DEVELOPMENT OF A WEARABLE HEALTH MONITORING SYSTEM USING ESP32 AND MULTI-SENSOR INTEGRATION

SAMUEL OLADELE

ELECTRONIC ENGINEERING AND COMPUTER SCIENCE MENG

REPORT

ASTON UNIVERSITY

REPORT

DEVELOPMENT OF A WEARABLE HEALTH MONITORING SYSTEM USING ESP32 AND MULTI-SENSOR INTEGRATION

Summary

This report presents the development of a modular, low-cost, and customizable wearable health monitoring device designed to measure body temperature, heart rate, blood oxygen saturation, and physical activity in real time. The project was motivated by the high cost, limited data accessibility, and lack of customisation in existing commercial health monitors.

The system was built around the ItsyBitsy ESP32 microcontroller, integrating three sensors: a high-accuracy temperature sensor (TMP117), a 6-axis motion sensor (LSM6DSO), and a heart rate/SpO_2 sensor (MAX30102). A desktop application was developed in Python to receive and display data wirelessly via Bluetooth. The interface supports real-time plotting, threshold-based alerts, and data export for further analysis. The final prototype was implemented into a wearable garment to ensure sensor contact and user comfort.

Testing confirmed the system's accuracy, with results showing average errors under 2% across all sensors. Response times and motion tracking performance were suitable for real-time applications. User feedback highlighted the dashboard's clarity and ease of setup, with minor concerns around the bulkiness of the prototype.

My individual contribution included the full design and integration of the hardware and software systems, firmware development for real-time data collection and wireless transmission, and the creation of a cross-platform graphical interface. Where existing sensor libraries or reference code were used, they were adapted to meet the specific needs of this project and are fully credited.

This project demonstrates that open-source tools and consumer-grade hardware can be combined to build accessible and reliable health monitoring systems. It offers a promising platform for further research, educational use, and future enhancements, such as mobile compatibility, cloud integration, and advanced health analytics.

This report work is dedicated to Jesus Christ, my Lord and Saviour, who without His constant support, I would not be where I am today.

Acknowledgments

I would like to express my gratitude to my primary supervisor, Alex Rozhin, for his assistance and guidance throughout this project.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Meaning
MCU	Microcontroller Unit
ECG	Electrocardiogram
I^2C	Inter-Integrated Circuit
GUI	Graphical User Interface
BLE	Bluetooth Low Energy
PCB	Printed Circuit Board
BPM	Beats Per Minute
PPG	Photoplethysmography
SpO_2	Blood Oxygen Saturation
IMU	Inertial Measurement Unit
UUID	Universally Unique Identifier
GATT	Generic Attribute Profile
OTA	Over-The-Air
CSV	Comma-Separated Values
JSON	JavaScript Object Notation
PyQt	Python Qt
mAh	Milliampere-hour
Wi-Fi	Wireless Fidelity
ADC	Analog-to-Digital Converter
IDE	Integrated Development Environment

1 Introduction

This report focuses on the development of a wearable health monitoring device using the Adafruit ItsyBitsy ESP32 microcontroller integrated via I2C communication with a TMP117 temperature sensor, LSM6DSO accelerometer and gyroscope, and MAX30102 heart rate and blood oxygen sensor. The ItsyBitsy ESP32 was chosen for its compact design and robust capabilities, and integrated Wi-Fi and Bluetooth functionalities , making it well-suited for high-speed data acquisition and real-time wireless transmission.

The device integrates three high-precision sensors:

- TMP117 Temperature Sensor: This sensor offers 16-bit resolution with an accuracy of ±0.1°C across a temperature range of –20°C to 50°C .

- LSM6DSO Accelerometer and Gyroscope: A 6-axis IMU that combines a 3-axis accelerometer and a 3-axis gyroscope. It supports accelerometer ranges of ±2/±4/±8/±16 G, with data rates up to 6.66 kHz, allowing for precise motion tracking .

- MAX30102 Heart Rate and SpO_2 Sensor: This integrated module performs photoplethysmography (PPG) using red (660 nm) and infrared (880 nm) LEDs for heart rate and blood oxygen saturation measurements .

The sensors were selected based on their accuracy, affordability, low power consumption, and seamless I^2C compatibility with the ESP32 platform. Their integration enables continuous tracking of temperature, motion, and cardiovascular metrics, with a high degree of data transparency and adaptability.

1.1 Background

Rapid advancement of wearable technology has revolutionised both the fitness and healthcare industry, enabling real-time monitoring of vital health parameters and empowering individuals to take proactive measures for their well-being. Wearable health monitoring devices integrate advanced sensors, wireless communication, and data processing capabilities into compact, user-friendly systems that can be worn on the body . These devices have the potential to transform traditional healthcare models by providing continuous, non-invasive monitoring, early detection of abnormalities, and user-personalized health insights .

In recent years, wearable health devices have evolved beyond basic fitness tracking into platforms capable of tracking a wide array of health metrics such as ECG, skin temperature, blood oxygen saturation, respiratory rate, and activity recognition. These trends indicate a growing emphasis on preventive care, remote monitoring, and personal fitness. Devices like the Apple Watch Series, Fitbit Sense, and BioIntelliSense BioSticker demonstrate the increased sophistication of modern wearables in both consumer and clinical contexts. However, many of these devices rely on private algorithms and platforms, limiting user access to raw data and potential for customisation.

Despite advancements in sensor technologies and sensor miniaturisation, the most prevalent setback to global deployment of wearable health monitoring systems remains the current price point of devices on the market. The incorporation of advanced technologies such as condensed sensors, processors, and wireless connectivity, contribute to elevated manufacturing costs . Increased further by the costs of research and development, making them less accessible to the general population, especially in the medical field, as it increases the financial burden for individuals on lower incomes, let alone poorer rural areas, leading to shortage or ineffective use of these technologies in regions that would benefit from its deployment .

Apart from high costs, current consumer-grade wearable devices often lack customisation and suffer from compromised accuracy during physical activity. Photoplethysmography (PPG), commonly used for heart rate and SpO_2 monitoring, is particularly susceptible to motion artifacts, causing unreliable measurements under dynamic conditions . Furthermore, limited data access and restricted third-party integration capabilities hinder their use in academic research and custom health monitoring solutions, posing challenges for researchers who require detailed data for analysis .

