Advanced IoT Health Tracking System with Real-Time Data Display

by

Xari Nikolaki Mouchtouris

23795824

Report presented in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering (Electrical and Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Dr W. Duckit

November 2024

Acknowledgements

First, I would like to express my deepest gratitude to **Jesus Christ, my Saviour**, for His unwavering love, grace, and guidance throughout this project and my undergraduate studies. His presence has been my strength and foundation.

I am deeply grateful to my mother, grandmother, brother, and my late father for their sacrifices, love, and constant support over the years. Their encouragement has been invaluable throughout my academic journey.

I am also sincerely grateful to the **IDC** for their continued financial support during my studies. This support has been instrumental and I am forever grateful.

My heartfelt thanks go to my supervisor, **Dr. William Duckitt**, for his guidance, encouragement, and expertise throughout this report. His mentorship has been crucial to the successful completion of this work.

Finally, I wish to thank my friends, who have supported me throughout this project and my degree, who are always ready with support, encouragement, and a listening ear during exams and challenging times.

Stellenbosch UNIVERSITY IYUNIVESITHI UNIVERSITEIT | forward together sonke siya phambili saam vorentoe

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present it as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly, all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

23795824 Studentenommer / *Student number*	Handtekening / *Signature*
XN MOUCHTOURIS Voorletters en van / *Initials and surname*	2024/11/03 Datum / *Date*

Abstract

English

This project outlines the design, implementation and verification of an IoT-based health monitoring device, capable of measuring temperature, heart rate, and oxygen saturation levels SPO_2. The device aimed to provide an accessible and cost-effective alternative to conventional monitoring solutions. The ESP32-C3 microcontroller is the central communication hub that connects all sensors, the LCD, and the web server. The data is presented in real-time on both an external display and an IoT web server, as well as historical data on the web server, using Google Firebase as the data storage platform. The Web server implements visualization of both real-time and historical data while allowing users to view trends, averages, and abnormal measurement alerts. Calibration with established medical devices confirmed the accuracy and reliability of the measurements, validating the system as an affordable, accessible option for personal health monitoring. Future improvements could include adding health metrics, improving portability via battery power, a smaller enclosure design, and enabling integration with healthcare provider systems.

Afrikaans

Die projek beskryf die ontwerp, implementering en verfikasie van 'n IoT-gebaseerde gesondheid toestel, wat temperatuur, hartkop en suurstofversadigingsvlakke (SPO_2) kan meet. Die toestel is ontwerp om 'n toeganklike en koste-effektief alternatief vir konvensionel gesondheids stelsels te bied. Die ESP32-C3 mikrobeheerder dien as die sentrale kommunikasie middelpunt, wat all die sesnors, die LCD skerm en die webbediener verbind. Die data word in werklike tyd op beide die eksterne skerm en die webbediener vertoon, asook historiese data, met Google Firebase as die data stoor spasie. Die webbediner bied visualisering van beide regte tyd en historiese data, wat gebruikers toelaat om tendense, gemiddeldes en waarskuwings oor abnormale metings te kan sien. Kalibrasie met gevestigde mediese toestelle het die akkuraatheid en betroubaarheid van die metings bevestig, wat die stelsel as 'n bekostigbare en toeganklike opsie vir persoonlike gesondheidsmonitering verifieer. Toekomstige verbeterings kan insluit die byvoeging van bykomende gesondsheids statistieke, verbeterde draagbaarheid deur battery aandrywinging, 'n kleiner omhulsel ontwerp en integrasie met gesondheidsorgverskafferstelsels.

Contents

Declaration ii

Abstract iii

List of Figures vii

List of Tables x

Nomenclature xi

1. Introduction 1

1.1. Background . . . 1
1.2. Problem Statement . . . 1
1.3. Project Objectives . . . 1
1.4. Scope . . . 2
1.5. Report Overview (Roadmap) . . . 2

2. Literature Review: 3

2.1. Definition of IoT devices: . . . 3
2.2. Description of Architecture block diagram . . . 3
2.2.1. IoT Sensors . . . 3
2.2.2. Microcontroller . . . 4
2.2.3. IoT Connectivity . . . 4
2.3. Health Sensors . . . 5
2.3.1. Temperature Sensors . . . 5
2.3.2. Oximeter Sensors . . . 6
2.3.3. Heart Rate Sensors . . . 8
2.4. Related Work . . . 10
2.4.1. IoT Health Wearable Monitoring System . . . 10

3. Hardware Selection: 12

3.1. System Diagram . . . 12
3.2. Component Selection . . . 12
3.2.1. Selection of Microcontroller . . . 13
3.2.2. Selection of Temperature Sensor . . . 14

3.2.3. Selection of Oxygen and Heart Rate Sensor 15
3.2.4. Selection of Display 16
3.2.5. Selection of Storage 17
3.2.6. Selection of Power Supply 17

4. Detail Hardware Design: 18
4.1. Detailed Hardware Design 18
4.1.1. Power Supply Design 19
4.1.2. Microcontroller 19
4.1.3. Heart rate and oximter module (SEN-15219) 21
4.1.4. Temperature Sensor (D6T-1A-01) 21
4.1.5. Display (Adafruit LLC 2050) 22
4.1.6. PCB Design 23
4.1.7. Enclosure Design 24
4.1.8. Physical Build 24

5. Detail Software Design: 25
5.1. Software Overview 25
5.2. Sensor Initialization and Communication 26
5.2.1. I2C Communication Setup 26
5.2.2. Sensor Software Initialization 27
5.2.3. Calibration of the temperature sensor 28
5.3. Web Server Setup: 29
5.4. Graphical Interface of Web Server: 29
5.4.1. User Registration and Login Pages 30
5.4.2. Dashboard Page 30
5.4.3. CSS Styling and Responsiveness 30
5.4.4. JavaScript and Chart.js Integration 30
5.5. Storage Management for Historical Mode 31
5.5.1. Real-time vs. Historical Data 31
5.5.2. Google Firebase for data storage 31
5.6. Error Handling and Alerts 31
5.6.1. Sensor Data Validation 32
5.6.2. Alerts for Abnormal Measurements 32
5.7. Adafruit LLC 2050 Display Software 32
5.7.1. SPI Setup 32
5.7.2. Software Structure for Display 33

6. Measurements and Results: 34
6.1. Method of Measurement . . . 34
6.2. Oscilloscope Measurements . . . 34
6.2.1. Power Supply . . . 34
6.2.2. I2C Lines . . . 35
6.3. Sensor Data Analysis . . . 35
6.4. Comparison with Standard Health Devices . . . 36
6.5. Web Server Functionality . . . 38
6.5.1. Real-Time Data Display . . . 38
6.6. Historical Data Display . . . 38

7. Conclusion 39
7.1. Summary . . . 39
7.2. Future Work . . . 39

Bibliography 41

A. Project Planning Schedule 45

B. Outcomes Compliance 46

C. Comparison of components considered for Hardware Selection 50

D. Full Schematic of device 52

E. Measurements of health monitoring device 53

F. Verification of Measurement Accuracy Against Established Medical Devices 56

G. Cad Design 57

H. PCB Design and 3D model 58

I. Physical build enclosure with PCB 61

J. Health Monitoring Web Server 63

K. Medical devices Used for Testing and Calibration 65

L. Github Software Repository 67

List of Figures

2.1. Architecture block diagram of a IoT device. 3
2.2. Architecture of an MCU, reproduced from [1]. 4
2.3. MCU Station Mode [2] 5
2.4. Semiconductor-based temperature sensor, [3]. 6
2.5. Architecture of an IF sensor, reproduced from [4]. 6
2.6. Transmissive pulse oximetry, [5] 7
2.7. Reflective pulse oximetry, [5] 7
2.8. Oximetry DC and AC component signal, [6]. 7
2.9. Block diagram of pulse oximeter sensor, reproduced from [7]. 8
2.10. Block diagram of heart rate sensor. 9
2.11. Analogue and Digital Heart Rate Signal. 10

3.1. Block diagram of system overview. 12

4.1. Detailed system diagram of IoT health device 18
4.2. LM7805CV Voltage Regulator Circuit 19
4.3. ESP32-C3 development board [8] 20
4.4. Block diagram of SPI connections 21
4.5. Schematic with connections of ESP32-C3 21
4.6. Circuit schematic of SEN-15219 22
4.7. Circuit schematic of D6T-1A-01 22
4.8. Circuit schematic of Adafruit LLC 2050 23
4.9. PCB design of Health System 24

5.1. Flow diagram of software overview diagram 25
5.2. Flow diagram of temperature acquisition and processing 28
5.3. Body Temperature vs Skin Temperature plot 28
5.4. System Architecture Diagram of the ESP32 Web server 29
5.5. Webpage layout for the health monitoring web server 30
5.6. Data Management Architecture for the health monitoring device 31
5.7. LCD Cursor illustration 33

6.1. Oscilloscope measurement of 5V power rail 35
6.2. I2C Lines Measurement 35
6.3. Noise on I2C Lines 35

6.4. Heart Rate Measurements (10 min) 36
6.5. Oxygen Level Measurements (10 min) 36
6.6. Temperature Measurements (10 min) 36
6.7. Heart Rate Comparison 37
6.8. Oxygen Level Comparison 37
6.9. Temperature Comparison 37
6.10. Web server Dashboard during measurement 38
6.11. Temperature Measurement within Detail View 38

D.1. Full Schematic of the Device 52

F.1. Heart Rate and SPO_2 comparison picture: (Timestamp: 12:10:03 am) . . 56
F.2. Heart Rate and SPO_2 comparison picture: (Timestamp:12:10:22 am) . . . 56
F.3. Heart Rate and SPO_2 comparison picture: (Timestamp: 12:10:29 am) . . . 56
F.4. Temperature Comparison with thermometer 56

G.1. Internal View of PCB Inside Enclosure 57
G.2. Orthographic view 1 57
G.3. Enclosure Cad Design 1 57
G.4. Top-down view of PCB with components 57
G.5. Orthographic view 2 57
G.6. Enclosure Cad Design 2 57

H.1. Top Layer of PCB 58
H.2. Bottom Layer of PCB 58
H.3. Top view of 3D model 59
H.4. Right side view of 3D model 59
H.5. Left side view of 3D model 60

I.1. Orthographic View of Working Device 61
I.2. Normal Measurements: Green 61
I.3. Abnormal Measurements: Red 61
I.4. Physical soldered PCB for Health Device 62

J.1. Dashboard page 63
J.2. Health information page 63
J.3. Detailed view of Heart Rate page 63
J.4. Detailed view of Oxygen levels page 63
J.5. Detailed view of Temperature page 64
J.6. Historical Main page 64
J.7. Historical Heart Rate Page 64
J.8. Historical Temperature Page 64

K.1. SPO_2 and Heart Rate measurement device [9] 65
K.2. Medical Thermometer [10] 65
K.3. Heart Rate measurement with Apple Watch Series 3 [11] 66

List of Tables

4.1. SPI Pin Configuration for the Adafruit LLC 2050 Display 22
4.2. PCB Track Design Specifications 23

5.1. Normal Ranges and Warning Thresholds for Health Measurements 32

6.1. Measurement of health sensors over 10 minutes 36
6.2. MAE Comparison for Heart Rate, Oxygen Levels, and Temperature 38

A.1. Project Planning Schedule for IoT Health Device 45

B.1. ECSA Outcomes Compliance 49

C.1. Comparison of Different Microcontrollers 50
C.2. Comparison of Temperature Modules 51
C.3. Comparison of Oximeter Modules 51

E.1. Comparison of Measurements: Established Medical Devices vs. IoT Health Device 53
E.2. Comparison of Oxygen Saturation (SpO_2) Levels: IoT Device vs. Medical Device 54
E.3. Comparison of Temperature Measurements: IoT Device vs. Medical Device 55

Nomenclature

Variables and functions

T	Temperature in degrees Celsius (°C).
HR	Heart Rate in Beats Per Minute (BPM).
SpO_2	Peripheral Capillary Oxygen Saturation in percentage (%).
V_{in}	Input Voltage to the system.
V_{out}	Output Voltage from the system.
R	Resistance in Ohms (Ω).
P	Power in Watts (W).
t	Time in seconds (s).
f	Frequency in Hertz (Hz).

Acronyms and abbreviations

AC	Alternating Current
ADC	Analogue-to-Digital Converter
ARM	Advanced RISC Machines
BPM	Beats Per Minute
DC	Direct Current
GPIO	General Purpose Input/Output
I2C	Inter-Integrated Circuit
I2S	Integrated Inter-IC Sound Bus
IoT	Internet of Things
IR	Infrared
LED	Light Emitting Diode
LCD	Liquid Crystal Display
MAE	Mean Absolute Error
MCU	Microcontroller Unit
MEMS	Micro-Electro-Mechanical System
NTP	Network Time Protocol
OHb	Oxyhemoglobin
PPG	Photoplethysmography
RISC	Reduced Instruction Set Computer
RHb	Deoxyhemoglobin (Reduced Hemoglobin)
ROM	Read-Only Memory
RAM	Random Access Memory
SpO_2	Peripheral Capillary Oxygen Saturation
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TFT	Thin-Film Transistor
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver/Transmitter
Wi-Fi	Wireless Fidelity

Chapter 1

Introduction

1.1. Background

The global market for patient monitoring is projected to grow significantly, driven by increasing demand for health tracking and advances in the Internet of Things (IoT) technology. In 2023, the market was valued at $43.7 billion and is expected to expand at a compound annual growth rate (CAGR) of 7.7% between 2024 and 2032 [12]. IoT has enabled the development of real-time health monitoring devices, such as smartwatches, providing users with an effective and convenient method to track vital health parameters. However, many of these systems remain costly and inaccessible to a large portion of the population, particularly in regions with limited access to healthcare. According to the World Health Organization (WHO), at least 50% of the global population lacks access to essential health services [13]. This illustrates the need for an accessible and cost-effective solution that delivers key health measurements to individuals.

1.2. Problem Statement

This report presents the hardware and software design, implementation, and testing of an advanced IoT-based health monitoring system, addressing the limitations of existing solutions in terms of cost, accessibility, and data management. The system is designed to display real-time data, store historical records, and provide users with key information through trends, averages, and health metrics. It will measure critical health parameters, including oxygen levels (SPO_2), heart rate, and temperature. With an integrated web server and cloud storage via Google Firebase, users will have the ability to remotely access their health metrics and track trends over time. This solution aims to close the gap in current health monitoring systems, which are often expensive, inaccessible and lack comprehensive historical data analysis.

1.3. Project Objectives

The IoT health monitoring system must achieve the following objectives to effectively address the identified problem:

1. Develop an **IoT-based health monitoring device** capable of measuring oxygen levels (SPO_2), heart rate, and temperature.
2. Provide **real-time health data display** on both a local display and a web server.
3. Implement **secure storage for historical data** using cloud storage to enable long-term tracking and trend analysis.
4. Enable **remote access to health metrics** via the web server.
5. Offer **user-friendly data visualization** to display trends, averages, and key insights.
6. Ensure **data security** by password-protecting both the web server and health metrics.
7. **Calibrate, compare, and test the device** using established medical measurement devices to validate precision.

1.4. Scope

The project's scope involves designing, developing, and testing an IoT-based health monitoring system capable of measuring oxygen levels (SPO_2), heart rate, and temperature. The device integrates MEMS sensors, a microcontroller, a real-time display, and a web server. The project includes hardware and software development, focusing on providing historical and real-time health data through the web server. The web server alerts users of abnormal results. The server ensures data privacy with password protection. The effectiveness and accuracy of the device will be validated against an approved medical device. The project excludes diagnostic tools for diagnosing specific illnesses. Additionally, the system lacks integration and transmission of health data to current healthcare provider systems. Instead, the focus is centred on individual health tracking and monitoring.

1.5. Report Overview (Roadmap)

Chapter 1 introduces the project, summarizing the problem statement, objectives, and scope of the work. **Chapter 2** discusses the relevant literature on IoT techniques, sensors, health monitoring devices, and related work in the field. **Chapter 3** presents an overview of the system and the hardware selection process for the project. **Chapter 4** discusses the device's hardware design, including sensor integration, PCB design, enclosure, MCU, power supply, and display components. **Chapter 5** covers the software design, including communication with sensors, calibration, real-time display updates, and the web server software. **Chapter 6** presents the measurements and results obtained from the health monitoring device, assessing the effectiveness and precision of the sensors. **Chapter 7** concludes the report by summarizing the project outcomes and proposing future improvements and potential expansion.

Chapter 2

Literature Review:

This chapter consists of the literature review and related work of the project.

2.1. Definition of IoT devices:

Internet of Things (IoT) devices are network-connected electronic devices, namely sensors, actuators, and appliances, that connect wirelessly to a network for the transmission and processing of data, [14]. These devices consist of sensors and electronic circuits that utilize the Internet for communication and data transmission. The sensor data is acquired by a microcontroller and transmitted over the internet to a server, where it is stored and processed for the user. The essential components that ensure the functionality of an IoT device include software, networks, and sensors, [15]. Figure 2.1 below illustrates the basic block diagram of the architecture of an IoT device.

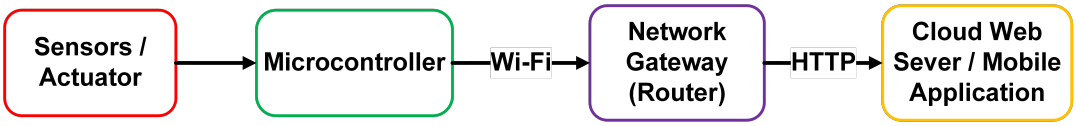


Figure 2.1: Architecture block diagram of a IoT device.

2.2. Description of Architecture block diagram

2.2.1. IoT Sensors

IoT sensors operate by converting physical data into electrical signals. This conversion is made possible through the use of Micro-Electro-Mechanical Systems (MEMS), [16]. Sensors that utilize MEMS technology are miniaturized; compact devices constructed from semiconductor materials. The sensor data is recorded from specific physical conditions, such as ambient temperature or light intensity. Changes in electrical capacitance or resistance prompt the sensor to record the data. The data is converted into an electrical analogue or digital signal. The electrical signal is the primary data source in IoT devices, that provides input data to be controlled and monitored by a microcontroller. Examples of IoT sensors are temperature sensors, humidity sensors, and heart rate sensors.

2.2.2. Microcontroller

A microcontroller unit (MCU) is a compact, self-contained computer integrated onto a single microchip, [1]. The single chip includes a processor, RAM, ROM and peripherals.The processor executes software instructions, the memory stores program and data memory, and the I/O ports enable the MCU to interact with the environment through sensors and devices. Figure 2.2 below illustrates the basic architecture of a Microcontroller Unit (MCU).

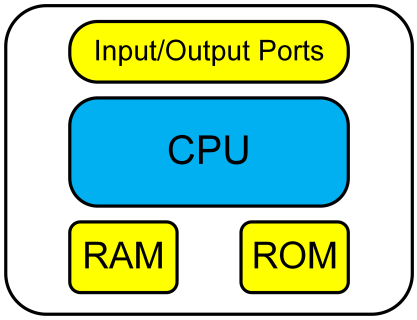


Figure 2.2: Architecture of an MCU, reproduced from [1].

Microcontrollers can be based on the Von Neumann or the Harvard architecture. Within the Von Neumann architecture, the processor communicates with a single data and address bus, to ROM and RAM memory simultaneously, [17]. In contrast, the Harvard architecture uses an address bus and a data bus to interact with the ROM and RAM. The Harvard architecture ensure faster executing of instruction, which is important when selecting the appropriate MCU for a IoT device.

