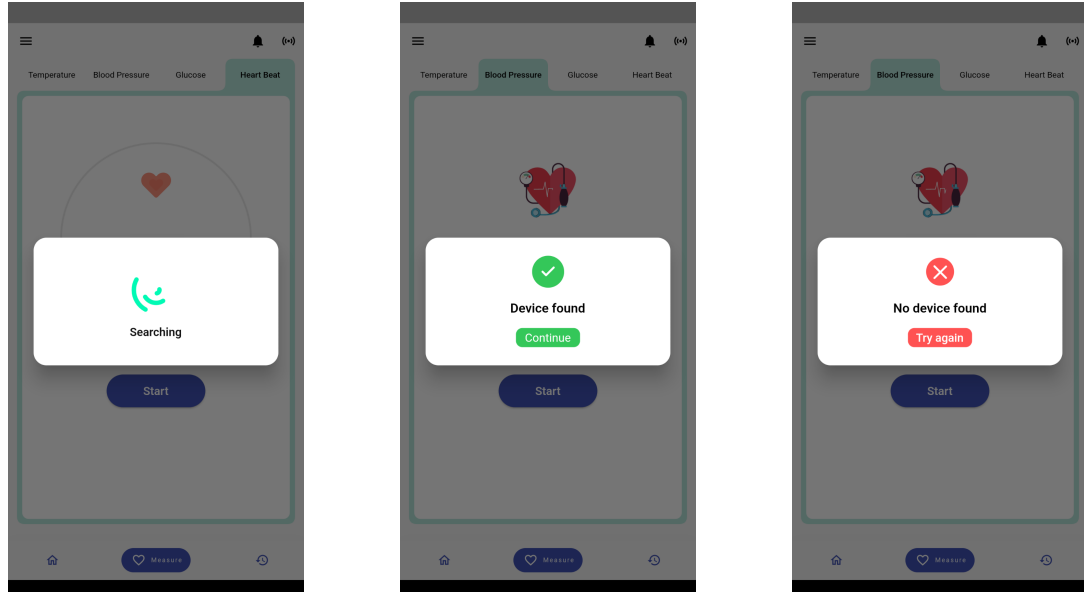


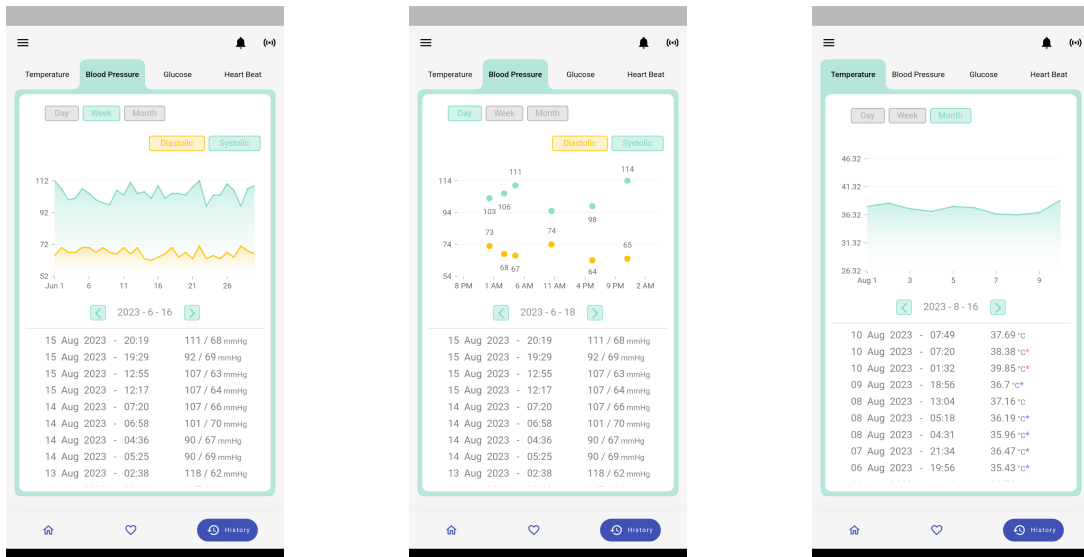
Figure B.1: Sign up process screens





(a) Search for Devices in the Network (b) Hardware Device was Found (c) Hardware Device not Found

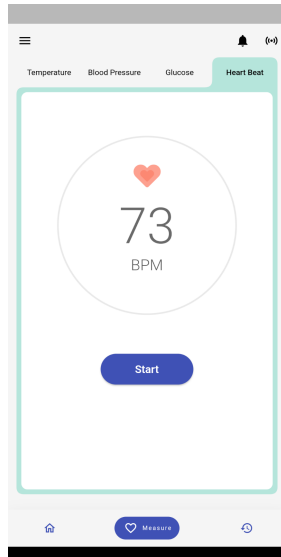
Figure B.2: Handshake Process Screens



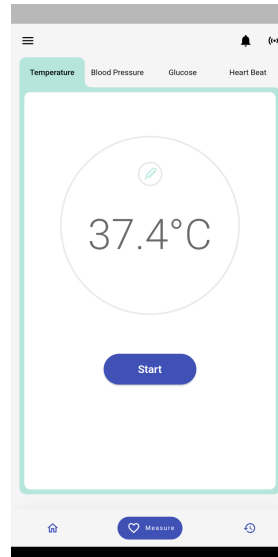
(a) Blood Pressure Week History Screen (b) Blood Pressure Day History Screen (c) Body Temperature Month History Screen