1.2 Scope

This project demonstrates the design, development, and testing of a compact, efficient, and affordable solution for continuous personal health monitoring, with real-world application potential. While this project report will cover the design, assembly, and basic functionality of the system, including the integration of hardware components, implementation of software algorithms for data processing, and creation of a user interface for data visualization. It does not include clinical validation or regulatory certification, as they were deemed outside the scope of this academic project. The system will also be evaluated for accuracy and performance, reliability with a focus on sensor calibration, user experience, real-time responsiveness, and potential societal impact.

1.3 Report Structure

Beginning with this section, where the background and project objectives are stated and discussed, the key structure of this report includes:

- A literature review of available resources relevant to wearable health monitoring technologies.
- Design and assembly of the hardware components, including microcontroller selection and sensor integration.
- Development of the embedded firmware for real-time data acquisition, processing, and BLE communication with the ESP32.
- Development of the desktop GUI with Python, for data visualization and logging, enabling real-time monitoring of health metrics.
- Evaluation and testing of the system's accuracy, responsiveness, and usability in real-world conditions and in comparison, to devices discussed in the literature review.
- User feedback analysis to guide future improvements in sensor placement, device form factor, and app functionality.

- The penultimate section will conclude the report, producing a final judgment on this project's success, a comparison against aims.
- Finally followed by the recommendations section that offers targeted improvements for future development.

1.4 Objectives

Developed between September 2024 and April 2025, the objectives of this project were formulated using the SMART criteria to ensure clear direction, practical feasibility, and alignment with the initial project description shown in Appendix A.These objectives were defined as follows:

1. **Develop a functional wearable health monitoring device** that integrates the TMP117 , MAX30102 and LSM6DSO sensors with the Adafruit ItsyBitsy ESP32 microcontroller. The system should be cost-effective, lightweight (<100g), and accurately measure temperature (±0.1°C), heart rate (±2 bpm), and SpO_2 (±2%).

2. **Design and implement a guided user interface** for real-time data visualization and logging. A desktop GUI will be developed using Python to display incoming sensor data at a minimum update rate of 1 Hz and support CSV export for analysis.

3. **Create a compact and ergonomic enclosure** suitable for extended use on the wrist, upper arm, or chest. The form factor will be optimized for comfort, durability, and secure sensor placement to reduce motion-related inaccuracies.

4. **Assess environmental and ethical impacts** by performing a PESTLE analysis to evaluate sustainability, safety, and potential for scalability. Findings will be incorporated into the final report with recommendations for future design improvements.

Extended Objectives

- Visualize sensor data using a mobile application.
- Store sensor data over a period of time, locally or remotely for later analysis.
- Enable real-time alerts for unusual readings (e.g., abnormal heart rate).

These extended objectives were designed to build upon the foundation established by the core objectives, contributing to a more comprehensive and refined product.

1.5 Methodology

This project involved a systematic and structured approach to design, a hybrid methodology combining elements of Agile and Waterfall was decided to be most effective for this project due to its flexibility, efficiency, and alignment with the project's objectives, as the structured phases of Waterfall provided a clear roadmap needed for planning and execution, with the clarity of this project's objectives aiding in providing well-defined goals and deliverables such as developing a wearable device with specific sensor capabilities. While Agile principles allowed

for iterative improvements and quicker adaptations and solutions to unforeseen challenges and events during the development process. Particularly, during the integration and calibration of multiple sensors that required frequent testing and refinement.

1.6 Project Timeline Summary

The project followed a structured timeline divided into nine key phases, as outlined below:

- **Project Initiation** (9 days: 07/10/24 – 15/10/24)
  - Defined scope, objectives, and risks, followed by plan adjustments and a proposal presentation.
- **Requirements Gathering and Planning** (15 days: 16/10/24 – 30/10/24)
  - Collected hardware/software requirements, purchased components, and finalized technical specifications.
- **Hardware Development** (21 days: 31/10/24 – 20/11/24)
  - **Phase 1**: Sensor integration (research, circuit design, prototype build).
  - **Phase 2**: Microcontroller setup (programming, testing).
- **Software Development** (28 days: 21/11/24 – 18/12/24)
  - **Sprint 1**: Mobile app setup, UI design, and basic data communication.
  - **Sprint 2**: Real-time data processing and graphical display.
  - **Sprint 3**: Bluetooth/Wi-Fi integration.
- **Iterative Testing and Prototyping** (21 days: 09/01/25 – 29/01/25)
  - **Cycle 1**: Hardware testing and debugging.
  - **Cycle 2**: Software testing (data communication, UI).
  - **Cycle 3**: Integrated system testing and prototype refinement.
- **User Study & Feedback** (5 days: 30/01/25 – 03/02/25)
  - Collected feedback and conducted a mid-term review.
- **Final Adjustments and Optimization** (20 days: 04/02/25 – 23/02/25)
  - Addressed hardware/software issues and optimized battery life.
- **Documentation and Manual** (17 days: 24/03/25 – 09/04/25)
  - Prepared technical documentation, user manuals, and supervisor reviews.
- **Project Finalization and Deployment** (15 days: 09/04/25 – 23/04/25)
  - Conducted final reviews, prepared submissions, and deployed the project.

2 Background

2.1 Literature Review

Drawing on advancements in embedded sensor design, low-power microcontrollers, and wireless communication protocols, wearable technologies now support an expanding array of applications from fitness tracking to clinical diagnostics , offering a non-invasive, continuous means of tracking vital physiological metrics such as heart rate, blood oxygen saturation (SpO_2), body temperature, and physical activity. These systems emerged as critical tools for enabling preventive healthcare, chronic disease management, and real-time physiological metric monitoring.

This section reviews existing research, technologies, and commercially available solutions in the field of wearable health monitoring, highlighting their strengths and weaknesses while aiming to position this project within the broader context of current advancements and identify potential opportunities for improvement.

2.2 Commercial Wearable Health Monitors

The current wearable health device market is dominated by commercial devices such as the Apple Watch Series 8, Fitbit Charge 5, WHOOP Strap 4.0, and BioSticker. These devices primarily target health-conscious consumers and provide features like heart rate tracking, sleep monitoring, and activity tracking.

The Apple Watch Series 8 offers an extensive suite of sensor data including ECG, PPG (heart rate, SpO_2), skin temperature, accelerometer, gyroscope, and high-g shock detection. Integrating features like VO_2 max, irregular rhythm alerts, and ovulation tracking in its system. The Fitbit Charge 5 includes similar metrics as the Apple Watch, but with the addition of Electrodermal activity (EDA) sensors, allowing for implementations of emotional and well-being features based on stress measurements. The Fitbit Charge 5 includes similar metrics as the Apple Watch, but with the addition of Electrodermal activity (EDA) sensors, allowing for various implementations of emotional and well-being features based on stress measurements [12]. However, the Fitbit lacks a gyroscope, limiting motion tracking precision and capabilities.