2.2.3. IoT Connectivity

IoT connectivity refers to the network technologies and communication protocols that enable IoT devices to connect and transmit data, [14]. The specific IoT device application, determine the connectivity protocol being used. The most used wireless protocol is Wi-Fi, Bluetooth and Cellular. Wireless protocols such as Wi-Fi enables fast data transmission to web based servers for data processing and management. Wi-Fi ensures real-time data transmission for remote monitoring, which is an essential aspect of any IoT device. Devices and microcontrollers that have Wi-Fi can be configured to either use Access Point Mode or Station Mode, [18]. Most IoT device make use of Station Mode, to enable it to connect to an access point (Router) for internet connectivity. Figure 2.3 below illustrate an MCU configured in station mode, connecting to the access point.

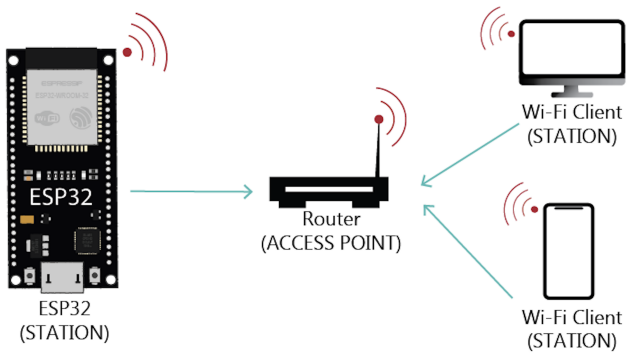


Figure 2.3: MCU Station Mode [2]

2.3. Health Sensors

2.3.1. Temperature Sensors

Temperature sensors are used in all different IoT devices, including medical and health devices. Temperature sensors provide a temperature reading, by using an electrical signal, [19]. The operating principle of temperature sensors involves measuring the voltage across the diode terminals. With a temperature increase, the voltage increase, causing a voltage drop between the base and emitter terminals of a transistor within the diode. Temperature sensors are divided into two categories, namely Contact temperature sensors and Non-contact temperature sensors. Contact temperature sensors make use of direct contact to measure the temperature of an object, [19]. Non-contact temperature sensors measure temperature without direct contact by detecting the radiation emitted from the object.

Comparative Analysis of Temperature Sensor Types

Thermostats

This temperature sensor is a contact sensor that features a bimetallic strip made of two different metals. When the temperature changes, the differing linear coefficients of expansion of the metals cause the strip to bend, producing a mechanical displacement that can be converted into a temperature reading, [19].

Semiconductor-based Sensors

The semiconductor-based sensor contains a dual integrated circuit (IC), with two similar diodes, [19]. These sensors feature a digital interface for effective communication between the sensor and the MCU of the IoT device, [20]. Each diode has voltage and temperature-sensitive current characteristics, allowing for effective measurement of temperature changes.

The output of the sensor is linear, but is less accurate than that of other sensors. The error range for the sensors is between 1 °C - 5 °C. Figure 2.4 illustrates a digital semiconductor-based temperature sensor (LM35) from Texas Instruments, [3].

Figure 2.4: Semiconductor-based temperature sensor, [3].

Infrared (IF) temperature sensor

Every object, that consists of a mass, emits energy in the form of heat. The emitting of heat from an object makes the principle of Infrared (IF) temperature sensors possible, [21]. An IF sensor uses a lens system by focussing IF light onto an object, and measuring the returning IF rays coming off an object, compared to the surrounding environment, [21]. The fundamental principle behind these sensors is Planck's Law. It states that the higher the temperature of an object, the higher the radiation emitted is at each wavelength, [22]. The intensity and wavelength of this radiation are reliant upon the object's temperature. The IF sensor makes use of a thermopile (detector) to measure the temperature difference, [23]. The thermopile converts the temperature difference into a voltage, which is directly proportional to the temperature difference between the object's temperature and the environment. A significant advantage to the sensors is that it's non-contact, which means direct contact is not necessary to measure temperature. Figure 2.5 below illustrates the architecture of an IF sensor.

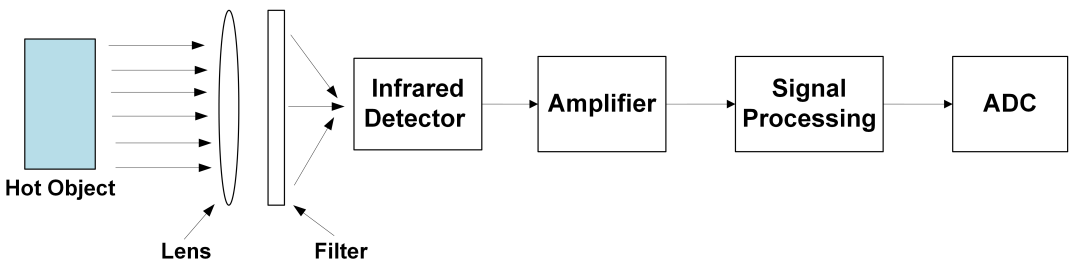


Figure 2.5: Architecture of an IF sensor, reproduced from [4].

2.3.2. Oximeter Sensors

The pulse oximeter has transformed modern medicine by enabling continuous monitoring of the functional oxygen saturation of haemoglobin in arterial blood (SaO_2) [24]. Pulse oximeter sensors implement an optical technique that compares the differences in light absorption spectra of deoxygenated (RHb) and oxygenated (OHb) haemoglobin [5].

The oxygen saturation (SpO_2) estimation relies on photoplethysmography (PPG). Pulse oximetry uses the principle that OHb and RHb absorb red and near-infrared (IR) light differentially, [24].

Principle of Oximeter Sensors

Pulse oximeter sensors relies either on transmission or reflection of light, [5]. Within transmissive pulse oximetry, the light sensors are placed on a patient earlobe or fingertip. The sensor emits light into the skin at two different wavelengths, and a photodetector measures the absorption changes of each wavelength on the opposite side of the skin. In contrast, reflective pulse oximetry can be measured on different parts of the body, instead of the earlobe or fingertip. In reflective oximetry, the photodetector is positioned adjacent to the light-emitting sensor, detecting both reflected and backscattered photons from the emitted wavelengths. Figure 2.6 and figure 2.7 below illustrates the transmissive pulse oximetry method and reflective pulse oximetry method. The light emitting sensor (red and IF light) is indicated as the symbol A, and the photodetector by the symbol B.

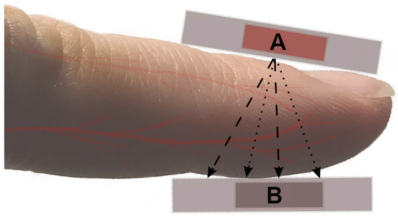


Figure 2.6: Transmissive pulse oximetry, [5]

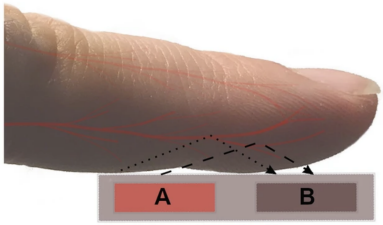


Figure 2.7: Reflective pulse oximetry, [5]

PPG Oximetry Signal Analysis:

noindent The photodetector detects the non-absorbed light from the LEDs, [25] The wavelength of the two LEDs make use of red (660 nm) and infrared (940 nm). The detected signal is processed using an inverted operational amplifier. Figure 2.8 below illustrates the inverted signal from the op-amp. The signal is divided into an AC and DC component and represents the light that was absorbed by the finger, [25]. The DC component consists of the venous blood, tissues, bones, and non-pulsating arterial blood. In contrast, the AC component consists of the pulsating arterial blood.

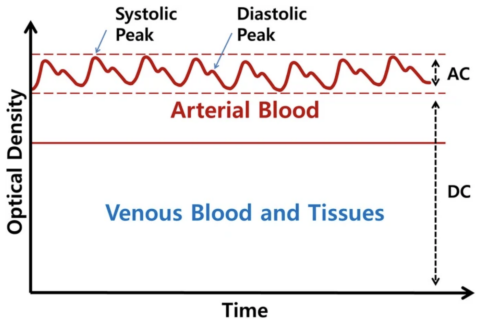


Figure 2.8: Oximetry DC and AC component signal, [6].

One method for the calculation of SpO2 is using the AC component and calculating the ratio (R). Equation 2.1 below illustrates the ratio equation, [25]. The variable I_{ac} represents the AC component of the light intensity. The equation uses both the intensities of red and infrared light.

$$R = \frac{\log_{10}(I_{\text{ac}})_{\text{red}}}{\log_{10}(I_{\text{ac}})_{\text{infrared}}} \tag{2.1}$$

Using the R value, the SpO2 can be calculated using curve approximation and calculate a linear approximation formula, [26]. The equation is derived by using health parameters, namely age, big data and overall health, [26]. Equation 2.2 below illustrates the approximated linear equation for SpO_2. The constants A and B are retrieved after performing curve approximation.

$$\text{SpO}_2 = A - B \times R \tag{2.2}$$

Figure 2.9 below illustrates a block diagram of a pulse oximeter. It illustrates that the LEDs requires a LED driver and the signal from the photodetector requires amplification and filtering before ADC conversion and the calculation of SpO_2.

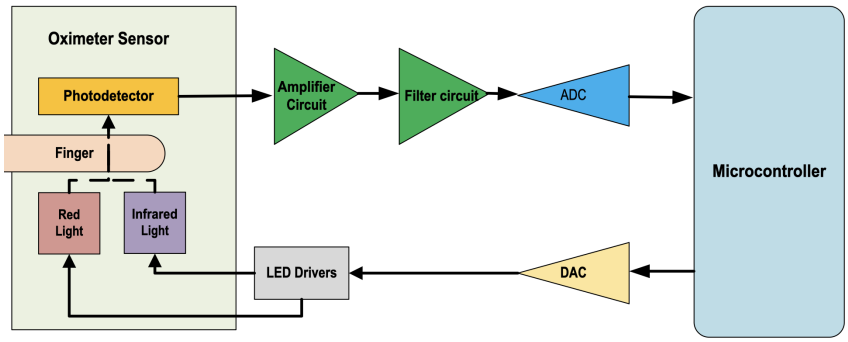


Figure 2.9: Block diagram of pulse oximeter sensor, reproduced from [7].

2.3.3. Heart Rate Sensors

An optical heart rate sensor aims to measure pulse waves by observing changes in blood vessel volume using an LED and a photodetector, [27]. A pulse detection-specific optical filter is included to minimize ambient light interference, allowing the sensor to provide precise readings even in an outdoor environment [27]. Photoplethysmography (PPG) is a cost-effective and non-invasive optical technique that is used to retrieve data from the skin's surface, [28]. PPG is used primarily for heart rate measurement, using a green LED with a photodetector [28]. The method of measuring heart rate is similar to the oximetry measurement discussed in section 2.3.2, except that heart rate measurement requires only one LED with one wavelength. Similarly to oximeter sensors, heart rate sensors that use PPG can operate in transmissive or reflective mode. The principles of both methods are identical, as discussed in Section 2.3.2 for oximeter sensors.

Heart Rate Signal Analysis:

The PPG heart rate signal measurement consists of both DC and AC components. The DC component reflects light signals from tissues, bones, and muscles, while also capturing the average blood volume in both arterial and venous blood [26]. In contrast, the AC component represents the systolic and diastolic phases of the cardiac cycle. By continuously measuring the peak systolic intervals of the AC component, changes in blood volume over time can be observed, [26]. Similar to the oximetry measurements, the AC component is used for the measurement of heart rate. Figure 2.8 above in section 2.3.2 illustrates the DC and AC components of a PPG heart rate signal. The AC component is required for the measurement of the heart rate signal, but the amplitude is much smaller than that of the DC component. Thus, the DC component of the PPG signal must be removed, to analyse the AC component, [29]. The DC component is removed by filtering circuitry and the AC component requires amplification, for the analysis of a heart rate, [29]. The filter circuitry usually consists of an active low-pass filter and a high-pass filter. The low-pass filter consists of the first filter stage and the second filter stage, consists of the high-pass filter. The output of the high-pass filter is fed into a comparator to convert the AC component into digital pulses. Figure 2.10 below illustrates the circuitry required for heart rate measurement.

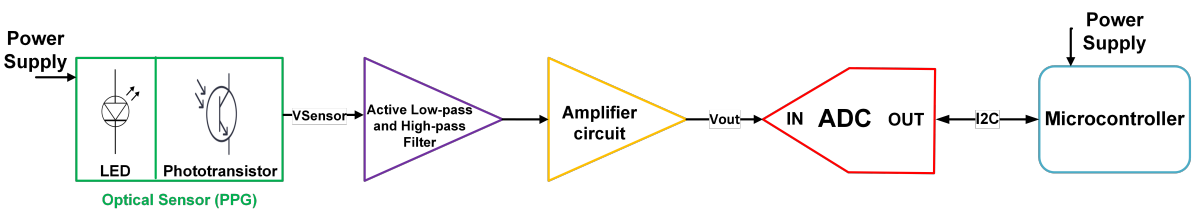


Figure 2.10: Block diagram of heart rate sensor.

The output signal from the amplifier circuit is an amplified signal of the original measured signal, but with only the AC component visible. The comparator or ADC converts the analogue signal to digital pulses. Using these pulses, the heart rate of an individual can be calculated. Figure 2.11 below illustrates the analogue heart rate signal and the digital heart rate pulses. Channel 1 illustrates the measured analogue pulse, and Channel 2 illustrates the digital pulse after the comparator.

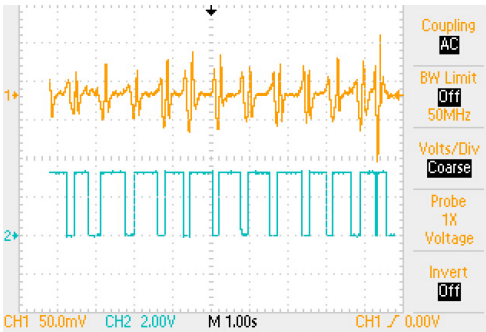


Figure 2.11: Analogue and Digital Heart Rate Signal.

Using the measured digital pulses, the heart rate (BPM) can be calculated from the number of pulses detected over a short interval using equation 2.3 below.

$$\text{Heart Rate (Bpm)} = \frac{\text{Number of Pulses in the Interval}}{\text{Time Interval (in seconds)}} \times 60 \tag{2.3}$$

2.4. Related Work

2.4.1. IoT Health Wearable Monitoring System

Objectives

In the paper, *IoT BASED HEALTH MONITORING SYSTEM*, [30], the project aims to develop a wearable health monitoring system that tracks the user's location, measures heart rate and monitors body temperature in real-time. The device uses technologies within Arduino UNO and ESP8266 NodeMCU to monitor and process health sensor data. Features such as data visualization of health data and phone alerts of sudden abnormal heart rate data and body temperature changes. By integrating IoT technologies into the wearable device, the system aims to create an effective method for health monitoring.

Method

The IoT-based health monitoring system consists of three main layers: the gateway layer, broker layer, and the data visualization and storing layer. The gateway layer includes the LM35 temperature sensor and SEN-11574 pulse sensor connected to an Arduino UNO and ESP8266 NodeMCU gateway. These sensors collect user body temperature and heart rate data, which is transmitted to the broker layer via MQTT protocol. The broker layer, managed by a Raspberry Pi 3 Model B+ running Raspbian Lite, hosts Mosquitto broker for MQTT communication, Node-RED for system programming and interaction with services like IFTTT and Thingsboard, and PHPMyAdmin for local database storage. Real-time health data visualization occurs on Thingsboard, while PHPMyAdmin stores

data for user tracking. Alerts for sudden health changes are managed by IFTTT, notifying users via email and redirecting them to Thingsboard for detailed monitoring.

Results

The results of the wearable health monitoring system were promising. The SEN-11574 pulse sensor demonstrated good performance, providing accurate measurements for both resting and active heart rates. The resting heart rate measurements showed an average of 66.1 beats per minute (BPM) with minor fluctuations around ± 2 BPM, attributed to the pressure applied by the fingertip on the sensor. This measurement is within the expected range of a healthy individual when at rest. For the active heart rates after exercising, the sensor consistently measured an average of 113.6 BPM. These results indicate that the pulse sensor effectively detected heart rate changes with high accuracy. However, the LM35 temperature sensor showed unreliable performance due to fluctuating output values, suggesting the need for a more reliable temperature sensor in future iterations of the system.

Shortcomings

Although the results were promising, the health system relied on multiple external web services to send email alerts and host health data instead of implementing a local web server on the microcontrollers. Furthermore, the health data were only visible on a computer or cell phone screen, rather than on an embedded display on the wearable device. An embedded screen would make the IoT device much more effective by allowing immediate access to health data and graphs directly on the wearable. Furthermore, due to budget constraints, the accuracy of the sensors was not considered and was not compared to established medical devices to determine the accuracy.

Chapter 3

Hardware Selection:

This chapter consists of the hardware selection of the IoT health device.

3.1. System Diagram

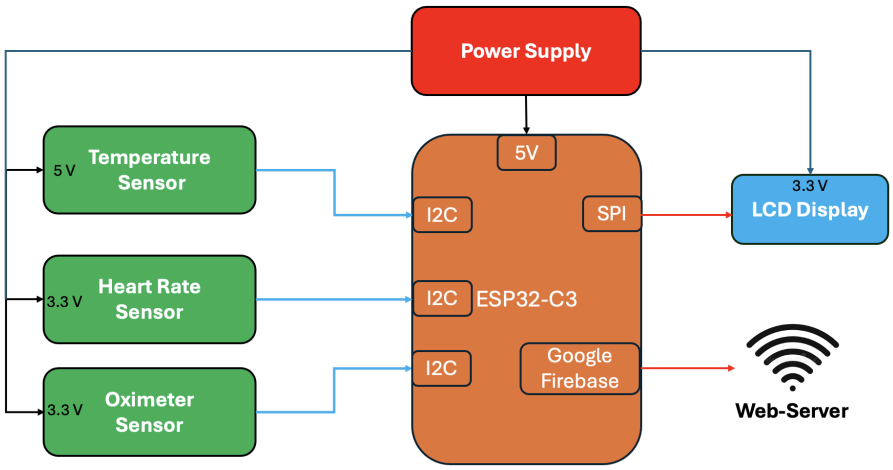


Figure 3.1: Block diagram of system overview.

Figure 3.1 illustrates the block diagram of the IoT health device. The health device is a health monitoring device that integrates multiple sensors and displays real-time and historical data. The ESP32-C3 microcontroller is used to gather data from three main sensors, the temperature sensor, the heart rate, and the oximeter sensor, with all sensors using I2C. The ESP32-C3 communicates over SPI with an LCD to present the data to the user and stores historical data on a Google Firebase. A Web server is incorporated to enable remote access to the user's health data, providing both real-time and historical data for viewing.

3.2. Component Selection

Choosing the most appropriate sensors and hardware components is essential for developing this IoT health device, as performance is highly dependent on the quality of the components. Various microcontrollers and sensor modules were evaluated and compared to ensure accurate results for real-time health monitoring.

3.2.1. Selection of Microcontroller

The selection of a microcontroller is an essential part of every IoT device. Similarly, for this health device, the microcontroller is an essential part of displaying real-time and historical data for users. Four different microcontrollers were evaluated to determine their suitability for this project, and the most optimal option was selected. The following features are most important when deciding the microcontroller for this project. The clock speed, flash memory, cost, wireless connectivity, communication protocols, and power consumption are the most important features when deciding the microcontroller for this project.