Figure B.3: Data History Screens

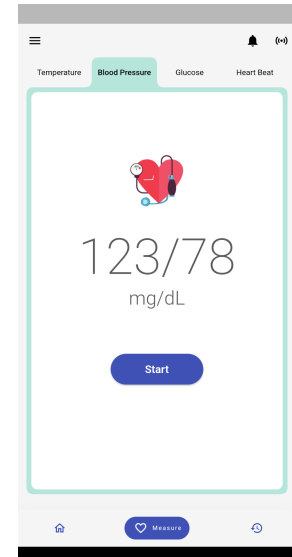
Note: In the Month and Week History screens, the data is presented in a graphical form, allowing users to visualize trends and patterns over time. In the Day View, the data is displayed as dots, providing a concise overview of the recorded measurements throughout the day.



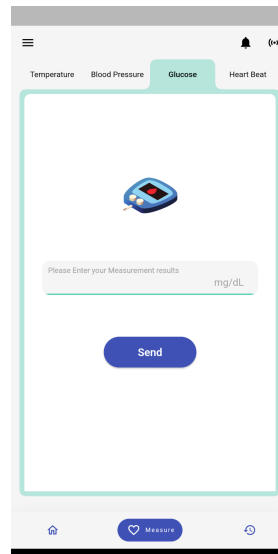
(a) Heartbeat Measurement Screen



(b) Body Temperature Measurement Screen



(c) Blood Pressure Measurement Screen

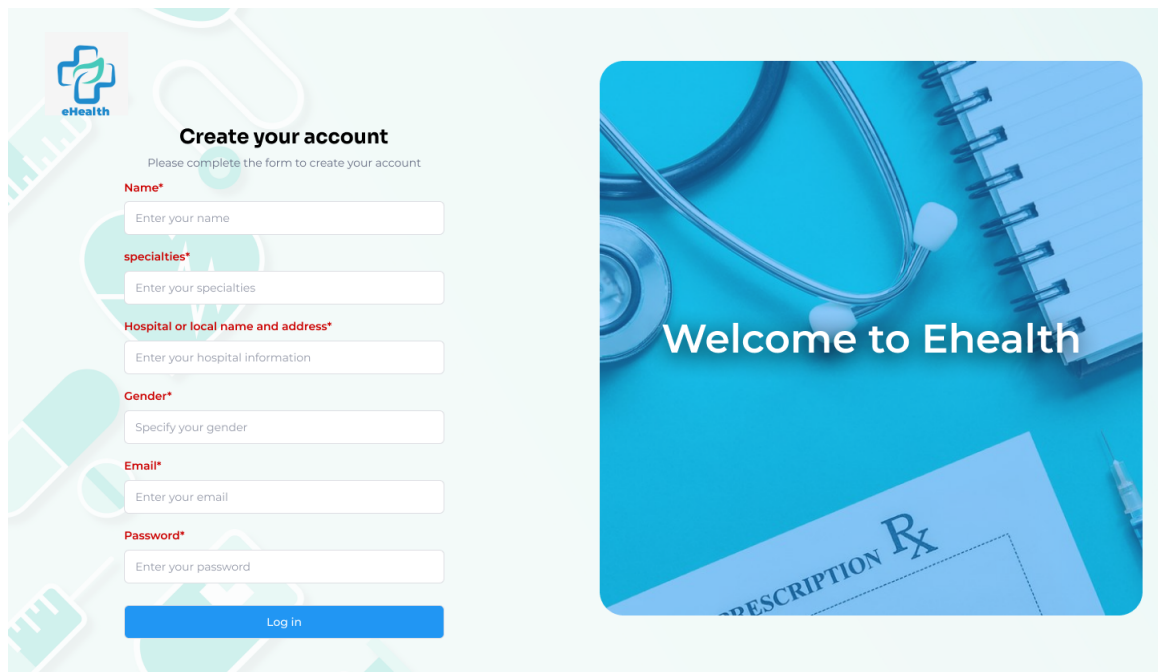


(d) Glucose Manual Input Screen

Figure B.4: Data Measurements Screens

# Appendix C

## Web Dashboard Screenshots



**Create your account**  
Please complete the form to create your account

**Name\***  
Enter your name

**specialties\***  
Enter your specialties

**Hospital or local name and address\***  
Enter your hospital information

**Gender\***  
Specify your gender

**Email\***  
Enter your email

**Password\***  
Enter your password

Log in

Welcome to Ehealth

Figure C.1: Doctor Sign up screen

**Bonjour, Dr.Rezrazi 😊**

**Today appointments** [See all](#)

Patient	Time	Age	Disease	Action
Berrachoua E Med Last visited 26-12-2022	10:30	42	fever	⋮
Djelti Amine Last visited 13-04-2023 <span>🔴 Serious case</span>	11:00	55	Anemia	⋮
Lagab Sarra Oumnia Last visited 26-12-2022	11:30	42	fever	⋮
Lotfi Elhamel Last visited 26-12-2022	10:30	42	fever	⋮
Tikou chlef Last visited 13-04-2023 <span>🔴 Serious case</span>	11:00	55	Anemia	⋮

← Previous 1 2 3 Next →