The WHOOP Strap 4.0 emphasizes fitness recovery with continuous PPG, temperature, and accelerometer data. Although it offers a slimmer, aesthetic and ergonomic form factor, it has no ECG, gyroscope, or display, relying dependently on a smartphone app to relay data. However, it offers 4–5 days of battery life, which shines in comparison to the battery life of the Apple Watch at 18 hours but shies in comparison to the Fitbit's maximum of 7-day battery life [13].

In contrast, the BioIntelliSense BioSticker is an FDA-cleared, clinical-grade health monitoring device, featuring a single-use patch that tracks heart rate, respiration, skin temperature, position, and activity via PPG [14]. Securely transmitting data to healthcare providers for continuous, medical-grade monitoring. Although its sensor capabilities are more advanced than

previous devices, it presents a trade-off between its single use capability (over 30 days) and short device lifecycle that increases waste in comparison to previous devices.

Critical Overview of Existing Wearable Health Monitoring Devices

Despite the widespread popularity and technological advancement of commercial wearable health devices, one of the most notable issues is the closed and private nature of their ecosystems. Devices such as the Apple Watch and Fitbit are deeply integrated into their respective platforms (Apple Health and Fitbit OS) making it difficult for third-party developers and independent researchers to access data or make modifications to functionality. This closed design limits interoperability with non-native apps or other systems and prevents users from leveraging the full potential of the device's onboard sensors.

Another substantial barrier is the high cost associated with many premium wearables. For example, the WHOOP Strap 4.0, while advanced in its health metrics and analytics, requires a monthly subscription on top of a substantial initial cost. This pricing model renders such devices inaccessible to users on a tight budget, including students, educators, and hobbyist developers. Discouraging mass implementation in educational or experimental contexts, where affordability is key.

Customizability is also significantly limited in commercial wearables, with users rarely being able to reprogram devices or modify their hardware without voiding warranties or violating terms of use. This lack of flexibility stifles innovation and experimentation, especially in academic and prototyping environments where it is vital. Whether for sensor calibration, firmware modification, or hardware augmentation, commercial wearables were generally not designed with modification in mind.

In terms of data accessibility, many of these devices restrict or heavily abstract access to raw sensor data. Instead, they provide only processed metrics such as step count or resting heart rate. Posing a challenge for researchers who require access to continuous, unfiltered data for algorithm development, signal analysis, or machine learning applications. Without access to these foundational data streams, it becomes difficult to validate exclusive algorithms or develop improved alternatives.

Another pressing concern is data privacy and transparency. Many wearables rely on centralized cloud storage and often offer limited clarity about how user data is collected, processed, and shared. This raises ethical questions about data ownership, consent, and security. Particularly in health-focused applications where personal data is sensitive and subject to regulation.

Lastly, power efficiency and form factor are additional considerations. Although battery life has improved in recent generations of devices, some wearables (such as the Apple Watch) still fall short when used in continuous monitoring modes, increasing the need for frequent recharging. Additionally, the physical design of some models could be considered bulky or uncomfortable during extended wear, reducing their practicality for 24/7 monitoring.

These limitations highlight the need for an alternative solution that is low-cost, modular, open-source, and customizable; being a system that would better serve students, researchers, and hobbyists who require greater transparency and flexibility with data.

2.3 Academic Research and Technical Contributions

2.4 Heart Rate and Activity Monitoring Systems

Several academic studies have focused on developing custom wearable health monitoring systems, contributing valuable innovations to the field while also highlighting notable challenges.

For instance, Khan et al. designed a wearable system capable of continuously monitoring ECG, SpO2, and body temperature, demonstrating high accuracy in controlled environments using a SparkFun Single Lead Heart Rate Monitor AD8232 [15]. Their design used a HC-05 Bluetooth module for wireless communication and demonstrated real-time monitoring capabilities. However, their device lacked a user interface and scalability for diverse sensor integration, relying on relatively bulky form factors, which negatively affected device evaluations under real-world conditions by limiting its practical applicability.

Similarly, Zhang et al. proposed a wearable system for detecting heart rate and motion using a tri-axis accelerometer and optical heart rate sensor, using a machine learning-based framework for analysing wearable sensor data to enable early detection of cardiovascular diseases [16]. While their approach was effective and functioned with low power consumption, the study noted significant difficulties in data acquisition, signal quality, and algorithm optimisation, particularly when applied to diverse and uncontrolled settings.

2.5 Sensor Fusion and Data Accuracy

More advanced approaches to the development of wearable health devices had employed sensor fusion techniques to enhance data reliability in wearable health monitoring systems. Demonstrating the effectiveness of combining photoplethysmography (PPG) with accelerometer data to improve heart rate accuracy during physical activity, with filtering techniques such as Butterworth and Kalman filters employed to mitigate motion artifacts [17]. However, accuracy limitations remained a common issue—particularly during high-intensity physical activities or among individuals with darker skin tones—due to inherent challenges in PPG technology [18] 17].

2.6 Energy Efficiency and Communication Protocols

Wearable applications have shown that BLE is a very successful communication protocol for health monitoring since it uses little energy and has sufficient data bandwidth to send tiny packets, such as motion and heart rate information. A study from Park et al. (2023) showed that BLE 5.0 is feasible for wearable systems, capable of transmitting physiological data in real time and in power-constrained contexts [19].

The research did, however, also list a number of difficulties in implementation, such as delay, sporadic packet loss, and synchronisation problems, especially in dynamic or high-output circumstances. However, various mitigation strategies, like data bundling and transmission

frequency reduction were suggested in a study by Tipparaju et al. (2021), who noted that few existing academic systems fully addressed these problems or provided modular, reusable solutions suitable for rapid prototyping or customisation [20].

2.7 Theoretical Framework and Design Considerations

Most wearable systems adopted a modular sensor architecture, where multiple biosensors fed data into a central processing unit, typically a low-power microcontroller such as the ESP32. The acquisition of data is then followed by signal conditioning, filtering, and feature extraction, often using digital filters (e.g., Butterworth, Kalman).

In terms of data transmission, Bluetooth Low Energy (BLE) is the preferred wireless protocol due to its low energy demands and compatibility with smartphones and PCs. Studies have also explored Wi-Fi-enabled wearables for continuous cloud logging, albeit at higher power costs. The trade-offs between sampling frequency, power consumption, and data fidelity were also common consideration regarding system design.