- **Clock Speed:** Higher clock speed is important for IoT devices because it allows for faster processing, which is essential when handling multiple sensors and displaying real-time and historical data.
- **Memory:** IoT devices require sufficient SRAM and Flash memory on the microcontroller is essential. This provides the necessary capacity to store health monitoring algorithms, web server scripts, and sensor calibration data, ensuring smooth operation and efficient data processing.
- **Cost:** The cost of the microcontroller must be carefully considered, ensuring effective budget management for the project. The microcontroller should not only meet the project's specifications, but also be cost-effective to ensure it stays within budget.
- **Wireless Connectivity:** Wireless communication is a critical aspect of the project for transmitting real-time and historical data. Since the device hosts a web server, the microcontroller must include Wi-Fi capability to transmit the data to browsers and mobile devices efficiently.
- **Wired Connectivity:** Communication with the sensors is essential, so the microcontroller must have sufficient GPIO pins and support UART, I2C, and SPI connectivity options to ensure that all sensors in the health device can be properly connected.
- **Power Consumption:** The microcontroller should have low power consumption, while connected to multiple sensors. Although the health device isn't battery-powered, a low-power usage microcontroller should be considered, since a battery can always be added later.

Table C.1 within Appendix **??** illustrates the comparison between four different microcontrollers considered for the project. After evaluating the microcontrollers in Table C.1, the **ESP32-C3-DevKitC-02** and **RP Pico** were identified as the best two options for this project due to their low power consumption, sufficient memory, and wide range of

wired connectivity. However, the ESP32-C3-DevKitC-02 was ultimately chosen for several reasons. The built-in Wi-Fi is ideal, for the integration of a web server, which is crucial for real-time and historical health data access. The microcontroller also supports both I2C and SPI interfaces, necessary for connecting the health sensors and an LCD. The ESP32-C3 also supports the highest memory capacity (4 MB Flash & 400 KB SRAM) of all the microcontrollers. This ensures the ESP32-C3 is well suited for handling the program data and storing real-time health data. Lastly, its cost of ±R150 makes it a highly cost-effective option compared to others like the **STM32F103C8T6** and **Arduino Nano 33 IoT**. For these reasons, the **ESP32-C3-DevKitC-02** was selected as the microcontroller for this project.

3.2.2. Selection of Temperature Sensor

The temperature sensor selection is a critical aspect of the IoT health device. Temperature is an important indicator of an individual's health, making accurate measurement essential. Therefore, choosing a precise and cost-effective sensor is crucial to the project's design. The following features were considered in the selection of the temperature sensor for the project.

Accuracy is a key factor when selecting a temperature sensor, as inaccurate readings can result in unreliable health data and compromise the effectiveness of the health monitor. It's also important to consider the sensor's **measurement mechanism** to ensure it suits the application's needs. **Compatibility** between the sensor and the microcontroller (MCU) is essential for seamless integration. Finally, the sensor's **temperature range** must be evaluated to confirm it falls within the typical range of human body temperature.

Table C.2 within Appendix C illustrates the comparison of four different temperature modules considered. From table C.2 above, it is evident that the **D6T-1A-01** and the **DS18B20** are the top temperature modules in terms of accuracy. These sensors will provide the most reliable results for measuring user temperature. The decision between these two modules hinges on the measurement method. The D6T-1A-01 was selected for this project for several key factors. It delivers high accuracy, which is essential for accurately measuring a user's temperature. This sensor operates without contact, utilizing infrared (IF) to measure temperature. In contrast, the DS18B20 requires physical contact, as it is a contact sensor. While the D6T-1A-01 utilizes infrared to measure skin temperature, calibration will be necessary to ensure accurate readings of the core body temperature, compensating for any differences between skin and internal temperature. The non-contact infrared measurement capability makes it ideal for situations where touchless temperature sensing is required and more comfortable for the user. The D6T-1A-01 operates in a 0°C to 60°C range, making it ideal for detecting fevers and abnormal body temperatures. The sensor's I2C interface simplifies integration with the chosen microcontroller and enables

efficient communication with the device. Although the **D6T-1A-01** is more expensive than other options, its accuracy, non-contact functionality, and ease of integration make it the best choice for this project.

3.2.3. Selection of Oxygen and Heart Rate Sensor

The selection of the oxygen and heart rate sensor is a crucial component of the IoT health device. The aim for the project is to find a module that combines both an oximeter and heart rate sensor into a single solution. Oxygen saturation levels and heart rate provide vital information about an individual's health. Choosing a sensor that offers high precision while remaining cost-effective is essential to the performance of the device. The following features below were considered in the selection of a heart rate and oximeter module for the project.

- **Type (Digital vs. Analogue):** The choice between digital and analogue is essential, since the project should use the appropriate sensor type. Digital sensors are preferred for health devices because the output is a digital value, which is efficient for real-time health monitoring. This reduces complexity and simplifies data processing when using multiple sensors in IoT devices. Analogue sensors require more calibration and need an ADC for signal conversion, which can increase the risk of noise and lead to less accurate health data.

- **Transmissive vs. Reflective:** The measurement mechanism is a key factor when selecting a heart rate and oximeter module. The module should be easy to mount, allowing users to take measurements effortlessly. Reflective sensors are commonly used in health devices to ensure proper contact between the sensor and the skin's surface. Reflective mode is more practical for health devices than transmissive mode, as it eliminates the need for a photodetector on the opposite side of the skin.

- **Connectivity (I2C vs. SPI):** The sensor's communication interface is a key factor in selecting the most suitable module for this project. The main types of communication protocols used for heart rate and oximeter sensors are SPI and I2C. I2C only uses SDA and SCL, and multiple sensors can connect to the same bus, which is ideal for IoT devices. SPI is faster than I2C, but requires that more data lines connect to the MCU.

- **Cost:** The cost of a Heart rate and Oximeter module is a significant factor of the project. More expensive modules provide higher accuracy and better performance. Although expensive modules perform better, the best possible option should be considered for the project, since it should be cost-effective.

Table C.3in the appendix illustrates the comparison of four different heart rate and oximeter modules. Based on the comparison in Table C.3, the **SEN-15219** and **MIKROE-4986**

are the two best options for the project. The **MAX86140EVSYS** sensor is very expensive and uses SPI, which requires multiple pins. The **BH1792GLC** is a more affordable option, but it comes in an SMD package, which isn't practical for this health device since the sensor should be mounted on the enclosure. Both the SEN-15219 and MIKROE-4986 have digital outputs and use the reflective measurement mechanism. They both utilize an I2C communication interface, ideal for the ESP32 since it only requires two GPIO pins. The SEN-15219 was selected for the project due to its smaller dimensions (25.4 x 12.7 mm), as the MIKROE-4986 is considerably larger (42.9 x 25.4 mm). The smaller package ensures the device is more compact. Additionally, the cost difference between the two is negligible, with the SEN-15219 priced at approximately R790 and the MIKROE-4986 at approximately R807. Given the similar pricing, the more compact size of the **SEN-15219** makes it the optimal choice, balancing functionality, cost, and form factor for this project.

3.2.4. Selection of Display

A key objective of the health device is to include a high-resolution display that ensures fast real-time visibility of health measurements for the user. The display should be large enough to ensure that all sensor data can be presented. The integration of a display into the device ensures immediate feedback to the user, allowing quick assessment of vital health measurements. The following features were considered in the selection of a display for the health device.

Resolution is a critical factor, as a high resolution display ensures that all health data are presented with clarity and readability, enhancing the overall user experience of the health system. **Display size** is another significant consideration, it should be large enough to present all relevant health data effectively. The decision between a **monochrome** and a **colour** display is also crucial, as each offers distinct benefits. Colour displays provide superior data visualization and an improved aesthetic experience for the user, while monochrome displays are notably more energy efficient. A **high refresh rate** is essential for real-time data updates, ensuring that fluctuating measurements are displayed seamlessly and accurately without delay. For this project, it was decided to consider color displays for best data visualization and to only consider LCD's for better performance. The following four LCDs were considered for the project.

1. Adafruit 3.5" 320x480 Color TFT Touchscreen Breakout - R732
2. Wave 1.8 inch Display, 128x160, SPI Interface - R118
3. 2.8' ILI9341 240x320 SPI TFT LCD Display - R245
4. ESP32-S3 Parallel IPS 4" Display ST7701 - Capacitive Touch - R839.00

The selection process for the display of the health device involved evaluating four options based on size, resolution, interface, and cost. The **Adafruit 3.5" 320x480 Colour TFT Touchscreen** was selected due to its high resolution and affordable price (R732), aligning with the project budget. The **Wave 1.8-inch Display, 128x160, SPI Interface** (R118) was not suitable due to its smaller size and lower resolution, which would compromise the display of health data. The **2.8" ILI9341 240x320 SPI TFT LCD Display** (R245) was also excluded for similar reasons: its lower resolution and smaller screen. Although the **ESP32-S3 Parallel IPS 4" Display ST7701** offers comparable size and advanced features, it was not chosen due to its higher price and the high demand for GPIO pins. Therefore, the Adafruit display was selected for its SPI interface, high resolution, and large screen.

3.2.5. Selection of Storage

The project requires storage for both historical health data and graphical resources (images/icons) for the web server that displays this data. Various storage options were considered, including cloud storage and external storage solutions. Google Firebase was selected for storing historical data due to its ability to provide scalable cloud storage, accessible from any device with an internet connection. Firebase offers seamless integration with the project's web server and ESP32, making it an ideal choice for real-time data storage and retrieval. For graphical resources, GitHub was selected as the cloud storage solution, allowing for easy hosting and management of personalized resources directly from a repository.

3.2.6. Selection of Power Supply

The selection of the power supply for the project was carefully considered, ensuring stable and efficient operation of all components. The main components to be powered are the ESP32-C3, the temperature sensor (D6T-1A-01), the oximeter and heart rate module (SEN-15219), and the display (Adafruit LLC 2050). The device will be powered by a 9V wall power supply, and the voltage will be stepped down to 5V. The LM7805 linear voltage regulator was chosen to step down the input voltage, providing an effective solution for supplying power to the components, as it delivers a stable 5V output. The ESP32 is powered directly by the 5V output from the LM7805, which also supplies power to the temperature sensor (D6T-1A-01), ensuring it operates within its required voltage range. Additionally, the 3.3V regulator integrated within the ESP32 is utilized to power the oximeter and heart rate module (SEN-15219), which require 3.3V for proper operation. This power supply configuration ensures that all components operate within their optimal voltage ranges, providing reliable performance.

Chapter 4

Detail Hardware Design:

This chapter outlines the detailed hardware design of the health monitoring system, based on the hardware components selected in chapter 3.

4.1. Detailed Hardware Design

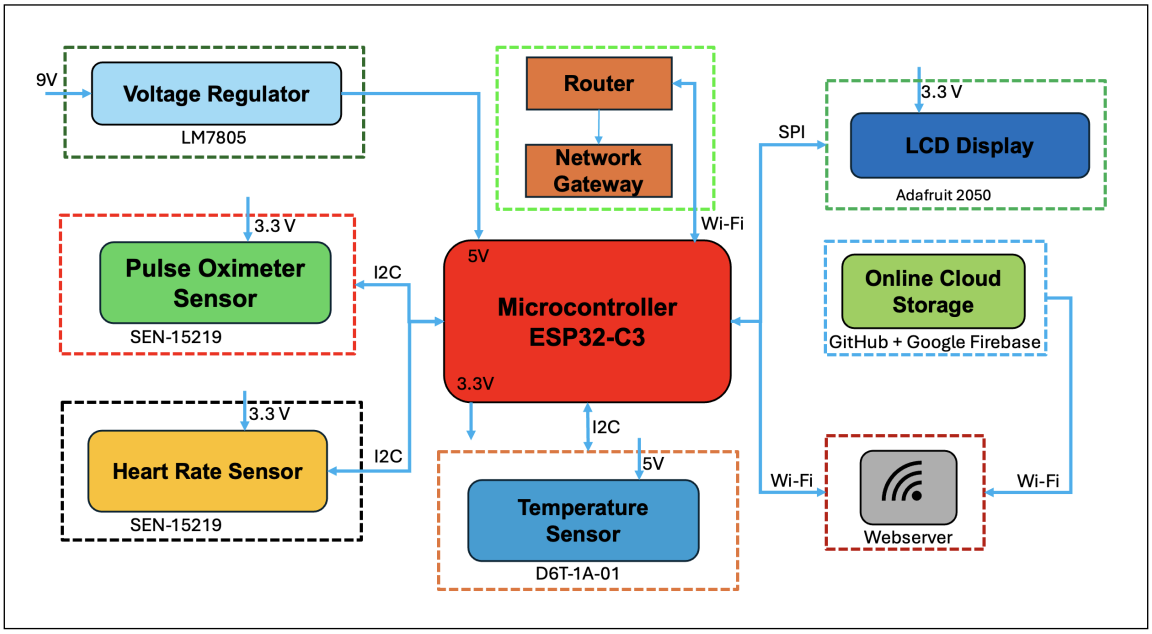


Figure 4.1: Detailed system diagram of IoT health device

Figure 4.1 above illustrates the detailed system diagram of the IoT health device. The ESP32-C3 microcontroller acts as the main processing unit, interfacing with multiple sensors and peripherals through different communication protocols. Power is supplied through a 9V wall power supply regulated by the LM7805 to 5V. The microcontroller and the temperature sensor (D6T-1A-01) both operate on the regulated 5V. The pulse oximeter and heart rate module (SEN-15219) are powered by 3.3V and communicate with the ESP32 via the I2C protocol. The LCD display (Adafruit 2050) are connected via SPI. Wi-Fi connections link the ESP32 to the router and enables internet connectivity, enabling data transfer to the web server and connection to online cloud storage (GitHub and Google Firebase). This design enables effective display of real-time and historical data display on the web server and TFT display.

4.1.1. Power Supply Design

The power supply design is divided into two sections: a 5V power supply and a 3.3V power supply. Figure 4.2 illustrates the linear voltage regulator circuit used to achieve a 5V output. The voltage regulator is powered by an external 9V DC power supply connected to a DC Jack (J1). The input voltage is fed into the LM7805CV (U1) regulator through a rocker switch (SW1) and a 1N4007 diode (D1). The rocker switch allows the health device to be switched on and off, while the 1N4007 diode protects the circuit from reverse polarity, preventing damage if V_{CC} and ground are reversed. The LM7805 regulates the input to a stable 5V DC output. The regulator requires three capacitors to ensure stable and reliable operation, [31]. The input capacitor (C1 - 100 nF) filters out noise and transient voltage spikes from the 9V input, ensuring the regulator receives a stable input voltage. The output capacitor (C2 - 100 nF) reduces high-frequency noise at the output, ensuring a clean and stable voltage supply. The additional output capacitor (C3 - 10 μF) enhances the transient response of the LM7805 by providing immediate charge during sudden load changes, stabilizing the output voltage and preventing voltage spikes, [31]. The second section of the power supply is the 3.3V supply, which is provided by the onboard regulator of the ESP32-C3. This 3.3V output is essential for powering the heart rate and oximeter module and the display. A 10 μF decoupling capacitor is placed on the 3.3V line of the ESP32 to reduce noise and stabilize the supply.

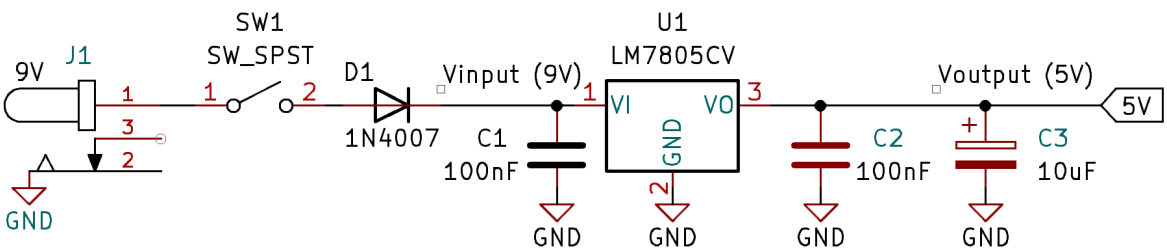


Figure 4.2: LM7805CV Voltage Regulator Circuit

4.1.2. Microcontroller

The ESP32-C3-DevKitC-02 was selected as the MCU for the health device in section 3.2.1. It is the core component of the health monitoring device, for processing, communication, and capturing of the sensor data. The ESP32-C3-DevKitC-02 features a 32-bit RISC-V architecture with a maximum clock speed of 160 MHz, 400 KB SRAM, and 4 MB flash, ensuring it is powerful enough to handle multiple sensor inputs required for the project. It features wired interfaces such as UART, I2C, SPI, and USB, which is essential for both the heart rate and oximeter module, temperature sensor, and the display. The MCU operates on 3.3V and provides both Wi-Fi and BLE 5.0 as connectivity options. These features ensure that all input peripherals can communicate with the MCU and that data can be processed to the web server and display. Figure 4.3 below illustrates the ESP32-C3

development board, [8].

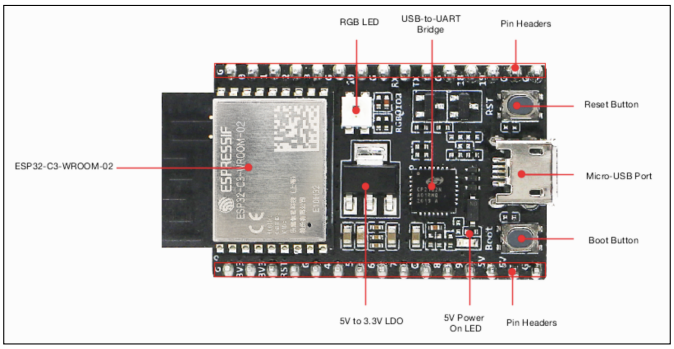


Figure 4.3: ESP32-C3 development board [8]

Power Supply for ESP32-C3

The ESP32-C3 is powered by an external linear regulator providing 5V, as detailed in section 4.1.1. The output of the LM7805CV is connected to a dedicated 5V pin on the ESP32-C3, with the regulator, MCU, and associated components sharing a common ground. The ESP32-C3 incorporates a low dropout (LDO) regulator that steps down the 5V input to a stable 3.3V. The MCU features two 3.3V pins on the development board, which are critical for supplying power to the sensors within the monitoring device. The temperature sensor (D6T-1A-01) is connected to the second 5V pin on the ESP32-C3, while the temperature and oximeter module (SEN-15219) and the display (Adafruit LLC 2050) are connected to a 3.3V pin on the MCU.

I2C Connections

I2C is a critical communication interface for the project, as it enables communication with the two health sensors. The temperature sensor (D6T-1A-01) and the heart rate and oximeter sensor (SEN-15219) both utilize I2C, requiring SDA and SCL lines to be connected to the microcontroller. The default I2C pins of the ESP32-C3 were reconfigured, with **GPIO pin 5** assigned for SDA and **GPIO pin 6** for SCL. Since I2C operates as a single bus, the SDA and SCL lines from each sensor are connected to the same GPIO pins on the microcontroller. According to the datasheet, [32], the SEN-15219 requires two additional data lines, a Reset (RST) line and an MFIO line, which also need to be connected to the ESP32. For reliable I2C communication, **pull-up resistors** are essential on the SDA and SCL lines to ensure proper logic levels and signal integrity. Thus, on both the SDA and SCL lines, a resistor should be placed from V_{CC} to each line. For the SEN-15219 an extra two data line is required, according to the datasheet [32]. These pins are a Reset (RST) line and an MFIO line.

SPI Connections

The ESP32 supports three SPI buses that can be configured for communication with peripheral devices. SPI (Serial Peripheral Interface) serves as the primary interface for the

Adafruit LLC 250 display. The display is configured to operate on a single SPI bus. The display requires five pins for operation. This setup leverages the ESP32's SPI capabilities, providing fast and reliable communication for effective graphical display. Figure 4.4 below illustrates the block diagram of the SPI connections, illustrating the connection between the display and the MCU. Figure 4.5 below illustrates the schematic of the connections to the ESP32-C3 for the project.