**Alerts** 2

- Rez med may have diabets [view](#)
- Rez med may have diabets [view](#)

**Notifications** 7

- Rez med sent a join request [view](#)
- Rez med sent a message [view](#)
- Rez med sent a message [view](#)
- Rez med sent a message [view](#)
- Tikou chlef sent a join request [view](#)

Figure C.2: Doctor Dashboard screen

**My Patients**

Find a patient

Gender: Tous | Age: Tous

Patient	Gender	Age	Location	Blood Type	Phone	Action
Rez Med example@gmail.com	Male	45 y.o	Alger	AB+	0781234567	<a href="#">View Data</a>
Rez Med example@gmail.com	Male	45 y.o	Alger	AB+	0781234567	<a href="#">View Data</a>
Rez Med example@gmail.com	Male	45 y.o	Alger	AB+	0781234567	<a href="#">View Data</a>
Rez Med example@gmail.com	Female	45 y.o	Alger	AB+	0781234567	<a href="#">View Data</a>
Rez Med example@gmail.com	Female	45 y.o	Alger	AB+	0781234567	<a href="#">View Data</a>

← Previous 1 2 3 4 5 6 7 8 9 10 11 Next →

Figure C.3: Doctor Patients screen

This is the Patients screen where the doctor can find a list of their patients along with their information. The doctor can filter and search for a specific patient and can click on any patient to view more data.

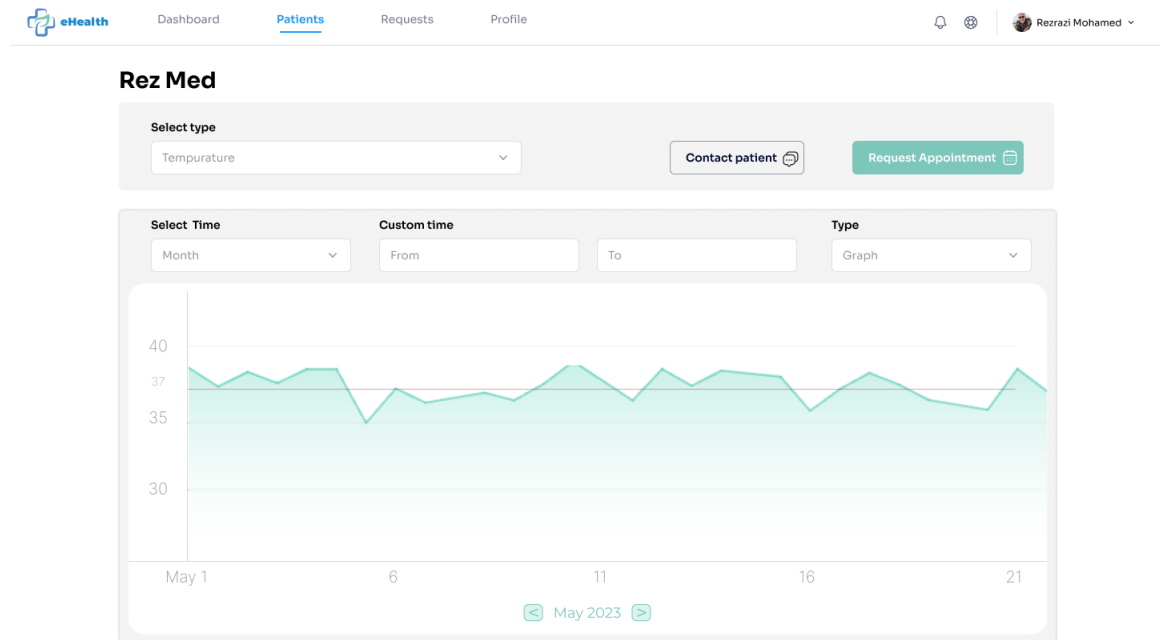


Figure C.4: Doctor History screen (Graph)

**Rez Med**

Select type: Temperature

Contact patient | Request Appointment

Select Time: Month | Custom time: From | To | Type: Graph

Date	Time	Value
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°
10 Aug 2023	07:23 AM	37.7 C°

Figure C.5: Doctor History screen (Table)