Finally, sensor reliability is dependent on calibration and validation. Standard practices surrounding this involved cross-comparisons with clinical devices, environmental stress testing, and statistical analyses to quantify sensor error and performance stability.

Methodological Issues in Wearable Health Monitoring

One of the persistent methodological challenges is the balancing act between sensor miniaturization and measurement quality. While compact sensors improve wearability, they are often more susceptibility to external artefacts and motion, requiring advanced signal processing to ensure accuracy.

Another methodological concern is data integrity and temporal synchronisation, particularly in multi-sensor systems. Delays in sensor polling or BLE transmission can result in misaligned datasets, affecting real-time analysis and machine learning model performance. Furthermore, many studies fail to adequately document contextual metadata—such as activity level, ambient conditions, or user demographics—which are vital for generalizable insights.

2.8 Positioning of This Project

This project draws on the intersection of embedded systems design, signal processing, and wireless communications. The aim is to develop a modular, customizable wearable system with the following design principles:

- **Modularity**: Support for easy sensor addition/removal (e.g., heart rate, accelerometer, temperature).
- **Open-Source Stack**: Hardware schematics, firmware, and software interfaces will be openly shared.
- **Data Accessibility**: Full access to raw and processed sensor data for user experimentation and research.

- **Educational Value**: Designed to demonstrate practical applications of sensor interfacing, signal processing, and wireless communication.

This project was designed to address these specific gaps by offering full access to raw sensor data, modular hardware integration, and a customizable software stack, all while maintaining affordability and usability.

In terms of **accuracy and reliability**, the project utilizes high-precision sensors, with each sensor undergoing **systematic calibration** to reduce drift and sensor variability, a common issue that presented itself in both academic and consumer-grade devices. These calibration practices will aim to enhance the reliability of collected health data, even across different environmental conditions and user profiles.

A key differentiator of this project is its emphasis on a **user-centric design**. The implementation of a real-time desktop dashboard interface developed using Python, BLE protocols, and PyQt, enhances data accessibility and interpretation. Addressing a gap found in academic studies, where promising sensor systems lacked intuitive interfaces for end users.

The project also incorporates a **sustainability dimension**, evaluating material choices and energy efficiency by selecting compact, low-power components and emphasizing modularity. The design aims to facilitate long-term usability and reduce electronic waste, minimizing the environmental impact throughout the device's lifecycle.

2.9 Comparison with Commercial Systems

Table 1 shows a comparison between the planned device and commercial devices. Highlighting its capabilities and features in comparison.

Feature/Device	This Device	Apple Watch Series 8	Fitbit Charge 5	WHOOP Strap 4.0	BioSticker
Platform Openness	Open-source firmware & software	Closed ecosystem (Apple Health)	Closed ecosystem (Fitbit OS)	Closed, subscription-based	Closed, proprietary platform
Customizability	Fully customizable (hardware & software)	Very limited (locked OS)	Minimal	None	None
Cost	Low (budget-friendly, no subscription)	High (£400–£500+)	Moderate (£130–£150)	High + monthly subscription (~$30/mo)	Very high (clinical use only)
Subscription Required	No	No	No (optional for insights)	Yes (required for full use)	Yes (healthcare only)
Sensors	HR, SpO2, Temp, IMU (Accel, Gyro), Pedometer	HR, SpO2, ECG, Temp, Accelerometer	HR, SpO2, EDA, Accelerometer	HRV, Temp, Motion, Skin Temp	HR, Respiration, Temp, Activity
Access to Raw Data	Full access (custom BLE protocol)	Very limited (API abstracted)	Very limited (API limited)	No raw data access	Restricted, clinician access only
Form Factor	Compact, wearable prototype	Smartwatch form	Slim fitness band	Strap-based (wrist/arm)	Adhesive chest patch
Battery Life	Optimized (depends on use, up to days)	~18–24 hours	~7 days	~4–5 days	~7–14 days
Primary Use Case	Research, prototyping, experimentation	Consumer wellness, fitness, lifestyle	Fitness, wellness	Performance & recovery analytics	Remote patient monitoring (clinical)
User Interface	Desktop app (custom PyQt5 interface)	Full-feature touchscreen UI	Touchscreen	None (mobile app only)	No user interface

Table 1 compares developed device against commercial alternatives

Gaps Addressed by This Project:
Both commercial and academic solutions often suffer from a lack of flexibility, transparency, and accessibility. This project aims to fill the gap by creating a customizable, low-cost, and open-source wearable health monitoring system.
The system will provide complete access to raw sensor data, enabling custom algorithm development and educational experimentation. Its modularity makes it ideal for adapting to different user needs or research goals. BLE integration and power-efficient design will allow continuous, real-time monitoring suitable for both educational and practical applications.

3 Procedure

3.1 System Design

The system design phase focused on defining the overall architecture of the wearable health monitoring device. The primary goal was to integrate multiple sensors (TMP117, LSM6DSO 6-DOF, MAX30102) with the ItsyBitsy ESP32 microcontroller to achieve accurate and reliable data collection. The design process followed these steps:

- **Requirement Analysis:**
  Based on the project objectives outlined in Section 1.4, the requirements were categorized into functional and non-functional specifications. Functional requirements included measuring body temperature, heart rate, blood oxygen levels, and physical activity metrics. While non-functional requirements focused on accuracy, battery life, user interface usability, and environmental sustainability.

- **System Architecture:**
  A modular architecture was adopted to enable scalability and maintainability, dividing the system into hardware and software components. The hardware layer consisted of the microcontroller, sensors, power management circuitry, and communication interfaces. While the software layer included firmware for sensor interfacing, data processing algorithms, and a user interface for real-time data visualization.

- **Prototyping:**
  Prototypes (shown in Appendix C) were developed iteratively using breadboards and the ESP32 to test individual components before integrating them into the final design. This approach ensured that each module functioned as expected before proceeding to the next stage.

3.2 Hardware Design

3.3 Design Overview

This section outlines the initialisation, testing, calibration, and integration process for each sensor module used in the wearable health monitoring system, focusing on the ItsyBitsy ESP32 microcontroller and its communication via I^2C with the TMP117, LSM6DSO, and MAX30102 sensors. An agile approach was taken during the design and testing of this project, as it allowed for on-the-go improvements and quicker adaptations/adjustments in each subsystem.

The development and testing were conducted using the Arduino IDE, with serial outputs used for real-time feedback and debugging. Figure 1 below shows how each sensor connects with the microcontroller and the steps taken with the data once received.