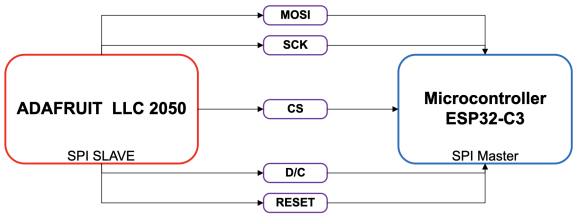


Figure 4.4: Block diagram of SPI connections

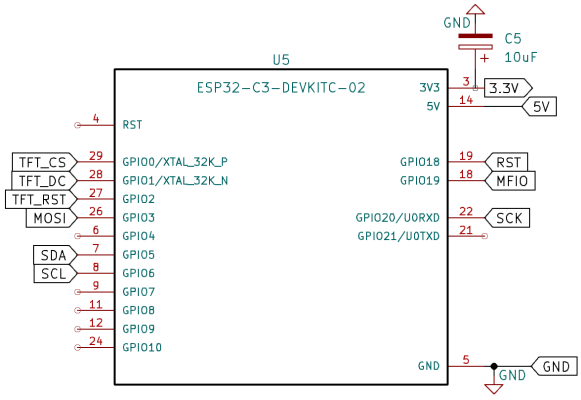


Figure 4.5: Schematic with connections of ESP32-C3

4.1.3. Heart rate and oximter module (SEN-15219)

The SEN-15219 module, produced by Sparkfun, is a heart rate and oximeter module. It was chosen in section 3.2.3, and it is integrated into the health monitoring system to measure heart rate and blood oxygen saturation (SPO_2) using the PPG. The sensor utilizes the reflective mode and measures heart rate and SPO_2 by measuring the changes in reflected light. As mentioned in section 4.1.2, this sensor connects to the ESP32-C3 for calibration and data capture. The module interfaces with the microcontroller via the I2C interface, using the SDA and SCL lines, which are connected to GPIO pins 5 and 6. The ESP32-C3 provides the 3.3V power supply required for the sensor, to operate. Additionally, the module requires two extra data lines to operate, namely the RST and MFIO pin, [32]. The pins are essential because it determines if the module enters data collection, [32]. To ensure reliable I2C communication, the SDA and SCL lines require appropriate pull-up resistors. The SEN-15219 includes internal 4.7 kΩ pull-up resistors, thus external resistors are not required. This helps maintain correct signal levels on the I2C bus, ensuring stable data transfer and safe operation for the module. Figure 4.6 below illustrates the circuit schematic for the SEN-15219, indicating the connections to power and the ESP32-C3.

4.1.4. Temperature Sensor (D6T-1A-01)

The D6T-1A-01 temperature sensor is integrated into the system to provide accurate and non-contact measurements. The sensor utilizes IF technology to measure the temperature

on the surface of an object. This contactless temperature measurement is ideal for touchless contact with users. The D6T-1A-01 communicates with the MCU through the I2C bus, using the SDA and SCL lines connected to the GPIO pins 5 and 6. The operation voltage of the sensor is 5V and is thus powered by the LM7805CV regulator. The sensor does not require additional pull-up resistors because the I2C bus is shared, and the SEN-15219 module already includes 4.7 kΩ pull-up resistors. The shared pull-up resistors ensure a stable signal and keep within the recommended operating voltage limits. The SEN-51219 operates at a 3.3V reference voltage, and the D6T-1A-01 can operate between 3V and 5V, therefore the lower reference voltage was selected. The sensor requires minimal connections to the ESP32-C3, making it easier to interface with the MCU. Figure 4.7 below illustrates the circuit diagram of the D6T-1A-01, indicating the connections to the I2C bus and power

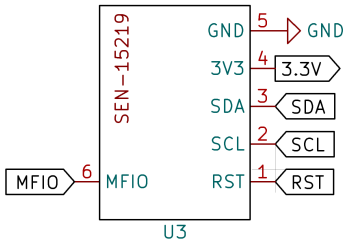


Figure 4.6: Circuit schematic of SEN-15219

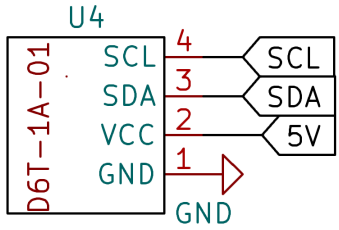


Figure 4.7: Circuit schematic of D6T-1A-01

4.1.5. Display (Adafruit LLC 2050)

The chosen display for the project is the Adafruit LLC 2050, with a 3.5” 320x480 colour TFT display. The display can operate in parallel and SPI mode. For this project, it was decided to use the SPI mode, since it works the best for the design. The display requires fewer connections to the MCU, with SPI mode and with limited pins on the ESP32-C3, making SPI more suitable. In contrast, parallel mode requires more pins to enable the display to work, and thus it is not a viable option for this design, since the ESP32-C3 will not have enough pins. The display in SPI mode requires 5 pins to function. Table 4.1 below details the pin connections between the ESP32-C3 and the display.

Function	ESP32 Pin	Description
Chip Select (CS)	GPIO 0	Configures the TFT display for communication via SPI.
Data or command selector pin (D/C)	GPIO 1	Controls data or command mode for the display
Reset (RST)	GPIO 2	Resets the display
Master Out Slave In (MOSI)	GPIO 3	Sends data from ESP32-C3 to the display
Serial Clock (SCK)	GPIO 20	Provides Clock Signal for SPI

Table 4.1: SPI Pin Configuration for the Adafruit LLC 2050 Display

The display requires a V_{CC} of 3-5V, thus the 3.3V supply of the on-board regulator would be sufficient for the display. This ensures stable operation of the display within the overall system. Figure 4.8 below illustrates the circuit schematic of the display and the connections to the MCU.

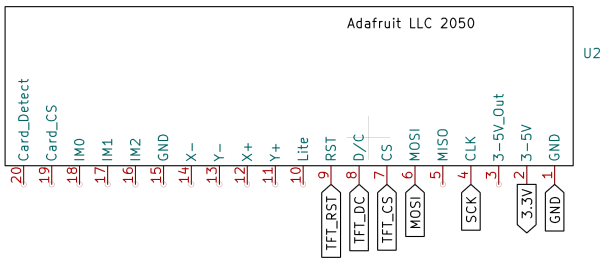


Figure 4.8: Circuit schematic of Adafruit LLC 2050

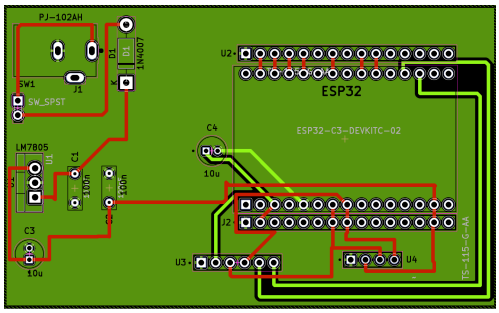
4.1.6. PCB Design

The full circuit of the health monitoring device was first soldered onto a prototype board to verify if the hardware design works. After verification, a PCB was designed within the software KiCad. The PCB for the system was designed as a two-layer PCB to effectively manage the routing of all signals and power supply lines. The design uses top and bottom layers, with the bottom layer consisting of a ground plane, to ensure the grounding of all components in the circuit. The use of the ground plane was incorporated since it reduces interference from the components and reduces the overall noise of the system. Table 4.2 below illustrates the specifications of the PCB. The PCB design incorporates all hardware into a single board that functions to achieve an accurate working device. Figure 4.9a and 4.9b below illustrate the top and bottom layer PCB and the 3D model of the circuit.

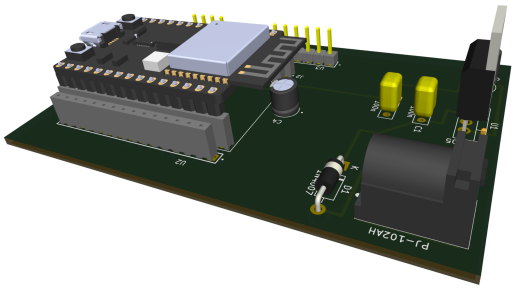
Parameter	**Specification**
PCB Type	2-layer board (Top and Bottom)
Material	FR-4
Track Width	25 mil (0.635 mm)
Track Clearance	15 mil (0.381 mm)
PCB Dimensions	86.5 mm x 53 mm
Ground Plane	Bottom layer

Table 4.2: PCB Track Design Specifications

Figure 4.9a and 4.9b illustrate the top and bottom layer PCB and the 3D model of the circuit. The PCB features headers, to make it simple for the sensors and display to be connected via Du Pont connectors. This design choice ensures the display and sensors are mounted on the enclosure, for easy access for the user.



(a) Top & Bottom PCB

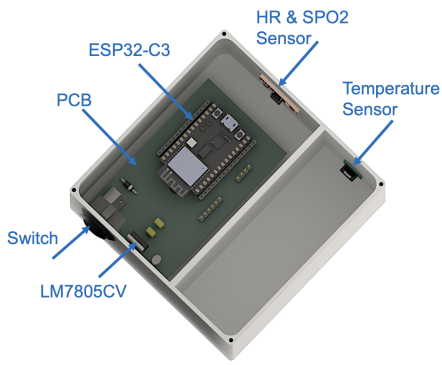


(b) 3D model of PCB

Figure 4.9: PCB design of Health System

4.1.7. Enclosure Design

It is crucial that the health device is housed in an enclosure to protect the sensors and circuitry. The sensors and display also require an enclosure, since it should be mounted on the enclosure for accessibility for the user, during measurement. The enclosure was designed using **Autodesk Inventor Professional** and 3D printed to house the PCB, sensors, and display. The enclosure measures 110 mm x 110 mm x 80 mm, which is sufficient to house the PCB since it is only 86.5 mm x 53 mm. The design of the enclosure is compact but allows enough room for all the sensors and the display to be mounted on the lid. The lid features a display cutout for the display to slot into that ensures that only the visible screen can be seen on the outside of the enclosure. The enclosure allows the temperature sensor (D6T-1A-01) and the heart rate and oximeter module (SEN-15219) to be mounted on the side for reachability for the user. The material used to 3D print is ABS grey, which is a suitable material for mounting circuitry and IoT devices. Figure 4.10a and 4.10b below illustrate the enclosure and the lid of the health monitoring device.



(a) Enclosure for device with mounted components

(b) Lid of enclosure for mounting of display

4.1.8. Physical Build

Appendices G and I illustrate the physical build, with the PCB and enclosure included.

Chapter 5

Detail Software Design:

This chapter outlines the detailed software design of the MCU, web server and external display.

5.1. Software Overview

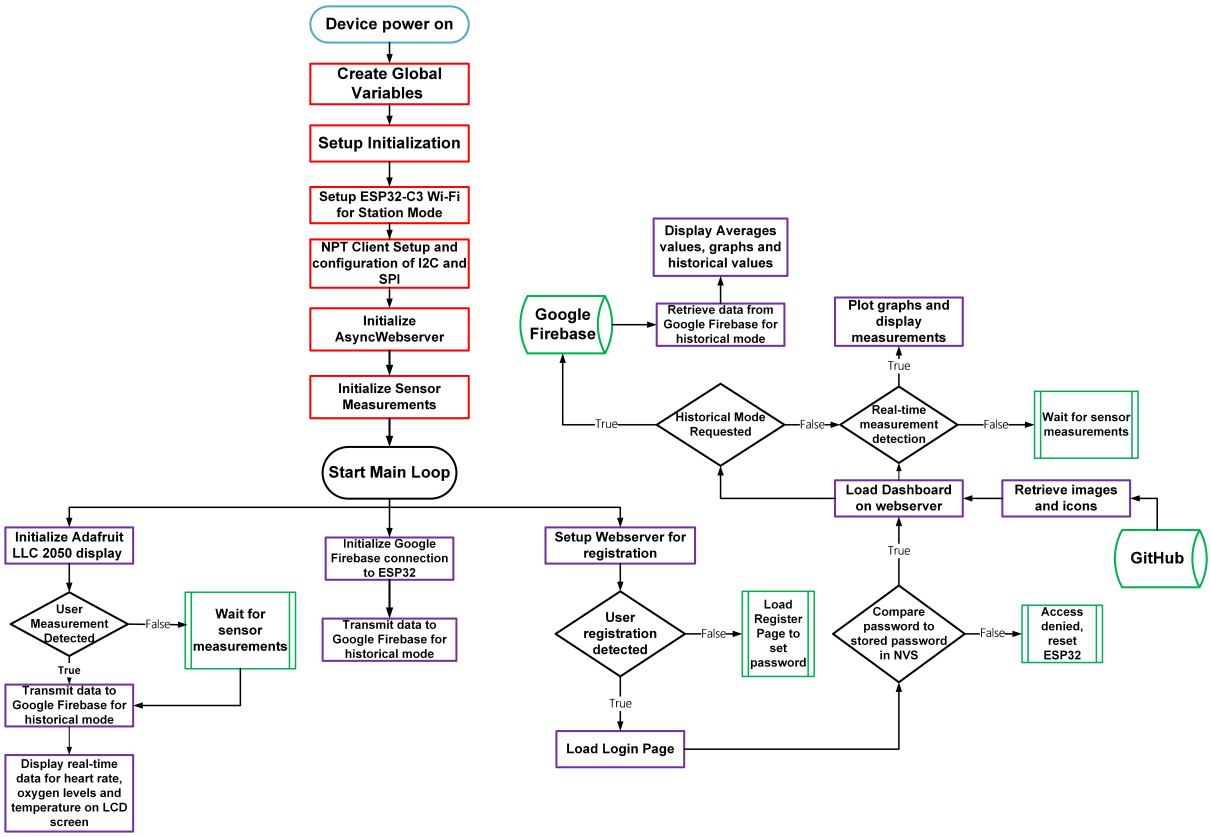


Figure 5.1: Flow diagram of software overview diagram

Figure 5.1 presents an overview of the software architecture for the IoT health monitoring system, outlining its functional structure and workflow. The system starts by powering the device, followed by the setup and configuration of global variables. Subsequently, the ESP32 establishes a Wi-Fi connection by configuring the MCU in station mode (STA mode). The system then initiates NTP time synchronization and configures the I2C and SPI communication protocols. After initializing the web server and sensors for temperature, heart rate, and oximeter readings, the system transitions into its main

operational loop. The main loop is divided into three primary sections: transmitting measurement data to Google Firebase, initializing the LCD (Adafruit LLC 2050), and displaying the data. The final section configures the web server, which offers historical and real-time modes. In historical mode, previous measurements are retrieved from Google Firebase, enabling users to view long-term data and averages of key health metrics, such as heart rate, oxygen levels, and temperature. The web server features separate pages for real-time data display and historical mode. The icons and images used in the web server are stored in a GitHub repository to enhance the visual graphics. In real-time mode, live sensor measurements are displayed for each parameter. The dashboard offers a comparison of heart rate, oxygen levels, and temperature measurements in real time. The detailed view of the real-time mode presents each measurement with graphs for improved clarity and user experience. In historical mode, previous measurements are retrieved from Google Firebase, with three distinct pages displaying the data, accompanied by graphs for each sensor. This allows users to observe long-term trends and averages.

A secure registration and login process is implemented and managed through the web server. User credentials are stored in the ESP32's Non-Volatile Storage (NVS), and access to the dashboard is granted based on these credentials. In the event of incorrect login details, the system resets the ESP32 and denies access. Additionally, the system incorporates an error-handling mechanism to manage alerts when sensor measurements fall outside established normal health ranges. This setup enables users to view health metrics on the LCD while the web server provides more detailed information about their health. The integration between the web server and the display ensures that the software offers a comprehensive solution for continuous health monitoring.

5.2. Sensor Initialization and Communication

The system incorporates multiple sensors, including a heart rate and oximeter module (SEN-15219) and a temperature sensor (D6T-1A-01). The sensors require software for initialization, data capture, and communication, all of which are essential for the project's functionality. The decision was made to use the open-source Arduino IDE to interface with the display, sensors, and MCU. Most of the software is written in C++, except the web server, which is developed using HTML, JavaScript, and CSS.

5.2.1. I2C Communication Setup

The communication between the ESP32-C3 and the sensors is implemented using the I2C protocol. The sensors make use of one I2C bus and are configured within the **setup()** function of the program. The initialization of the I2C protocol requires the **Wire.h** library included in the program. The I2C communicates with the sensors using the serial data line (SDA) and serial clock line (SCL), and for this project, the SDA was assigned to

GPIO 5 and the SCL to GPIO 6. This bus is initialized by the **Wire.begin(SDA, SCL)** command, and the initialization is performed immediately on start-up, to ensure data capture. The I2C bus does not require internal pull-up resistors in software, since the SEN-15219 module already has a 4.7 kΩ resistor, which ensures that the SDA and SCL lines are pulled high.

5.2.2. Sensor Software Initialization

Heart rate and Oximeter module (SEN-15219)

The heart rate and oximeter module (SEN-15219) is initialized within the **setup()** function of the program, and the **SparkFun Bio Sensor Hub** library, provided by SparkFun, is used for the initialization and communication with the sensor. The library simplifies communication by providing functions for both data setup and retrieval. After initializing the sensor using the **bioHub.begin()** function, this function was configured to set the sensor to continuous measurement mode to ensure constant measurement of heart rate and SPO_2. The library makes use of a structured **bioData** object, which stores the values of each sensor measurement. The heart rate and SPO_2 variables are being processed within the main loop to constantly update, and displayed on the Adafruit display and the web server.

Temperature Sensor (D6T-1A-01)

The D6T-1A-01 sensor operates using the I2C communication protocol. The sensor was initialized using the example code provided by the manufacturer, Omron [33]. The address of the device is 0x55 and the I2C address for communication is 0x0A. The software communicates with the sensor by issuing commands to retrieve temperature data. The temperature readings are retrieved in the `loop()` function through the following process:

- The command to start a temperature reading is sent to the sensor using `Wire.beginTransmission(D6T_ADDR)`, followed by `Wire.write(D6T_CMD)` to initiate communication.
- The sensor responds with raw temperature data in a buffer, which is read using the `Wire.requestFrom()` function.
- The raw data is processed by converting it to Celsius.
- The processed data is stored in the variable `temp_IF_sensor`, which is then updated and displayed on both the web server dashboard and the display.

Figure 5.2 below illustrates the software flow diagram of temperature acquisition and processing.

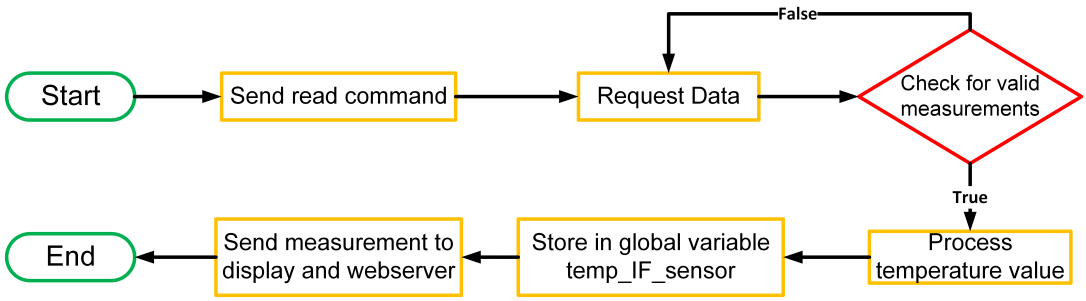


Figure 5.2: Flow diagram of temperature acquisition and processing

5.2.3. Calibration of the temperature sensor

Calibration of the D6T-1A-01 temperature sensor is necessary because it is an infrared (IR) sensor, meaning that the user does not have direct contact with it. This sensor measures skin temperature, which is typically lower than the core body temperature. Software calibration is used to accurately estimate body temperature from the sensor's skin temperature readings. The calibration process involved measuring skin temperature with the sensor and comparing it with body temperature obtained from a medical contact thermometer. This procedure was repeated at different times and under varying temperature conditions to ensure precision. The data is used to calculate the temperature offset between body temperature and skin temperature. This offset is used within software to scale the skin temperature to an approximate body temperature during a user measurement. Equation 5.1 below illustrates the equation used to calculate the approximate offset.

$$\text{Approximate Offset} = \frac{1}{n}\sum(\text{Body Temp} - \text{Skin Temp}) \tag{5.1}$$

Figure 5.3 below shows a graph generated by Python that illustrates the relationship between body temperature and skin temperature. Applying equation 5.1 above to the data points retrieved in figure 5.3, the achieved offset results in 3.34 °C. This offset is then accounted for within software, to allow the temperature measurement to approximate the core body temperature.

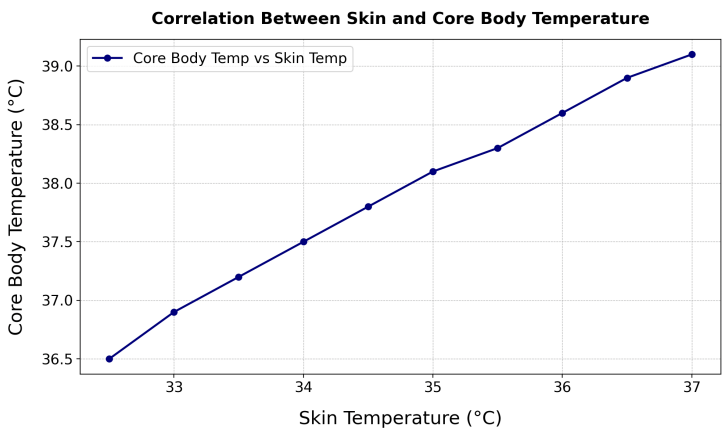


Figure 5.3: Body Temperature vs Skin Temperature plot

5.3. Web Server Setup:

The web server for the system was implemented using the built-in Wi-Fi capabilities of ESP32-C3 to host a local server for real-time and historical monitoring. The MCU connects to a router in station mode, allowing it to connect to the Wi-Fi network. After connection, the web server can be accessed on any device connected to the same router. The server operated on port 80, which is the default port for the ESP32-C3, for HTTP communication. The **AsyncWebServer library** on the MCU was used because it handles multiple simultaneous connections. The web server hosts multiple pages, including a dashboard for real-time data display, a detailed view with interactive graphs of the real-time data, and a historical mode that presents previous measurement data. The software was written using three languages, namely HTML, JavaScript, and CSS, to create an interactive and responsive web server. Password protection for the web server is essential to ensure the security of health data. When a user accesses the system for the first time, a registration page prompts them to set a password. This password is stored in the ESP32's non-volatile storage (NVS), allowing it to be retained even after the device powers off. This approach enhances the security of both the web server and the user's health data. Historical data is securely stored in cloud storage (Google Firebase) and retrieved when historical mode is requested. The data is then transferred to the web server for display. Figure 5.4 below illustrates the architecture diagram, showing the interactions between all components of the ESP32 Web server.

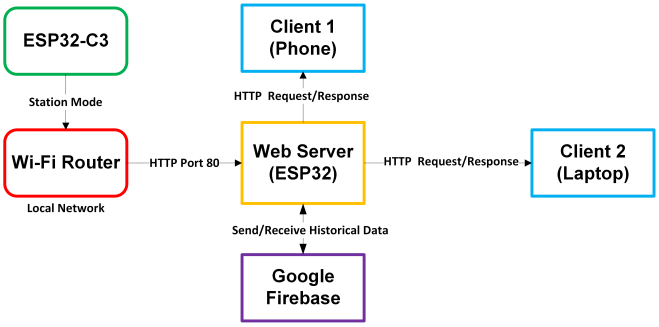


Figure 5.4: System Architecture Diagram of the ESP32 Web server

5.4. Graphical Interface of Web Server:

The Web server consists of six distinct pages that make up the overall system. The main pages include the registration, login, dashboard, detailed view, health information, and historical information pages. Each web server page is styled with **CSS**, and the structure and functionality are coded in **HTML** and **JavaScript**.

5.4.1. User Registration and Login Pages

The registration and login pages provided user authentication, to ensure that health data are secure. The pages make use of form fields that ensure simple use for users. The web pages are styled using CSS, and the blue and grey colour scheme enhances the aesthetic while maintaining a professional look. CSS also ensures proper screen adjustment on various devices and screen sizes.

5.4.2. Dashboard Page

The Web server dashboard page serves as an overview of health data. The page displays live data during the measurement for the user's convenience and compares the measurements using a histogram graph. The **Chart.js** JavaScript library is used to visualize the data in the graph. The measurements are displayed in a large container to ensure visibility. The dashboard dynamically updates using JavaScript, fetching the latest data from ESP32-C3 every second to provide real-time monitoring. The page is styled using **CSS** with a toggle button that allows users to switch between dark and light modes.

5.4.3. CSS Styling and Responsiveness

The overall design of the Web server is consistent across all pages using CSS, to ensure the same fonts, colour schemes, buttons, and layout of the web server. CSS transitions are implemented for a smooth experience. The software for the web server adjusts to be responsive on all devices and to be scaled to the correct screen resolution.

5.4.4. JavaScript and Chart.js Integration

JavaScript is essential in the real-time data retrieval and updating of each page of the webpage. The detailed view of the web server features health alerts for the user, using JavaScript functions. Additionally, integration of the **Chart.js** library enables dynamic graphs that improve user experience. The range of graphs changes to be most suitable for the current web page. Line charts, Histograms, Radar charts, and Pie charts are used for data visualization. Figure 5.5 below illustrates the layout of the web page for the health monitoring web server.

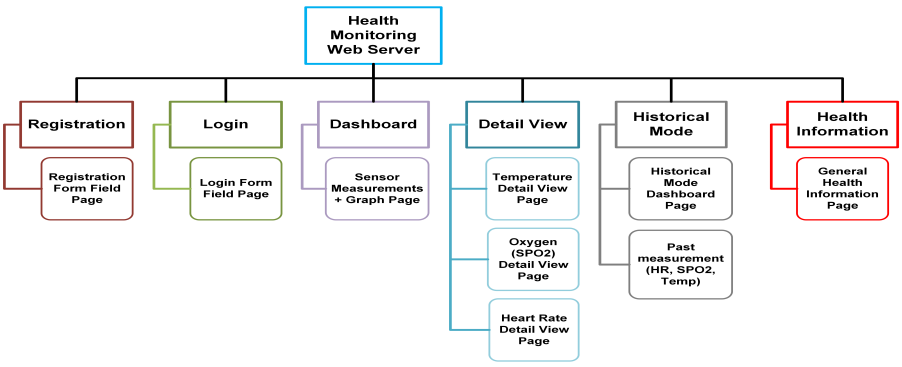


Figure 5.5: Webpage layout for the health monitoring web server

5.5. Storage Management for Historical Mode

5.5.1. Real-time vs. Historical Data

Data from the temperature, heart rate, and oximeter sensors are collected in real-time for the system. Although real-time data are displayed directly on the web server within the dashboard, historical data require effective storage management to ensure long-term tracking and retrieval of health data.

5.5.2. Google Firebase for data storage

Google Firebase serves as the primary storage solution for historical health data and its retrieval for display on the web server. Firebase provides a NoSQL cloud-based database that integrates seamlessly with ESP32-C3. Sensor data is regularly sent to Firebase, where each data entry is assigned a unique ID. The data can later be accessed through the Web server's historical mode, allowing users to view important health trends, averages, and past measurements.

Data Handling Strategy

The ESP32 uses built-in Wi-Fi to transmit valid measurement data to **Google Firebase** at regular intervals, optimizing storage usage and preventing data overload. As shown in the data flow diagram below, the ESP32 initializes a connection to Firebase, authenticates, and formats the data in JSON with a unique identifier before pushing it to Firebase. Data retrieval is handled by two functions, each serving a different page on the web server. Both functions retrieve data in **JSON format** from Firebase, using the **QueryFilter** object to filter and limit results. One function retrieves and displays the averages of historical measurements, while the other retrieves individual past measurements for detailed analysis. Figure 5.6 illustrates the interaction between ESP32, Google Firebase, and the web server, capturing the complete data logging, processing, and retrieval workflow.

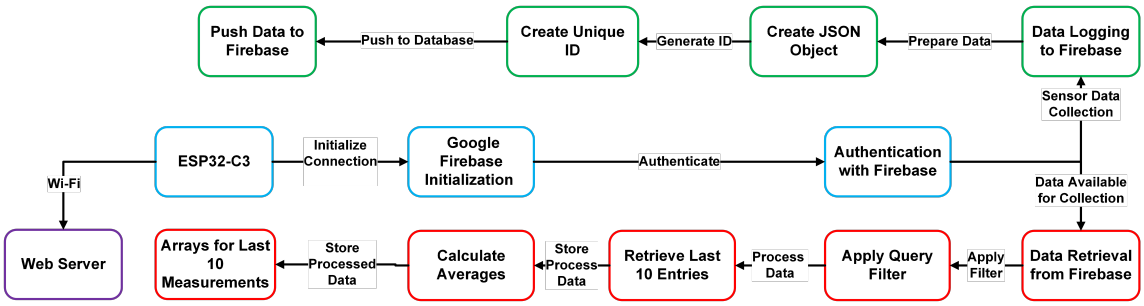


Figure 5.6: Data Management Architecture for the health monitoring device

5.6. Error Handling and Alerts

The health monitoring system incorporates error-handling functions to ensure the validity of the measured sensor data. Error handling is crucial to maintaining proper system

functionality during unexpected events, such as network disconnections, incorrect data capture, and sensor malfunctions.

5.6.1. Sensor Data Validation

Each health sensor is constantly monitored for potential errors, to ensure fast reacting mechanisms. The error mechanisms are as follows:

- **Invalid sensor data** occurs when the sensor returns a zero value or the measured data is outside the expected range. In these cases, a global variable is set to indicate an error.
- **Missing data** occurs when the system fails to retrieve data from the sensor. The system enables a retry mechanism, attempting to reconnect to the sensor or re-read the data for correct data retrieval.

5.6.2. Alerts for Abnormal Measurements

The software is designed to alert individuals about abnormal health measurements. The software is written to react to abnormal measurements and alert users when measurements fall outside normal ranges. Table 5.1 below illustrates the normal ranges and warning thresholds for alerts for the health monitoring system.

Measurement	Normal Range	Warning Thresholds
Heart Rate	$50 \leq \text{BPM} \leq 100$	$\text{BPM} < 50$ or $\text{BPM} > 100$
Temperature	$36°\text{C} \leq T \leq 37.5°\text{C}$	$T < 36°\text{C}$ or $T > 37.5°\text{C}$
Oxygen Levels	$95\% \leq \text{SpO}_2 \leq 100\%$	$\text{SpO}_2 < 95\%$

Table 5.1: Normal Ranges and Warning Thresholds for Health Measurements

Both the web server and LCD issue alerts when measurements fall outside the normal range. The web server includes a notification bell that alerts users to abnormal readings, while the LCD uses colour to indicate measurement status. Green text on the LCD illustrates normal measurements, while red text indicates abnormal measurements.

5.7. Adafruit LLC 2050 Display Software

The system implements a TFT display (Adafruit LLC 2050) to provide real-time health measurements of the heart rate, oximeter, and temperature to the user. The display allows the user to assess the measurements immediately. The display features a 3.5-inch TFT LCD, and this ensures custom visuals and text for the display.

5.7.1. SPI Setup

The display can operate both in parallel mode and SPI mode, as mentioned in section 4.1.5. The decision was made to interface with the display using SPI since it uses fewer GPIO pins on the MCU. SPI is a synchronous protocol that allows the MCU to send commands

and data to the display efficiently. In this configuration, the display serves as the SPI slave, and the ESP32-C3 acts as the SPI master. The library used for the display is the **Adafruit_HX8357 Library**, provided by Adafruit. The SPI communication is initialized within this library. The following pins are required for SPI initialization: MOSI, SCK, CS, DC, and RESET. The SPI clock frequency is set to 24 MHz, ensuring fast and reliable data transmission for health data. Although a higher frequency could be configured, it would slow down the web server operations and sensor data retrieval, leading to potential lag. This library helps with the process of setting pixels, drawing texts, and rendering any graphical elements.

5.7.2. Software Structure for Display

The LCD illustrates live real-time data from the sensors for easy readability for the user. The screen is updated frequently to avoid lag, flickering, and any interruption in real-time data display. The software follows the following process:

1. **Fetching Sensor Data:** The system retrieves the health data from each global variable. The data is processed and formatted for the LCD.

2. **Displaying Data:** The cursor position for each measurement is set for the display, and the `tft.print()` function is used to render the data onto the screen.

3. **Updating the Display:** The display is refreshed every second to show the latest value, within the main loop of the program. This ensures live data, without screen flickering.

Figure 5.7 below illustrates how a cursor is set for the display using the **tft.setCursor(x, y)** function to write text to the display.

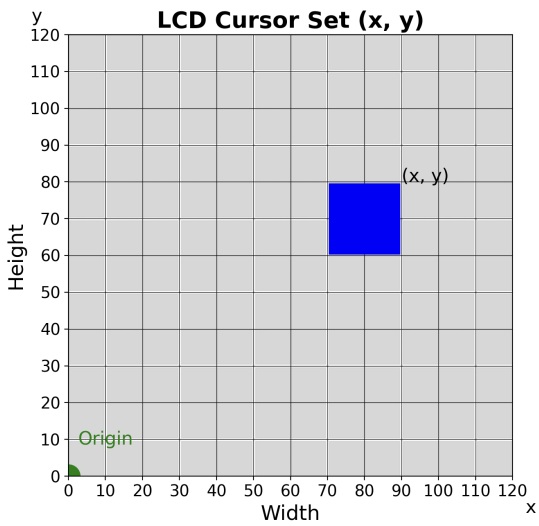


Figure 5.7: LCD Cursor illustration

Chapter 6

Measurements and Results:

This section presents the measurements taken from the IoT health monitoring device to evaluate the precision and effectiveness of the integrated hardware, software, and IoT connectivity.

6.1. Method of Measurement

The measurement process involves capturing real-time data from the health sensors embedded in the IoT device. Measurements were retrieved using a combination of data from the oscilloscope, Google Firebase, LCD, and web server screen. Sensor measurements uploaded to Google Firebase for historical data mode were retrieved to assess heart rate, oxygen levels (SpO2), and temperature reading accuracy. The data were verified by generating Python plots and comparing them with typical resting and active health measurements in humans. In addition, the data were compared with medical devices to evaluate the effectiveness of the calibration. The oscilloscope was used to verify the voltage stability of the power supply and the amount of noise on the power rail. It also played a crucial role in checking the sensor data lines and measuring any noise present, ensuring the accuracy of the sensor measurements.

6.2. Oscilloscope Measurements

This section discusses the measurements taken using an oscilloscope and a probe, to observe the power rails and communication lines (I2C). Key parameters, such as voltage levels and noise levels on both the power rails and I2C lines, were captured to evaluate the performance of the system.

6.2.1. Power Supply

Figure 6.1 below illustrates the 5V regulator supply rail. Channel 1 in Figure 6.1 shows that the power rail measures 5.04V, which is close to the expected output voltage. Channel 2 displays the noise present on the power rail, with an V_{Max} of 26 mV. This noise level is acceptable for the system, as all connected sensors operate digitally and are less sensitive to minor voltage fluctuations.

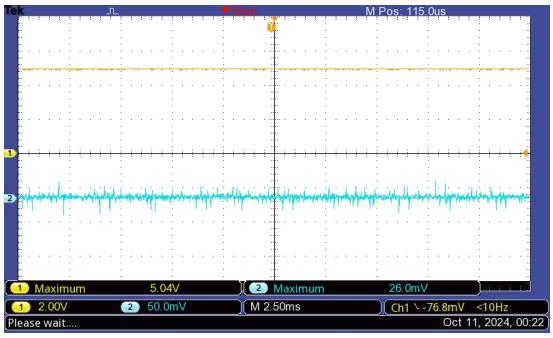


Figure 6.1: Oscilloscope measurement of 5V power rail

6.2.2. I2C Lines

Figure 6.2 presents the I2C lines measured using the oscilloscope. Channel 1 represents the SDA (data) line, while Channel 2 displays the SCL (clock) line. Both signals exhibit stable behaviour, with voltage levels of $V_{SDA} = 3.58$V and $V_{SCL} = 3.70$V. These measurements confirm proper communication between the sensors and the microcontroller, indicating that the I2C lines operate within the expected voltage range. Figure 6.3 depicts the noise measurement on the I2C lines, with the oscilloscope set to AC coupling. The peak-to-peak noise measured on the SDA line is $V_{pp} = 48$mV, and on the SCL line, $V_{pp} = 44$mV. This noise level is unlikely to impact sensor accuracy, as the ripple remains well within the logic level margins of the I2C protocol. Consequently, the minor noise present will not interfere with reliable signal detection.

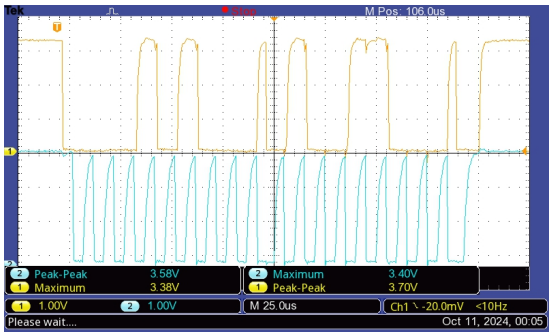


Figure 6.2: I2C Lines Measurement

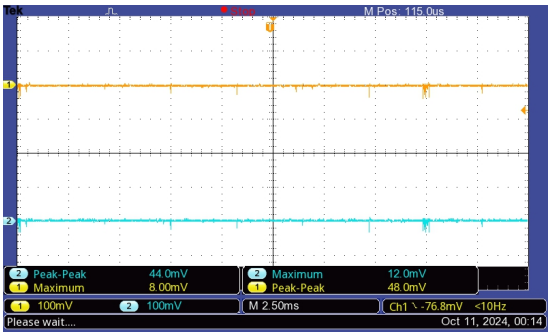


Figure 6.3: Noise on I2C Lines

6.3. Sensor Data Analysis

The following measurements were taken at 1 minute intervals over a total duration of 10 minutes. During this period, sensor data was collected and stored, including heart rate, oxygen level, and temperature, for analysis. Table 6.1 below illustrates the data points collected. Figures 6.4, 6.5, and 6.6 below illustrate Python-generated graphs of measurements over time. Each graph displays 10 points within the 10-minute measurement window.