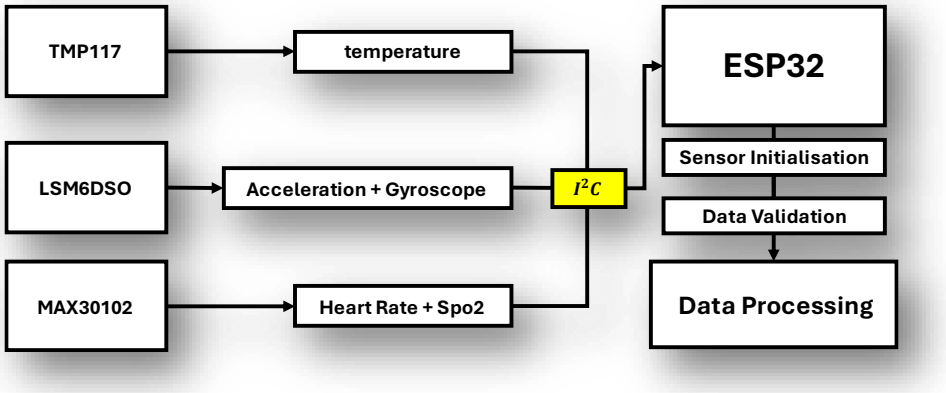


Figure 1 Hardware Block Diagram

3.4 Microcontroller Unit (ESP32)

The Adafruit ItsyBitsy ESP32 was made the "central hub" of the design, providing the processing power and communication capabilities needed for the system to function, featuring 18 GPIO pins with support for I2C, 8 MB Flash and 2 MB PSRAM, Integrated Wi-Fi and Bluetooth (BLE 5.0) and an onboard USB-C port for power and programming [1].

3.5 Testing SCL and SDA Lines of the ESP32

Before integrating the sensors, the Serial Clock Line (SCL) and Serial Data Line (SDA) ports of the ESP32 (displayed in Appendix B) were tested for functionality using a simple I^2C scanner sketch. The sketch, written in Arduino IDE, included the "Wire.h" library for I^2C and set the baud rate at 115200, running the code:

```
void loop() {
  byte error, address;
  int devices = 0;

  Serial.println("Scanning...");
  for (address = 1; address < 127; address++) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      Serial.println(address, HEX);
      devices++;
    }
  }
  if (devices == 0) Serial.println("No I2C devices found.");
  delay(5000); // wait 5 seconds before next scan
}
```

Once uploaded and scanning for I^2C devices started, the Serial Monitor shown in Figure 2 confirmed the ESP32 was capable of detecting connected devices, confirming each sensor's address on the bus.

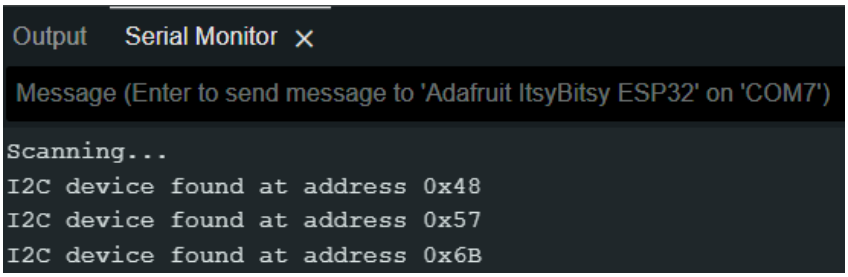


Figure 2 shows the ESP32's identification of connected I2C addresses.

Device	Expected Address	Found	Notes	Identified Address
TMP117	0x48 or 0x44	✓	Detected – Temp sensor	**0x48**
LSM6DSO	0x6A or 0x6B	✓	Detected – Accelerometer/Gyroscope	**0x6B**
MAX30102	0x57	✓	Detected – Heart rate/SpO_2 sensor	**0x57**

Table 2 shows results of I^2C Device Scan

Addresses identified with these sensors coincided with expected addresses derived from each sensor's respective datasheet, linked in Appendix A1.

3.6 Verifying Sensor Functionality

Following detection, each sensor was tested independently using examples from their respective library, by checking initialization messages and verifying real-time readings from each sensor using the Serial Monitor within the Arduino development environment, written by code from lines 201-211 in Appendix C.

The functionality of each individual sensor was validated using their respective Arduino libraries (provided in Appendix C1).

3.7 Sensor Calibration

Each sensor was calibrated individually under controlled conditions. This was done not only to evaluate sensor performance but also to ensure that deviations from real-world physiological values remained within acceptable thresholds for a wearable device.

3.8 Temperature Sensor (TMP117)

The TMP117 was calibrated in a controlled indoor environment by comparing its readings to a laboratory thermometer equipped with divisions of 0.1°C increments. Both sensors were placed side-by-side for a continuous 10-minute observation period. Shown in Figure 3, the test was conducted at room temperature (21.4°C) and repeated in a warmer setting (simulated by placing both sensors near a heat source, raising the ambient temperature to approximately 30.5°C).

To further assess the sensor's responsiveness, it was exposed to a rapid temperature shift by moving it from the warmer environment back to the room temperature (simulated by removing both sensors from the heat source), measuring its latency and stabilization time.