Figure 6.4 above illustrates a resting heart rate over 10 minutes. The heart rate ranges between a minimum of 60 BPM and a maximum of 68 BPM, demonstrating low fluctuation,

Data Point	Heart Rate (BPM)	Oxygen Level (%)	Temperature (°C)
1	60	98	36.45
2	63	98	36.70
3	64	98	36.55
4	62	98	37.10
5	65	98	36.85
6	63	99	36.60
7	66	99	36.89
8	61	99	36.75
9	68	98	36.90
10	65	98	36.70

Table 6.1: Measurement of health sensors over 10 minutes

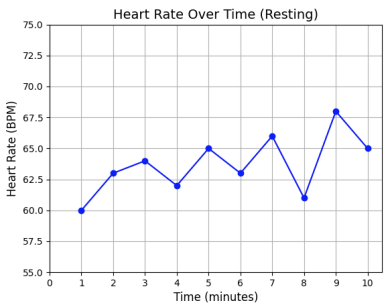


Figure 6.4: Heart Rate Measurements (10 min)

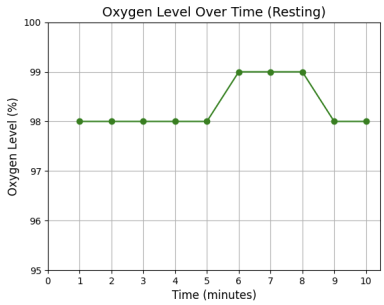


Figure 6.5: Oxygen Level Measurements (10 min)

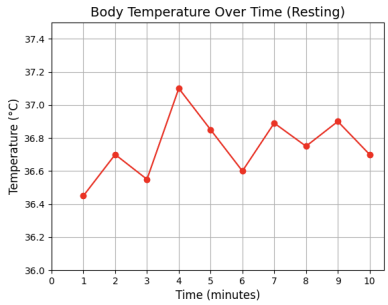


Figure 6.6: Temperature Measurements (10 min)

which is expected during rest. The average heart rate is 64 BPM, with a standard deviation of $\sigma = 2.37$, reflecting consistent heart activity with minimal variability. Oxygen levels (SPO_2) in Figure 6.5 remain relatively stable between 98% and 99%, with a slight spike at 5 minutes and a minor drop at 9 minutes, which aligns with normal bodily patterns, since oxygen levels generally do not fluctuate drastically. The temperature readings in Figure 6.6 remain within the normal range, with a minimum of $T = 36.45°C$ and a maximum of $T = 37.10°C$, indicating small natural variations. The measurements align with normal ranges of measurement for a healthy individual.

6.4. Comparison with Standard Health Devices

The IoT health device was tested to validate the accuracy of its health measurements by comparing its results with established monitoring devices. The selected comparison devices include an Apple Watch and certified medical devices that measure the relevant health parameters. Temperature measurements were compared with a certified digital thermometer, heart rate was validated using a medically approved blood pressure monitor, and oxygen levels were compared with a transmissive pulse oximeter.

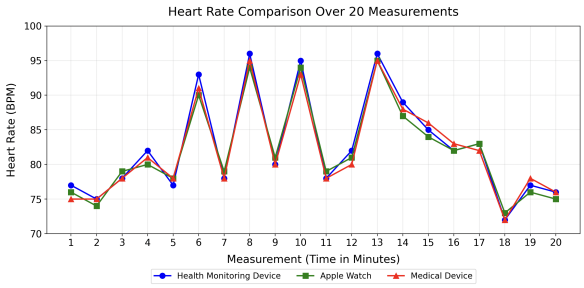


Figure 6.7: Heart Rate Comparison

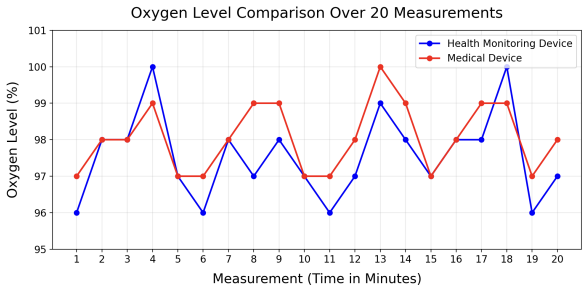


Figure 6.8: Oxygen Level Comparison

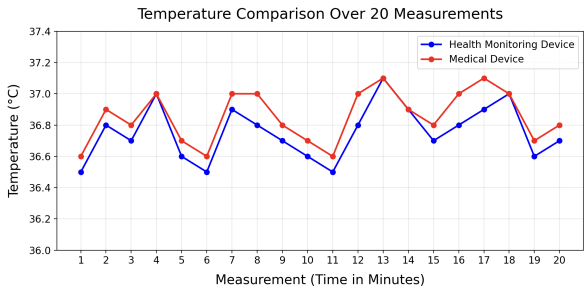


Figure 6.9: Temperature Comparison

Figure 6.7, 6.8 and 6.9 above illustrate the measurement comparison between the IoT health device and medical devices. The plots illustrate that the measurements are closely aligned with only small variations. The mean absolute error (MAE) can be used to determine the error in measurement, using equation 6.1 below. In Equation 6.1, y_i represents the value measured by the medical device, $\hat{y}_i$ represents the corresponding value measured by the IoT health device, and n denotes the total number of measurements used to calculate the Mean Absolute Error (MAE). Table 6.2 below illustrates the MAE for each health parameter.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.1}$$

The MAE for heart rate measurements aligns closely with both the Apple Watch and the medical device, indicating minimal deviation. This shows that the data captured by the sensor are accurate and fall within an acceptable range. For oxygen levels, the MAE between the IoT device and the medical pulse oximeter is 0.70%, indicating a small error margin and confirming the accuracy of the sensor. Similarly, the temperature sensor shows an MAE of 0.1 °C compared to the medical thermometer, aligning with the expected performance, as the accuracy is within 0.1%. These comparison tests validate the accuracy of the device, confirming that the measurements reflect reliable and precise health data collected from individuals. The images in Appendix F provide visual evidence of the measurement comparisons between the IoT health device and established medical devices. The pictures illustrate the test of heart rate and SPO_2, and temperature. The discrepancies are minimal, and the system is accurate.

Table 6.2: MAE Comparison for Heart Rate, Oxygen Levels, and Temperature

Measurement	Apple Watch (MAE)	Medical Device (MAE)
Heart Rate	1.15 BPM	0.85 BPM
Oxygen Levels	–	0.70 %
Temperature	–	0.10 °C

6.5. Web Server Functionality

The web interface serves as the main platform for displaying both real-time and historical health data, Individuals can view heart rate, oxygen levels, and temperature measurements on a browser by connecting to a local network. The web server is divided into multiple pages, including the Dashboard, Detailed View, Health Information, and Historical mode.

6.5.1. Real-Time Data Display

The dashboard displays real-time data of all health parameters with a histogram to compare the measurements. The web server data is updated every 1 second, to ensure real-time display. Figure 6.10 below illustrates a screenshot from the dashboard of the web server, during real-time measurement. Figure 6.11 below illustrates a screenshot of a heart rate measurement within the detailed view page. This illustrates the functionality of the web page for displaying real-time measurements.

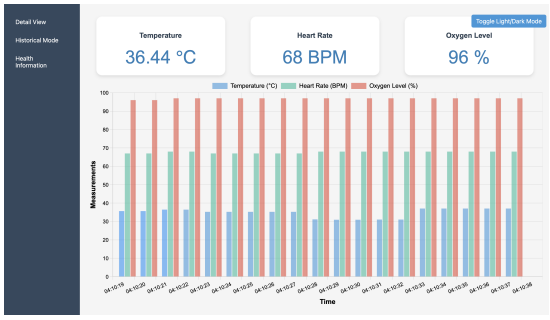


Figure 6.10: Web server Dashboard during measurement

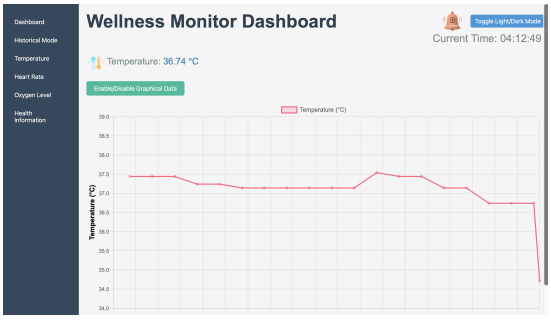


Figure 6.11: Temperature Measurement within Detail View

6.6. Historical Data Display

The historical mode allows users to view previously recorded health measurements, providing insights into trends and averages for heart rate, oxygen levels, and temperature. Data is retrieved from Google Firebase, ensuring secure and reliable storage. This feature enables users to track changes over time and compare past measurements. Figures J.6 and J.7 in Appendix J illustrate screenshots from the historical mode, displaying the average measurements and the historical heart rate page.

Chapter 7

Conclusion

7.1. Summary

The project focused on designing and implementing an IoT-based health monitoring system that measures and captures health data, specifically heart rate, oxygen saturation levels (SPO_2), and body temperature. Although current health tracking solutions exist, they are often costly and are not easily available for personal use outside a healthcare provider setting. This project aims to develop a more affordable and accessible alternative to current solutions. The best possible components were identified via an extensive marketing analysis (Chapter 3), to find the most accurate and cost-effective sensors and components. The hardware design integrated reliable sensors, including the SEN-15219 for measuring heart rate and SPO_2 and the D6T-1A-01 for temperature. The ESP32-C3 microcontroller was selected to ensure precise and real-time data capture. The software design enabled real-time data display on an LCD screen and a web server that allows users to measure real-time and historical data. Additionally, the web server visualizes the data with charts, illustrating trends and averages for users. Google Firebase was used to securely store historical data in the cloud, supporting long-term tracking and trend analysis. The web-based dashboard provided visualization of health metrics and included alert notifications for abnormal readings, enhancing the user experience. The system was calibrated and compared to established medical devices. The measurements were compared to establish discrepancies between the IoT health monitoring device and medical devices. The measurements confirmed that it is within an acceptable range, confirming the accuracy and reliability of the data across all sensors. The results, presented in Chapter 6, demonstrate that the system is accurate and effective for personal health monitoring. In conclusion, the project offers an accessible and cost-effective health monitoring device suitable for personal use.

7.2. Future Work

The device achieved the objectives set out in section 1.3, though there are opportunities to enhance its functionality and expand its capabilities to further improve the user experience, increase accuracy, and enhance its applications in personal health monitoring. The following improvements can be implemented in future versions of the health monitoring device.

- **Integration of Additional Health Metrics:** The system could be expanded to measure additional health metrics, such as blood pressure (BP), electrocardiogram (ECG), and respiratory rate, providing a more comprehensive health monitoring system.
- **Battery Power Integration:** Converting the device to battery power would enhance user convenience and portability.
- **Integration with Healthcare Provider Systems:** The device's IoT capabilities could be developed to integrate with electronic health records (EHR), allowing data exchange with healthcare providers.
- **Size Reduction of Design:** The device could be redesigned to reduce its size, making it more compact. A wearable form factor would further improve ease of use and convenience for users.

Bibliography

[1] Particle, "How to select a microcontroller for your iot devices," accessed: 17 July 2024. [Online]. Available: https://www.particle.io/iot-guides-and-resources/mcu-vs-soc-vs-microprocessor-for-iot/#:~:text=A%20microcontroller%20unit%20(MCU)%20is,e.g.%20cellphones%3B%20household%20electronics

[2] W. Sterlini *et al.*, "Esp32 useful wi-fi library functions (arduino ide)," https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/, 2023, accessed: 23 July 2024.

[3] DigiKey Electronics, "Lm35dz/lft4," n.d., accessed: 24 July 2024. [Online]. Available: https://www.digikey.co.za/en/products/detail/texas-instruments/LM35DZ%2FLFT4/3640794

[4] ScienceDirect Topics, "Infrared thermometer," https://www.sciencedirect.com/topics/engineering/infrared-thermometer, n.d., accessed: 02 August 2024.

[5] T. Leppänen *et al.*, "Pulse oximetry: The working principle, signal formation, and applications," in *Advances in the Diagnosis and Treatment of Sleep Apnea*, ser. Advances in Experimental Medicine and Biology, T. Penzel and R. Hornero, Eds. Springer, Cham, 2022, vol. 1384. [Online]. Available: https://doi.org/10.1007/978-3-031-06413-5_12

[6] C. Park and B. Lee, "Real-time estimation of respiratory rate from a photoplethysmogram using an adaptive lattice notch filter," *BioMedical Engineering OnLine*, vol. 13, no. 1, p. p. 170, 2014.

[7] J. 21 and A. Information, "Tutorial: Improve sensor performance and snr in pulse oximeter designs," Analog Devices, 2010, accessed: 29 August 2024. [Online]. Available: https://www.analog.com/en/resources/technical-articles/tutorial-improve-sensor-performance-and-snr-in-pulse-oximeter-designs.html

[8] ESPRESSIF, "Esp32-c3-devkitc-02," https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32c3/esp32-c3-devkitc-02/user_guide.html#hardware-revision-details, n.d., accessed: 12 September 2024.

[9] Oxyaider, "Blood oxygen meter SPO2 monitor fingertip pulse oximeter," 2022, accessed: 14 October 2024. [Online]. Available: https://www.oxyaider.co.za/product/blood-oxygen-meter-spo2-monitor-fingertip-pulse-oximeter/

[10] Dr Trust, "Dr Trust USA Flexible Tip Digital Thermometer 609 for infants, kids, and adults, fever checking device, oral, rectal, and under arm body temperature measurements," n.d., accessed: 14 October 2024. [Online]. Available: https://drtrust.in/products/dr-trust-usa-digital-thermometer-flexible-tip-609

[11] Apple Support, "Monitor your heart rate with Apple Watch," n.d., accessed: 14 October 2024. [Online]. Available: https://support.apple.com/en-us/120277

[12] PR, "Global patient monitoring devices market report and forecast 2024-2032," https://www.expertmarketresearch.com/reports/patient-monitoring-market, no date, accessed: 01 October 2024.

[13] World Health Organization, "World bank and who: Half the world lacks access to essential health services, 100 million still pushed into extreme poverty because of health expenses," https://www.who.int/news/item/13-12-2017-world-bank-and-who-half-the-world-lacks-access-to-essential-health-services-100-million-still-pushed-into-extreme-poverty-because-of-health-expenses, no date, accessed: 01 October 2024.

[14] A. Gillis, B. Posey, and S. Shea, "What are iot devices?: Definition from techtarget, iot agenda," 2023, accessed: 08 July 2024. [Online]. Available: https://www.techtarget.com/iotagenda/definition/IoT-device

[15] I. Magazine, "What is iot? a beginner's guide - 5 things you need to know about iot," IoT Now News - How to run an IoT enabled business, 2023, accessed: 08 July 2024. [Online]. Available: https://www.iot-now.com/2023/04/05/97056-what-is-iot/#:~:text=The%20IoT%20works%20by%20connecting,performance%2C%20and%20control%20their%20behavior

[16] J. Segarra, "How do iot sensors work?" 2021, accessed: 15 July 2024. [Online]. Available: https://www.soracom.io/blog/how-do-iot-sensors-work/

[17] P. Viorel, "A guide for selecting the right microcontroller for your iot project," 2023, accessed: 17 July 2024. [Online]. Available: https://www.iiot-world.com/industrial-iot/connected-industry/a-guide-for-selecting-the-right-microcontroller-for-your-iot-project/

[18] D. Instruments, "Networking, iot and wifi 101," https://defineinstruments.com/blog/networking-iot-and-wifi-101/, 2021, accessed: 23 July 2024.

[19] Apure, "How does a temperature sensor work?" 2022, accessed: 27 July 2024. [Online]. Available: https://apureinstrument.com/blogs/how-does-a-temperature-sensor-work%EF%BC%9F/#:~:text=Temperature%20sensors%20work%20by%20measuring,classified%20according%20to%20their%20connections

[20] Industrial Quick Search, "Types, uses, benefits, design," accessed: 28 July 2024. [Online]. Available: https://www.iqsdirectory.com/articles/thermocouple/temperature-sensors.html

[21] J. Gyorki *et al.*, "Infrared temperature sensor: What is it and how does it work?" Sensor Tips, n.d., accessed: 02 August 2024. [Online]. Available: https://www.sensortips.com/temperature/infrared-temperature-sensor/

[22] Unacademy, "Understanding the planck law," https://unacademy.com/content/neet-ug/study-material/physics/understanding-the-planck-law/#:~:text=A%20black%2Dbody%20absorbs%20and,depends%20on%20the%20body's%20temperature., 2022, accessed: 02 August 2024.

[23] Hukseflux, "What is a thermopile?" https://www.hukseflux.com/library/what-is-a-thermopile#:~:text=Thermopiles%20are%20sensors%20that%20convert,parts%20that%20perform%20the%20measurements., n.d., accessed: 02 August 2024.

[24] E. D. Chan, M. M. Chan, and M. M. Chan, "Pulse oximetry: Understanding its basic principles facilitates appreciation of its limitations," *Respiratory Medicine*, vol. 107, no. 6, pp. 789–799, 2013.

[25] S. Lopez, "Pulse oximeter - fundamentals and design," NXP, n.d., accessed: 28 July 2024. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN4327.pdf

[26] "Development of smart iot based biomedical heart rate oximeter for driving safety," *Academic Journal of Engineering and Technology Science*, vol. 6, no. 12, 2023.

[27] R. Semiconductor, "Heart rate sensor module: Rohm semiconductor - rohm co., ltd." https://www.rohm.com/sensor-shield-support/bh1790glc#:~:text=An%20optical%20heart%20rate%20sensor,optical%20sensor%20and%20green%20LED., accessed: 2 August 2024.

[28] Ansys, "Using ppg for designing optical heart rate sensors," https://www.ansys.com/blog/modeling-human-skin-and-optical-heart-rate-sensors, accessed: 02 August 2024.

[29] N. Saquib, M. T. I. Papon, I. Ahmad, and A. Rahman, "Measurement of heart rate using photoplethysmography," https://cs-people.bu.edu/papon/pdfs/2015a_NS_MTIP_IA.pdf, accessed: 04 August 2024.

[30] L. Yuan, "Iot based health monitoring system," 2019, accessed: 18 July 2024. [Online]. Available: http://eprints.utar.edu.my/4040/1/3E_1501917_FYP_report_-_CHEE_YUAN_LIM.pdf

[31] ST, "Positive voltage regulator ics," https://www.st.com/content/ccc/resource/technical/document/datasheet/41/4f/b3/b0/12/d4/47/88/CD00000444.pdf/files/CD00000444.pdf/jcr:content/translations/en.CD00000444.pdf, 2018, accessed: 12 September 2024.

[32] Sparkfun, "Sparkfun pulse oximeter and heart rate monitor hookup guide," https://learn.sparkfun.com/tutorials/sparkfun-pulse-oximeter-and-heart-rate-monitor-hookup-guide#hardware-overview, n.d., accessed: 14 September 2024.

[33] Omron-Devhub, "D6T-2JCIEEV01-Arduino/examples/D6T-1A/d6t-1a.ino at master · Omron-devhub/D6T-2JCIEEV01-arduino," https://github.com/omron-devhub/d6t-2jcieev01-arduino/blob/master/examples/d6t-1a/d6t-1a.ino, n.d., accessed: 20 September 2024.

Appendix A

Project Planning Schedule

Table A.1: Project Planning Schedule for IoT Health Device

Week	Date Range	Task
Week 1	22 July – 28 July	**Literature Review**: Research on sensors, technologies, and methodologies
Week 2	29 July – 4 August	**Literature Review (Writing)**: Document literature review section in report.
Week 3	5 August – 11 August	**Component Selection (Design)**: Decide on components and order components, write hardware selection in the report. Submit supervisor agreement form on Sunlearn.
Week 4	12 August – 18 August	**Hardware Design (Schematic)**: Design circuit schematic, layout, and write Hardware design section in the report
Week 5	19 August – 25 August	**PCB Design and Enclosure Design**: Create PCB and enclosure layout
Week 6	26 August – 1 September	**Write PCB and Enclosure Design**: Print the PCB and enclosure and document it in the report.
Week 7	2 September – 8 September	**Software Design (Coding)**: Develop software, integration of sensors and web server
Week 8	9 September – 15 September	**Write Software Design**: Document software design in report
Week 9	16 September – 22 September	**Integration (Hardware)**: Assemble and test hardware, and integrate software
Week 10	23 September – 29 September	**Measurements**: Collect data and measurements
Week 11	30 September – 6 October	**Write Measurement Results**: Write measurement section in report
Week 12	7 October – 13 October	**Final Draft Hand In**: Refine report for draft hand-in
Week 13	14 October – 20 October	**Write Conclusion and Refinement**: Complete the report conclusion and proofread
Week 14 & 15	21 October – 4 November	**Final Submission Prep**: Prepare for submission, review, and submit the report before 12pm 4 November 2024.

Appendix B

Outcomes Compliance

Outcome	Motivation	Sections
GA 1: Problem Solving	The project identified a need for an accessible IoT health monitoring device that addresses limitations in current health systems, including cost, accessibility, and data management. A comprehensive literature review informed the development of a device capable of capturing and displaying real-time and historical health data, shown on both a local display and a cloud-based web server. The design, implementation, and testing phases overcame hardware and software challenges, including sensor calibration, hardware integration, communication protocols, secure data transmission, and effective data display on multiple interfaces.	1.1, 1.2, 1.3, 2
GA 2: Application of Scientific and Engineering Knowledge	The project applied knowledge of Internet of Things (IoT), sensor calibration, microcontroller software development, and data security. Engineering principles in electronics, programming, embedded systems, circuit design, data visualization, and wireless communication were integrated into the design of hardware and software components, supporting the functionality of the health monitoring system.	3, 4, 5, 6.2, 6.3, 6.4

GA 3: Engineering Design	The IoT health monitoring device was designed through a step-by-step process focused on accuracy, reliability, and accessibility. Component selection was guided by an initial literature review and market analysis to ensure compatibility and sensor accuracy, as well as efficient data processing. The hardware design prioritized stable power regulation and effective integration of communication protocols, optimising GPIO pin usage on the ESP32-C3. Modular software development enabled secure data handling, error management, and fast health data transmission. Iterative testing at each stage optimized performance, resulting in a robust system that meets the project objectives.	3, 4, 5
GA 4: Investigations, Experiments and Data Analysis	The health monitoring device underwent extensive tests and experiments to validate its accuracy and functionality. Initial testing focused on sensor calibration and communication integrity, with oscilloscope measurements confirming stable power delivery and low noise levels on the sensor communication lines. Sensor accuracy was validated by comparing data collected from the health monitoring device with that from standard medical monitoring devices, ensuring the system's measurements were within acceptable ranges. Futhermore, the effectiveness of the web server was tested, to ensure reliabile data visualisation and data management of historical data. This iterative testing and analysis process ensured high reliability and accuracy, meeting the project's performance objectives.	5.2.3, 6

GA 5: Engineering Methods, Skills, and Tools, Including IT	The health monitoring system was developed using a range of engineering methods, tools and skills. Autodesk Inventor Professional was used for the enclosure design, while kiCad was utilized to design the PCB layout and 3D model. Programming in Arduino IDE provided a method for controlling the microcontroller, with a web server built using HTML, CSS, and JavaScript to display historical and real-time data. Historical data is stored on Google Firebase. Python was implemented for generating plots for analysis of measurements. During testing, multimeters, oscilloscopes, bench power supplies, prototype boards, and soldering irons were essential engineering tools for validating and integrating the hardware. Progressive testing methods ensured both hardware and software functioned as expected. Skills in circuit design, embedded programming, and troubleshooting were paired with engineering methods, such as iterative testing and modular design, resulting in a well-integrated health monitoring device.	4.1.6, 4.1.7, 5.1, 5.2, 5.3, 5.4, 5.5, 6.1, 6.2, 6.3, 6.4
GA 6: Professional and Technical Communication	The health monitoring system project demonstrates effective professional and technical communication through a comprehensive written report and an oral presentation. The report provides a structured, detailed account of the design, implementation, testing, and analysis processes, using clear technical language and visual aids like diagrams and plots to support understanding. The report and presentation showcase competence in written and oral communication, which is essential for conveying technical information.	All sections
GA 8: Individual Work	All aspects of the project, from design to testing and documentation, were completed independently, demonstrating a high level of individual effort under the guidance of supervision.	All sections

GA 9: Independent Learning Ability	The project required independent learning to acquire knowledge beyond what was provided in the E&E undergraduate program. This involved researching literature and online resources to gain an understanding of various health sensors, calibration techniques, data security, and IoT integration. Skills in web server development using HTML, CSS, and JavaScript, as well as PCB design with KiCad, were self-taught through technical documentation, online tutorials, and academic papers. Additionally, principles of User Interface (UI) design were studied to create a clear and accessible web interface, and wireless communication protocols were explored to enable reliable data transfer between the embedded system and remote platforms. This self-directed learning ensured that all aspects of the project and its objectives were met.	2, 3.2, 4.1.6, 5.2, 5.3, 5.4, 5.5

Table B.1: ECSA Outcomes Compliance

Appendix C

Comparison of components considered for Hardware Selection

Parameter	ESP32-C3-DevKitC-02	STM32F103C8T6	RP Pico	Arduino Nano 33 IoT
Architecture	32-bit RISC-V	32-bit ARM Cortex-M3	32-bit ARM Cortex-M0+	Single-Core ARM Cortex-M0+
Clock Speed	160 MHz	72 MHz	133 MHz	48 MHz
Memory	400 KB SRAM, 4 MB Flash	20 KB SRAM, 64 KB Flash	264 KB SRAM, 2 MB QSPI Flash	32 KB SRAM, 256 KB Flash
Power Consumption	Ultra-low power	Low power	Low power	Low power
Operating Voltage	3.3 V	3.3 V	3.3 V	3.3 V
Wireless Connectivity	Wi-Fi, BLE 5.0	None	2.4GHz 802.11n wireless LAN, Bluetooth 5.2	Wi-Fi, BLE 4.2
Wired Connectivity	UART, I2C, SPI, USB	UART, I2C, SPI, USB	UART, I2C, SPI, USB	UART, I2C, I2S, SPI, USB
Cost	± R150	± R430	± R100	± R450

Table C.1: Comparison of Different Microcontrollers

Parameter	MCP9700-E/TO	D6T-1A-01	DS18B20	WATERPROOF DS18B20
Type	Analogue	Digital	Digital	Digital
Range	-40°C - 125°C	0°C - 60°C	-55°C - 125°C	-55°C - 125°C
Accuracy	±4°C	Very High	±0.5°C	±0.5°C
Mechanism of measurement	Contact	Non-Contact (IF)	Contact	Contact
Operating Voltage	2.3V - 5.5V	4.5V - 5.5V	3V - 5.5V	3V - 5.5V
Connectivity	ADC Pin	I2C	GPIO Pin	GPIO Pin
Cost	± R10	± R300	± R136	± R200

Table C.2: Comparison of Temperature Modules

Parameter	SEN-15219	BH1792GLC	MIKROE-4986	MAX86140EVSYS
Type	Digital	Digital	Digital	Digital
Dimensions	25.4 x 12.7 mm	2.8 x 2.8 mm	42.9 x 25.4 mm	2.048 x 1.848 mm
Mounting type	Through-hole	Surface-mount device (SMD)	Through-hole	Surface-mount device (SMD)
Transmissive / Reflective	Reflective	Reflective	Reflective	Reflective
Operating Voltage	3.3 V	3.3 V	3.3/5 V	1.8 V
Connectivity	I2C	I2C	I2C	SPI
Cost	± R790	± R205	± R807	± R2670

Table C.3: Comparison of Oximeter Modules

Appendix D

Full Schematic of device

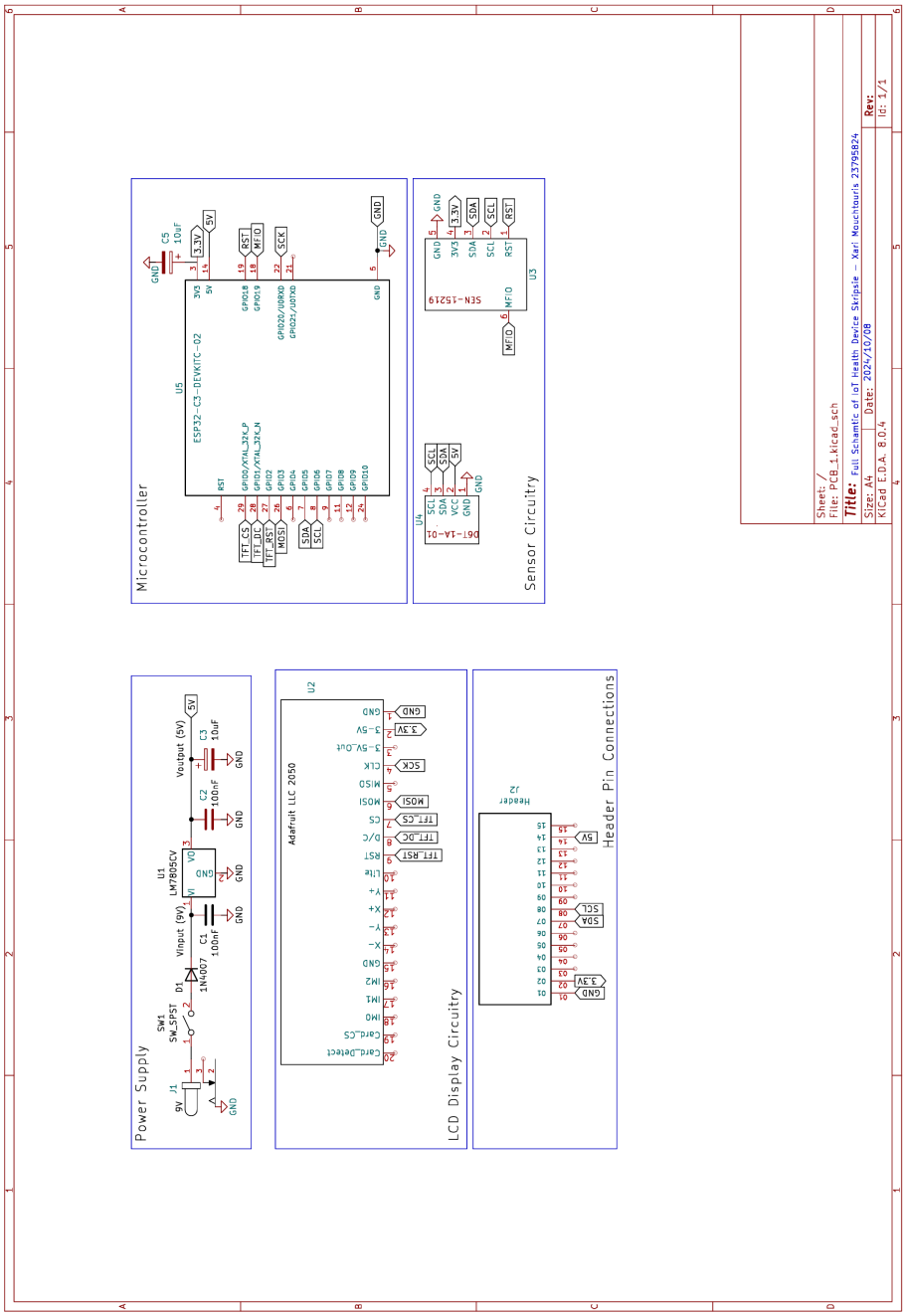


Figure D.1: Full Schematic of the Device

Appendix E

Measurements of health monitoring device

Table E.1: Comparison of Measurements: Established Medical Devices vs. IoT Health Device

Measurement (#)	IoT Device (BPM)	Apple Watch (BPM)	Medical Device (BPM)
1	77	76	75
2	75	74	75
3	78	79	78
4	82	80	81
5	77	78	78
6	93	90	91
7	78	79	78
8	96	94	95
9	80	81	80
10	95	94	93
11	78	79	78
12	82	81	80
13	96	95	95
14	89	87	88
15	85	84	86
16	82	82	83
17	83	83	82
18	72	73	72
19	77	76	78
20	76	75	76

Table E.2: Comparison of Oxygen Saturation (SpO_2) Levels: IoT Device vs. Medical Device

Measurement (#)	**IoT Device (SpO_2 %)**	**Medical Device (SpO_2 %)**
1	96	97
2	98	98
3	98	98
4	100	99
5	97	97
6	96	97
7	98	98
8	97	99
9	98	99
10	97	97
11	96	97
12	97	98
13	99	100
14	98	99
15	97	97
16	98	98
17	98	99
18	100	99
19	96	97
20	97	98

Table E.3: Comparison of Temperature Measurements: IoT Device vs. Medical Device

Measurement (#)	IoT Device (°C)	Medical Device (°C)
1	36.5	36.6
2	36.8	36.9
3	36.7	36.8
4	37.0	37.0
5	36.6	36.7
6	36.5	36.6
7	36.9	37.0
8	36.8	37.0
9	36.7	36.8
10	36.6	36.7
11	36.5	36.6
12	36.8	37.0
13	37.1	37.1
14	36.9	36.9
15	36.7	36.8
16	36.8	37.0
17	36.9	37.1
18	37.0	37.0
19	36.6	36.7
20	36.7	36.8

Appendix F

Verification of Measurement Accuracy Against Established Medical Devices

These images illustrate real-time comparison measurements for heart rate and oxygen saturation (SpO_2) between the IoT health device and an established medical device. Each timestamped image provides evidence of consistency between the devices' readings, demonstrating the accuracy of the IoT health monitor.

Figure F.1: Heart Rate and SPO_2 comparison picture: (Timestamp: 12:10:03 am)

Figure F.2: Heart Rate and SPO_2 comparison picture: (Timestamp:12:10:22 am)

The final image captures a temperature comparison between the IoT device and a standard thermometer, highlighting the device's capability to measure temperature accurately.

Figure F.3: Heart Rate and SPO_2 comparison picture: (Timestamp: 12:10:29 am)

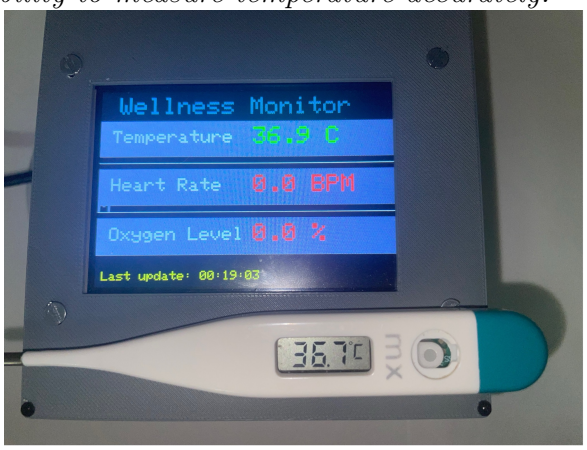


Figure F.4: Temperature Comparison with thermometer

Appendix G

Cad Design

Figures G.1, G.2, G.4 and G.5 illustrate the CAD design of the physical build with the PCB included.