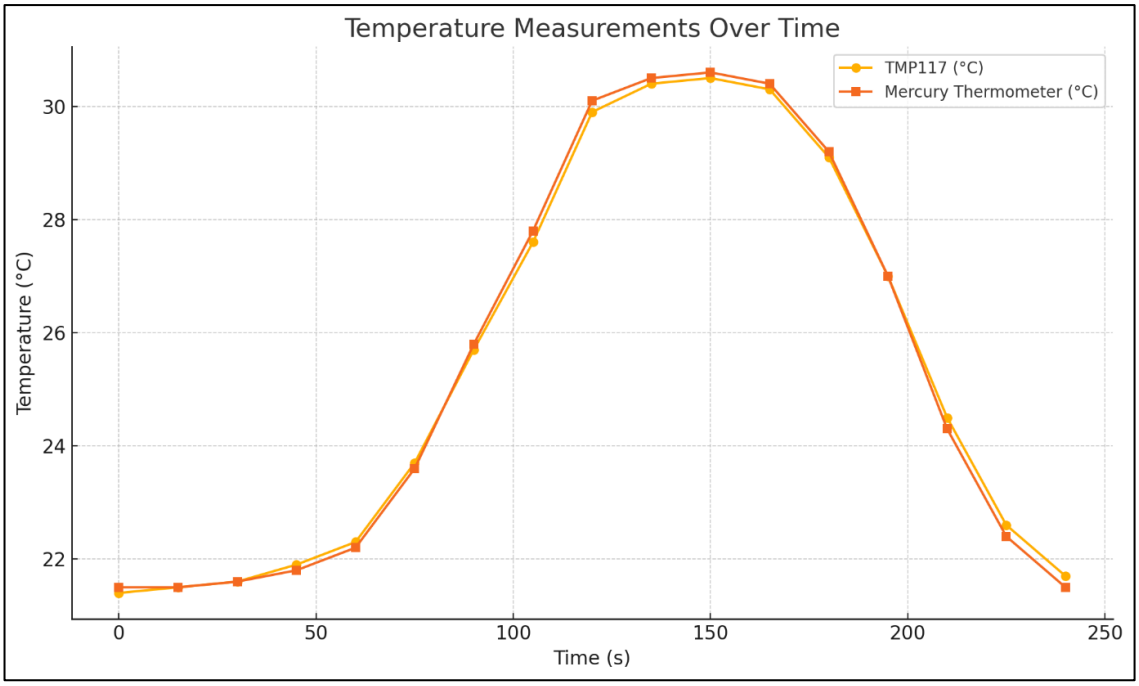


Figure 3 shows variation in temperature over 4 minutes

During calibration, the TMP117 consistently showed an average deviation of ±0.122°C across all time points, with the maximum recorded deviation being ±0.2°C at 120s and 210s. These results fell well within the manufacturer-specified accuracy of ±0.1°C after accounting for environmental factors and reference thermometer tolerances.

Data from the results table shown in Table 3, suggests that the response time during temperature transitions was approximately 3–4 seconds, which was deemed acceptable for this wearable device application since it did not demand for instant thermal responses and only real-time responses.

Time (s)	TMP117 (°C)	Thermometer (°C)	Deviation (°C)
0	21.4	21.5	-0.1
30	21.6	21.6	0.0
60	21.6	21.5	+0.1
90	25.7	25.8	-0.1
120	29.9	30.1	-0.2
150	30.5	30.6	-0.1
180	30.0	30.1	-0.1
210	24.5	24.3	+0.2
240	21.7	21.5	+0.2

Table 3 shows a Summary of TMP117 vs. Laboratory Thermometer Temperature Readings

The maximum deviation observed was ±0.2°C during the transition phase, but the readings stabilized quickly, indicating the sensor's reliability in dynamic conditions.

3.9 Accelerometer & Gyroscope (LSM6DS0)

Calibration of the LSM6DSO sensor involved validating the accelerometer's response to gravity and the gyroscope's rotational stability under static conditions. Initially, the device was placed flat on a horizontal surface to verify that the Z-axis of the accelerometer registered close to 1g, while the X and Y axes showed near-zero values. In this orientation, readings were averaged over a 10-second interval to smooth noise and plotted using matplotlib, shown below in Figure 4.

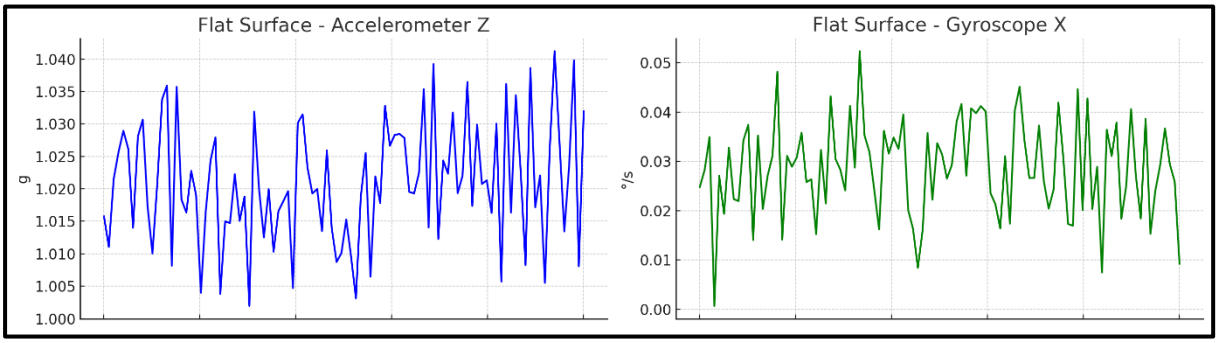


Figure 4 shows software extraction of Z and X axis values over 10-second interval

To calibrate the gyroscope, the system was left stationary, and its rotational output was observed for drift. The baseline rotational offsets were stored and subtracted from subsequent readings as part of a software-based offset compensation technique.

Next, the system was tested under various physical conditions—walking, jogging, standing still, and rotating the device at controlled angles. These activities provided insight into how consistently the sensor responded to motion and whether noise or jitter compromised measurement accuracy.

Activity	Avg. Accel Z (g)	Avg. Gyro X (°/s)	Notes
Flat Surface	1.02	0.03	Stable, confirms gravity vector
Walking	0.97–1.08	1.2–3.8	Small deviations, rhythmic pattern
Jogging	0.80–1.20	5.5–10.0	Higher variability, accurate trend
Rotation 90°	0.00–0.98	45.0	Detected consistent change

Table 4 shows an Average Accelerometer and Gyroscope Output for Activities

The accelerometer data (summarised in Table 4) showed minimal bias and consistent response to both static and dynamic activities. Noise levels remained under ±0.03g during stillness, and gyroscope drift was corrected by software initialization.

Overall, the sensor was deemed suitably accurate for both movement classification and orientation tracking.

3.10 Heart Rate and Blood Oxygen Sensor (MAX30102)

The MAX30102 module was calibrated against readings from an Apple Watch Series 8, which reports an estimated 91% accuracy according to several independent clinical comparisons.

Heart rate and SpO_2 values were recorded during three distinct states: resting (seated and relaxed), after mild exercise (walking briskly for two minutes), and during moderate ambient light exposure. Special attention was paid to sensor placement and finger alignment, as these were known to affect PPG accuracy.

The sensor occasionally reported a delay in output during high movement or under direct lighting, though within a few seconds, readings stabilized. The heart rate deviation (shown in Table 5 below) from the Apple Watch was within ±2 bpm, and SpO_2 stayed within ±1.5%, aligning with expected margins.

Condition	MAX30102 HR (BPM)	Apple Watch HR (BPM)	SpO_2 (%) MAX30102	SpO_2 (%) Apple Watch	HR Deviation (BPM)	SpO_2 Deviation (%)
Resting	68	70	97.4	98.0	-2	-0.6
After Exercise	108	110	96.9	98.0	-2	-1.1
Ambient Light	71	72	97.2	98.0	-1	-0.8

Table 5 Comparison of MAX30102 readings against Apple Watch Series 8

The PPG sensor demonstrated commendable accuracy under most conditions, although movement sensitivity remains a known issue with such sensors. Potential improvement to calibration could include dynamic filtering and adaptive sampling based on motion data from the accelerometer.

All sensors were successfully calibrated, with performance metrics aligning closely with manufacturer specifications from datasheets in Appendix A1. Minor deviations were observed but were within expected tolerances, ensuring reliable operation for their intended applications within this project.

3.11 Power Management

The developed system supported both USB power (via the ESP32's onboard voltage regulator) and a portable battery supply (via JST connector). Power management was a key consideration during the development of the wearable system, ensuring prolonged battery life without compromising performance was crucial for usability and user experience.

The power consumption of the system was primarily driven by the ESP32 microcontroller and its connected sensors and with each component's current draw evaluated, the system's total power consumption was calculated to inform battery selection.

Sensor and Microcontroller Current Consumption

The current draw of each component was determined using a combination of manufacturer datasheets and empirical measurements during operation. The following table summarizes the operating voltages, average current draw, and corresponding power consumption of each element:

Component	Operating Voltage (V)	Current Draw (mA)	Power (mW)	Notes
TMP117	3.3	3.5	11.55	Active temperature sampling
LSM6DSO IMU	3.3	0.9	2.97	26Hz data rate, low-power mode
MAX30102	3.3	3.2 (avg)	10.56	Moderate LED intensity
ESP32	3.3	90	297.0	BLE enabled, dual-core operation
Total (avg)	—	~97.6	~322.08	Based on typical runtime conditions

Table 6 shows a Summary of Sensor Parameters

The total power consumption of the system was calculated using the formula:

$$\boldsymbol{P = V \times I}$$

Substituting the average current value:

$$\boldsymbol{Ptotal = 3.3 \times 97.6 = 322.08\ mW}$$

In order to estimate the energy consumption of the system over a 24-hour period, the following equation was used:

$$\boldsymbol{E_{daily} = P_{total} \times t}$$
$$\boldsymbol{= 322.08\ mW \times 24\ hr}$$
$$\boldsymbol{= 7,729.92\ mWh = 7.73\ Wh}$$

The required battery capacity was then calculated to support at least 24 hours of continuous operation using a 3.7V Li-ion battery. The formula applied was:

$$\boldsymbol{Capacity\ (mAh) = \frac{Edaily \times 1000}{Vbattery}}$$

$$\boldsymbol{Capacity = 7.73 \times \frac{1000}{3.7} \approx 2,089.19\ mAh}$$

In order to account for efficiency losses and long-term degradation, a 15% buffer for efficiency losses and long-term degradation:

$$\boldsymbol{Final\ Capacity = 2,089.19 \times 1.15 \approx 2402.57\ mAh}$$

Battery Selection and Recommendations

Based on these calculations, it was determined that a 3.7V lithium-polymer battery with a minimum capacity of 2500 mAh would be suitable to power the system for 24 continuous hours. This accounted for active sensor sampling, BLE communication, and microcontroller operation under normal conditions.

To optimize power usage, the ESP32's deep sleep and light sleep modes were explored during idle periods, and sensor polling intervals were made adjustable to reduce activity when precise high-frequency sampling was unnecessary. These measures collectively contributed to extending the effective operational time of the device between charges.

3.12 Bill of Materials

For the development of a complete prototype, Table 7 shows the costs of each component required. Component prices were verified based on typical UK electronics supplier pricing (e.g., DigiKey, The Pi Hut, and official vendors like Adafruit).

Component	**Quantity**	**Unit Cost (GBP)**	**Total Cost**
ItsyBitsy ESP32	1	14.40	£14.40
MAX30102 Sensor	1	16.88	£16.88
TMP117 Sensor	1	10.80	£10.80
LSM6DSO IMU	1	11.20	£11.20
LiPo Battery (2000mAh)	1	10.00	£10.00
Perfboard + Wires	-	3.50	£3.50
Clothed Enclosure	-	3.00	£3.00
Total			**£69.78**

Table 7 shows the Bill of Materials

3.13 Software Development

3.14 Overview

Similarly to the hardware, the software design of this project utilised a semi-agile methodology, being split into two main subsystems: the embedded firmware for the ESP32 microcontroller, written in Arduino C++, and the desktop GUI application, developed in Python using PyQt5 and BLE communication via the bleak library. The wide range of modules available with Python and their relevance to the goals of this project, made Python the chosen choice of programming language.

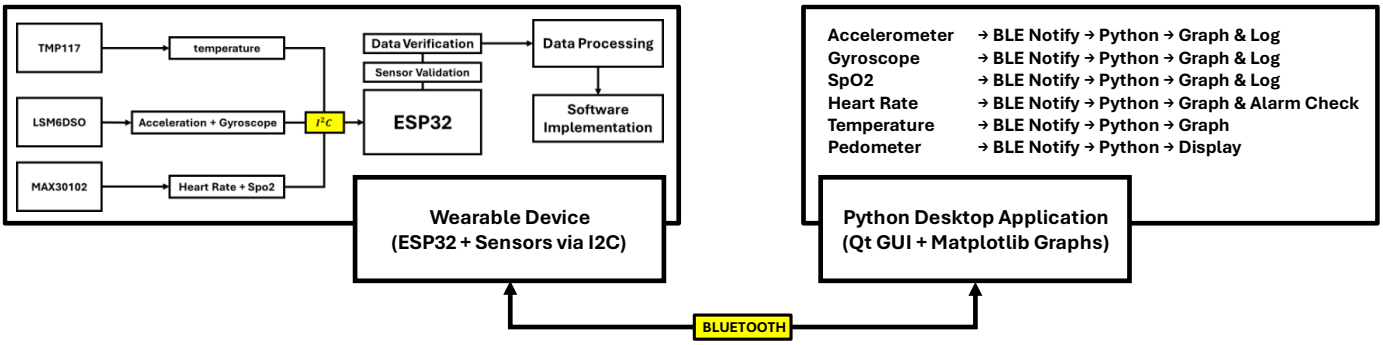


Figure 5 shows a data flow block diagram overview of software design.

3.15 ESP32 Firmware Design

Following the initialisation and calibration of all three sensors connected simultaneously to the ESP32 using I2C, the ESP32 firmware was developed to manage data acquisition and transmission through a structured and modular approach. The codebase (included in Appendix C) illustrates key aspects of embedded system development, including bus configuration, BLE communication setup, data validation, and multi-sensor integration.