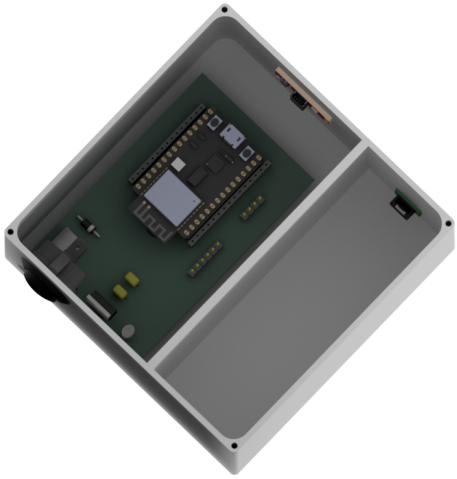


Figure G.1: Internal View of PCB Inside Enclosure

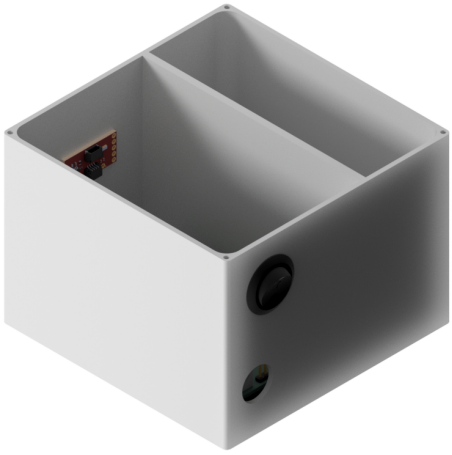


Figure G.2: Orthographic view 1

Figure G.3: Enclosure Cad Design 1

Figure G.4: Top-down view of PCB with components

Figure G.5: Orthographic view 2

Figure G.6: Enclosure Cad Design 2

Appendix H

PCB Design and 3D model

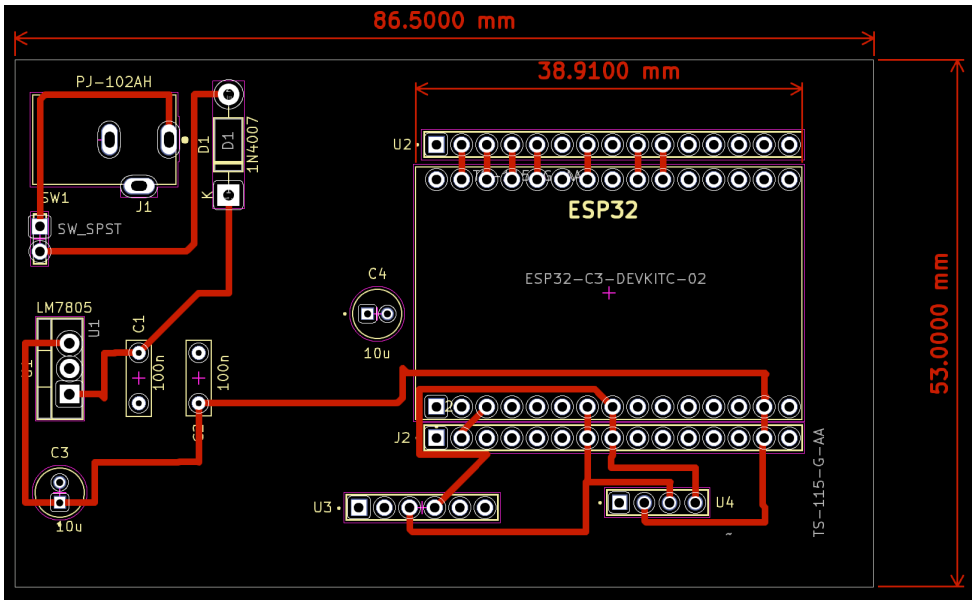


Figure H.1: Top Layer of PCB

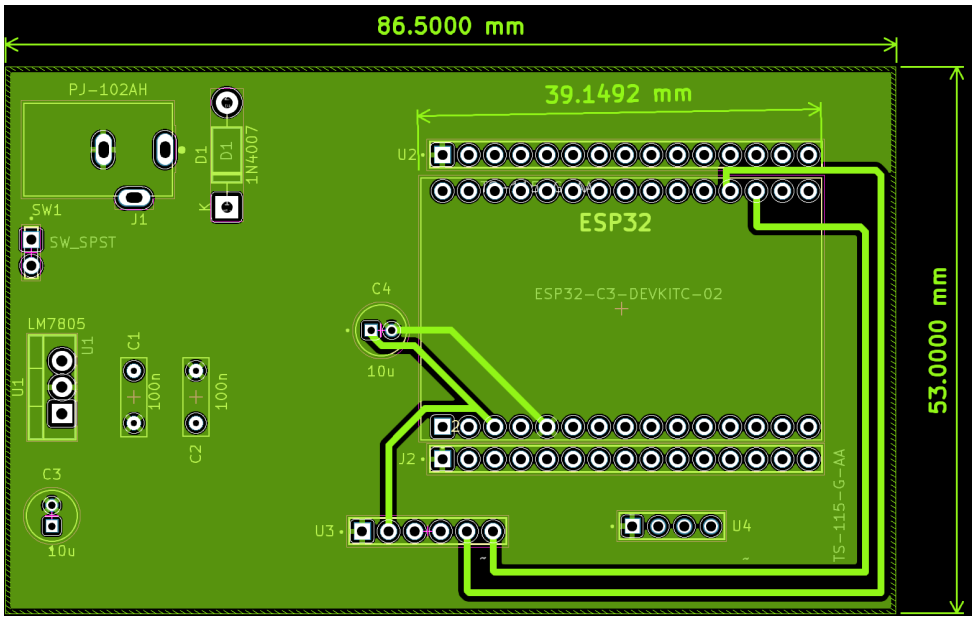


Figure H.2: Bottom Layer of PCB

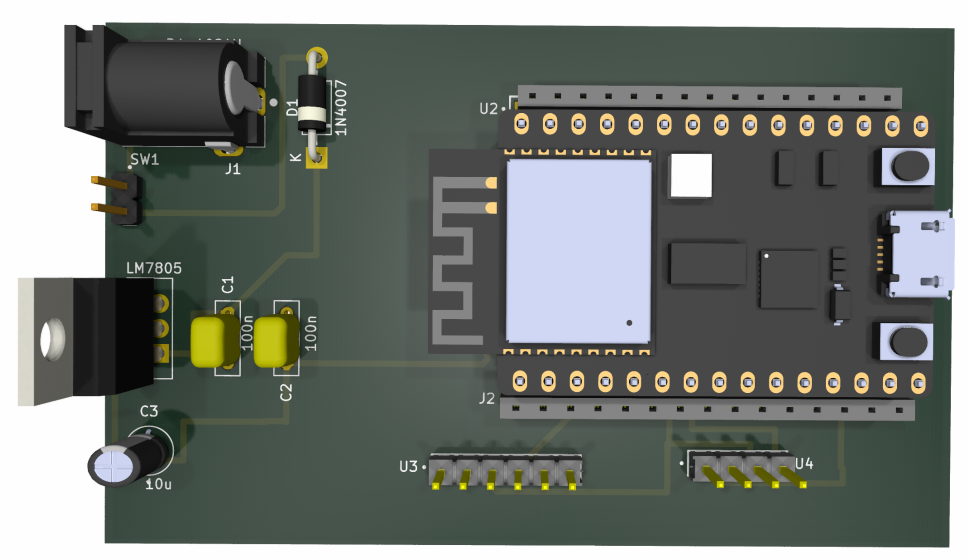


Figure H.3: Top view of 3D model

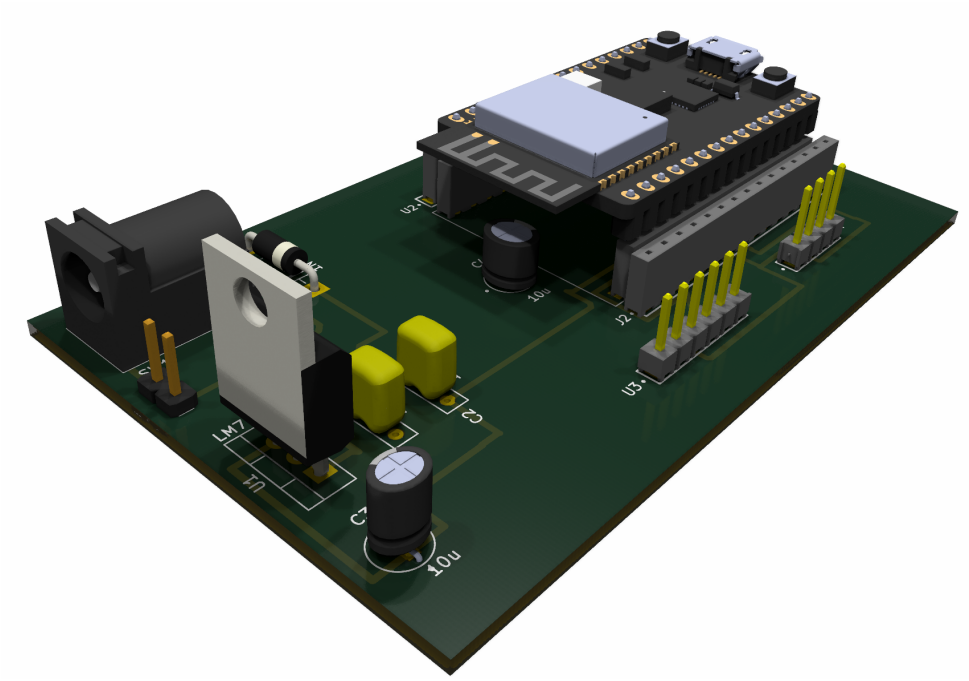


Figure H.4: Right side view of 3D model

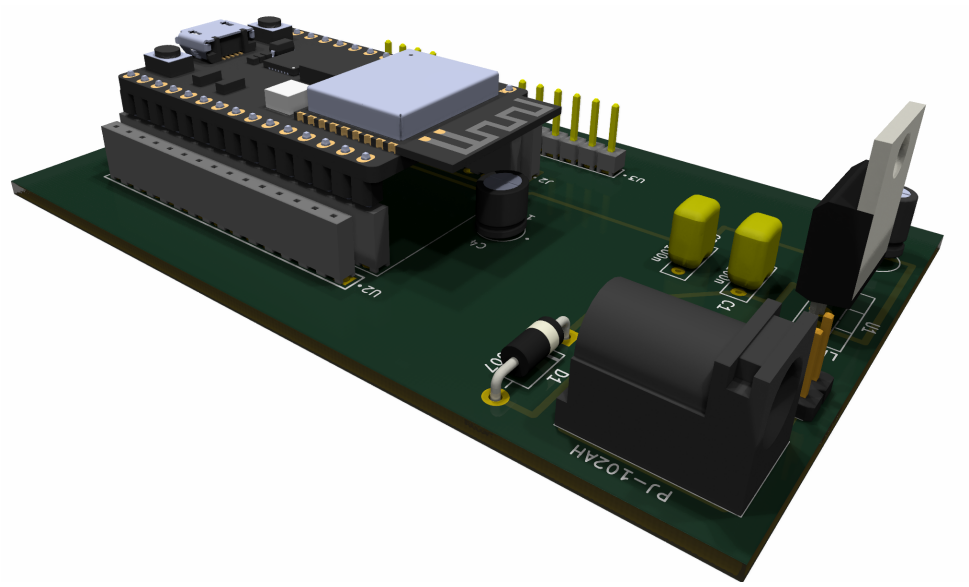


Figure H.5: Left side view of 3D model

Appendix I

Physical build enclosure with PCB

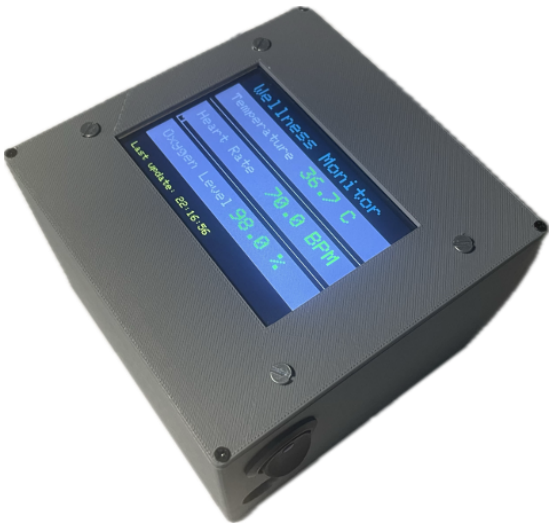


Figure I.1: Orthographic View of Working Device

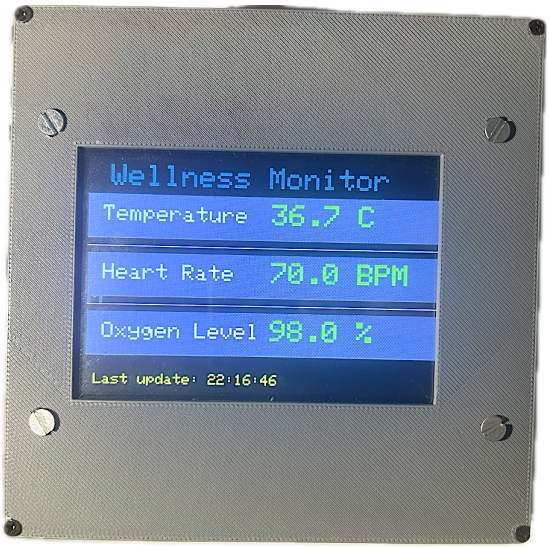


Figure I.2: Normal Measurements: Green

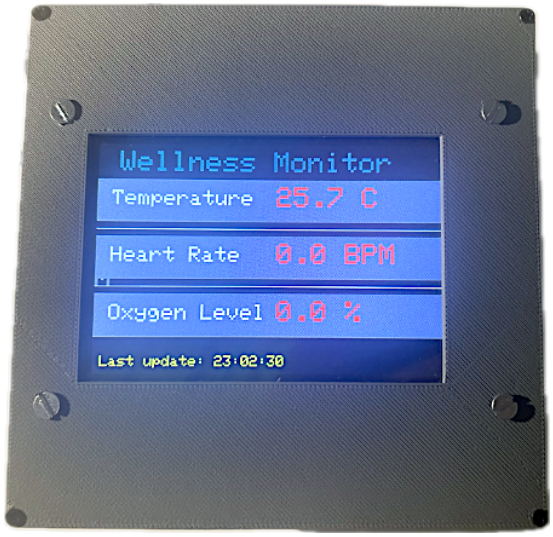


Figure I.3: Abnormal Measurements: Red

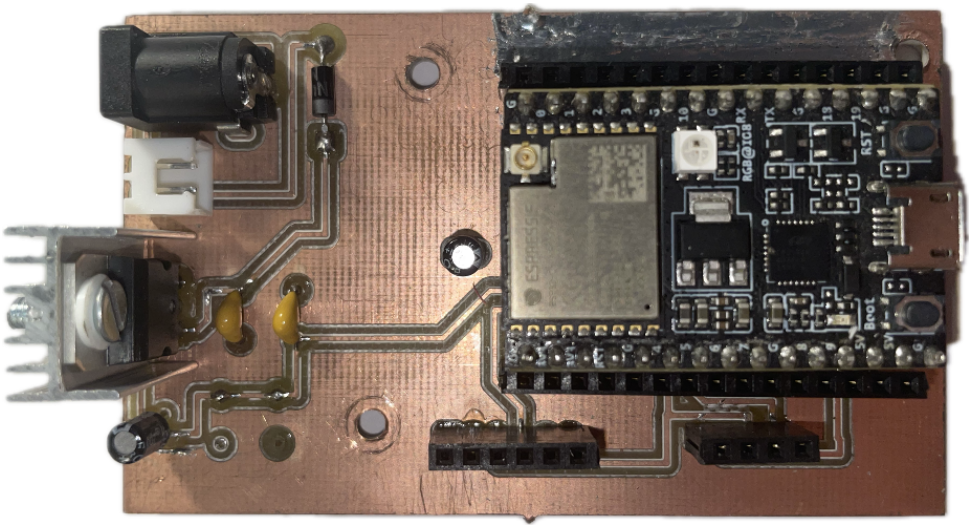


Figure I.4: Physical soldered PCB for Health Device

Appendix J

Health Monitoring Web Server

Figure J.1: This is the main Dashboard page, providing an overview of current health metrics.

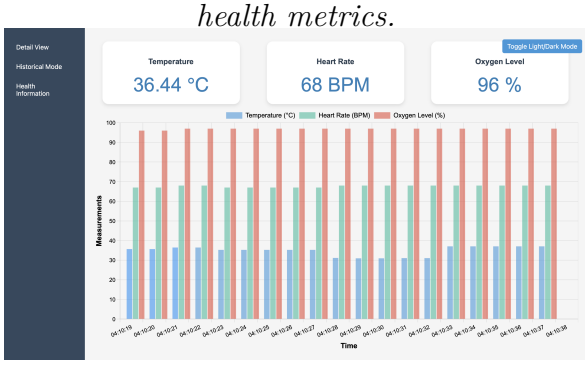


Figure J.1: Dashboard page

Figure J.2: Health Information page that displays information on normal health ranges.

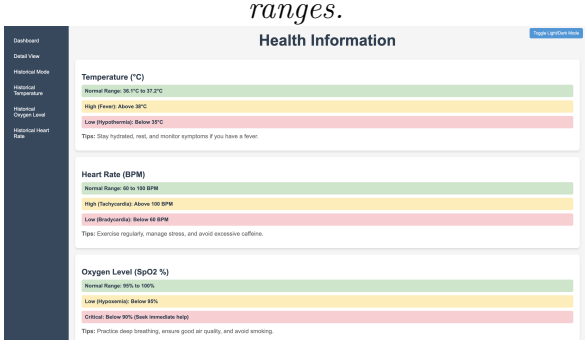


Figure J.2: Health information page

Figure J.3: This page provides a detailed view of the Heart Rate measurement page, during real-time tracking.

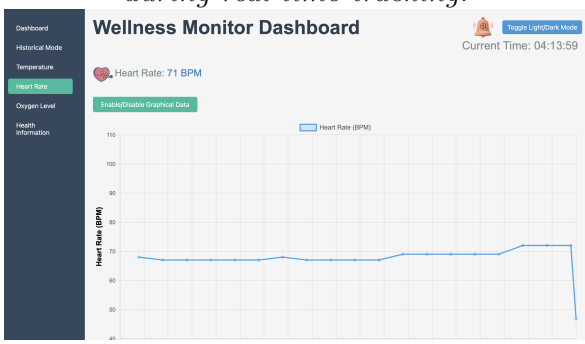


Figure J.3: Detailed view of Heart Rate page

Figure J.4: Detailed page for Oxygen levels, displaying real-time measurement.

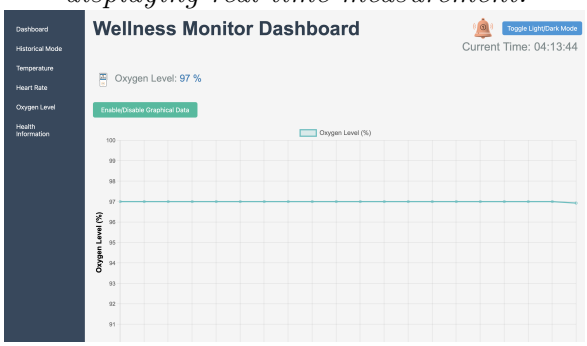


Figure J.4: Detailed view of Oxygen levels page

Figure J.5: Detailed Temperature page with real-time tracking

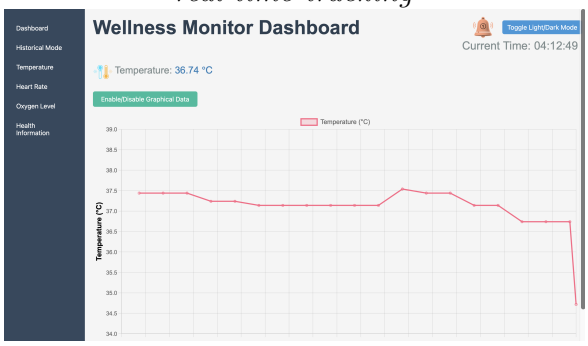


Figure J.5: Detailed view of Temperature page

Figure J.6: Historical Main page that allows access to historical data

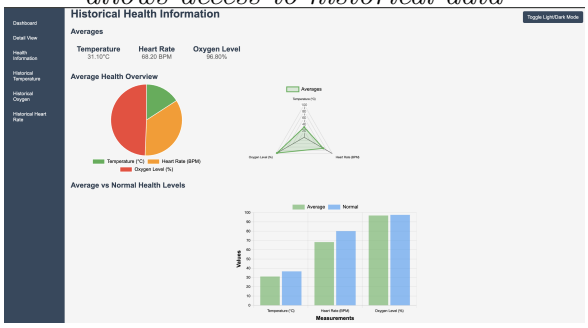


Figure J.6: Historical Main page

Figure J.7: Historical Heart Rate page showing past heart rate data for trend analysis.

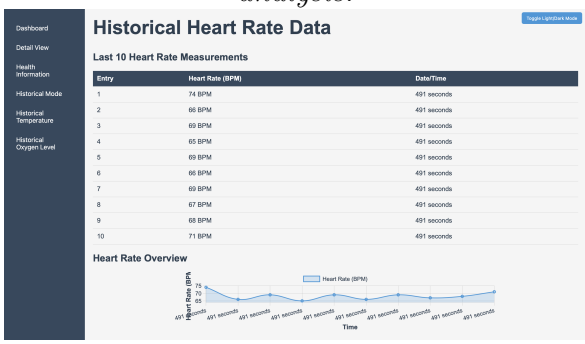


Figure J.7: Historical Heart Rate Page

Figure J.8: Historical Temperature page displaying past temperature measurements.

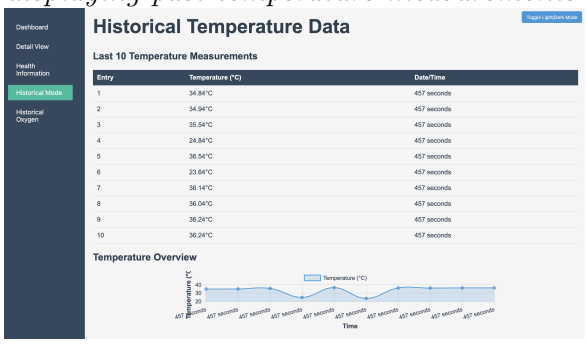
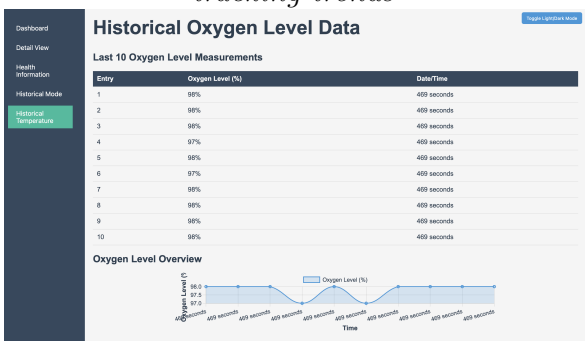


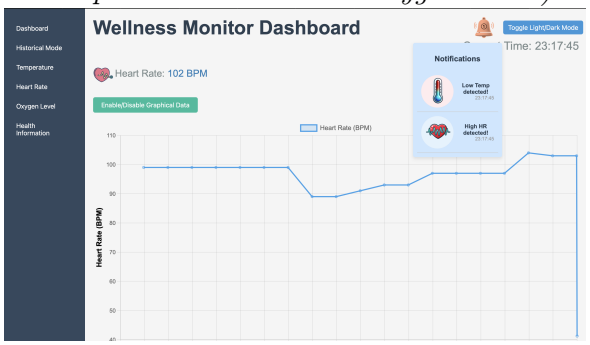
Figure J.8: Historical Temperature Page

Historical data for oxygen levels, useful for tracking trends



(a) Historical Oxygen Page

An abnormal notification alert on the webpage (High Heart Rate and Low Temperature & Normal Oxygen Level)



(b) Abnormal notification alert on webpage

Appendix K

Medical devices Used for Testing and Calibration

Figure K.1: SPO_2 and Heart Rate measurement device [9]

Figure K.2: Medical Thermometer [10]

Figure K.3: Heart Rate measurement with Apple Watch Series 3 [11]

Appendix L

Github Software Repository

The software developed for this project is available in the project's GitHub repository.