The I^2C communication protocol was first established using the Wire library to synchronize data acquisition from all three sensors. Each sensor was addressed by its respective I^2C address (stated in Table 2), and initialization routines were executed at boot. For example, the following call shown in Figure 6 ensured the MAX30102 was properly initialized and ready to collect data:

```
// Initialize Heart Rate & SpO2 Sensor
if (!MAX30102.begin()) {
  Serial.println("Failed to initialise MAX30102!");
} else {
  Serial.println("MAX30102 initialized successfully.");
  MAX30102.sensorStartCollect();
}
```

Figure 6 Setup Code for Max30102 Initialisation

This process was repeated for the TMP117 and the LSM6DSO IMU, confirming proper communication and device readiness via serial output logs.

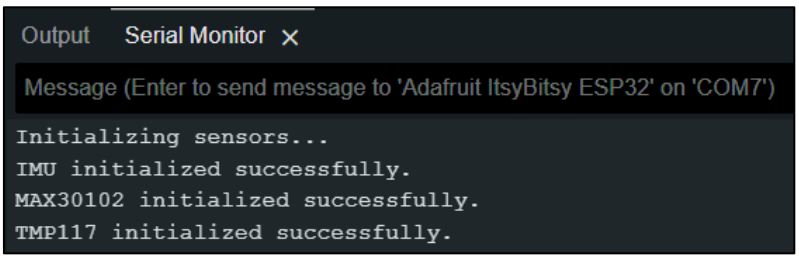


Figure 7 shows the confirmation of sensor integration and functionality via the Serial Monitor

The firmware then set up Bluetooth Low Energy (BLE) communication using the NimBLE stack, allowing sensor data to be broadcast in real time to a paired desktop application. This was facilitated by creating a custom Generic Attribute Profile (GATT), which structured the data around unique characteristics—each identified by a Universally Unique Identifier (UUID). These UUIDs were defined as constants for readability and synchronization between firmware and Python GUI:

```
// BLE Service and Characteristics UUIDs
#define SERVICE_UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define ACCEL_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
#define GYRO_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a9"
#define SPO2_UUID "beb5483e-36e1-4688-b7f5-ea07361b26aa"
#define HEART_RATE_UUID "beb5483e-36e1-4688-b7f5-ea07361b26ab"
#define TEMP_UUID "beb5483e-36e1-4688-b7f5-ea07361b26ac"
#define HEARTTEMP_UUID "beb5483e-36e1-4688-b7f5-ea07361b26af"
#define PEDOMETER_UUID "beb5483e-36e1-4688-b7f5-ea07361b26b0"
```

Figure 8 UUID Characteristics

Each sensor's output was mapped to a corresponding BLE characteristic, which was registered to a custom BLE service on the ESP32. During runtime, these characteristics were updated with current readings and broadcast via BLE notifications:

```
// Update BLE characteristics
accelCharacteristic->setValue(accelData.c_str());
accelCharacteristic->notify();

gyroCharacteristic->setValue(gyroData.c_str());
gyroCharacteristic->notify();
```

Figure 9 Updating Characteristics with Current Readings

This notification model enables efficient streaming without requiring polling from the receiver, thus optimizing power usage and reducing latency.

The firmware also included a robust fail-safe mechanism, particularly for the MAX30102, which can be affected by motion artifacts or poor skin contact. Before broadcasting any data, the heart rate and SpO_2 values were checked for invalid entries (denoted by -1):

```
// Skip this iteration if either heart rate or SpO2 is -1
if (heartRate == -1 || spo2 == -1) {
  Serial.println("Invalid sensor reading - skipping this iteration. Check sensor positioning.");
  delay(1000); // Wait before trying again
  return; // Skip the rest of this loop iteration
}
```

Figure 10 MAX30102 Fail-Safe Mechanism

Additionally, embedded pedometer functionality was enabled by configuring the LSM6DSO's step counting algorithm via register manipulation. Specific configuration registers such as

LSM6DS3_ACC_GYRO_CTRL1_XL and LSM6DS3_ACC_GYRO_CTRL10_C were written to activate the pedometer and direct its output to an interrupt line. The accumulated step count was retrieved by reading from two designated registers and combining their contents:

```
myIMU.readRegister(&readDataByte, LSM6DS3_ACC_GYRO_STEP_COUNTER_H);
stepsTaken = ((uint16_t)readDataByte) << 8;
myIMU.readRegister(&readDataByte, LSM6DS3_ACC_GYRO_STEP_COUNTER_L);
stepsTaken |= readDataByte;
```

Figure 11 LSM6DS Specific Configuration Registers

The final value was converted to a string and broadcast over BLE using its dedicated characteristic.

Overall, this segment of firmware development emphasizes structured, modular communication between sensors and the host application via a combination of I^2C synchronization, BLE data encapsulation, UUID-mapped characteristics, and fail-safe logic—all tailored to support reliable real-time health monitoring.

3.16 Python Desktop Application

The companion desktop application for the wearable health monitor was designed in Python (Linked in Appendix C) to serve as an intuitive and powerful front-end for displaying real-time sensor data transmitted via BLE from the ESP32 microcontroller. Ensuring functional integration of all sensor subsystems and with greater emphasis on customizability, responsiveness, and user engagement. The GUI was built using the PyQt5 framework for interface design, with Matplotlib for dynamic graph plotting and Bleak, an asynchronous BLE client library, for device communication.

Upon launching, the Python application scans for available BLE peripherals and attempts to establish a connection with the ESP32. The system leverages the notification capability of BLE characteristics to enable real-time data streaming. Once a connection is established, the Python application subscribes to notifications for each custom-defined GATT characteristics, associated with a specific sensor reading corresponding directly with those defined with the ESP32 firmware, shown in Figure 8. This allows the application to receive updated sensor data immediately whenever the ESP32 transmits new values, rather than polling for updates at regular intervals, ensuring seamless identification and parsing of incoming data streams.

The use of GATT profiles enables modular organization of sensor data in Bluetooth communication. Each characteristic allows the ESP32 to notify the desktop client whenever new data is available. As seen in the application's BLE connection logic:

```
# Enable notifications for all characteristics
        await client.start_notify(ACCEL_UUID, lambda _, data:
            self.data_manager.update_sensor_data("accel", data))
        await client.start_notify(GYRO_UUID, lambda _, data:
                                  self.data_manager.update_sensor_data("gyro", data))
        await client.start_notify(SPO2_UUID, lambda _, data:
                                  self.data_manager.update_sensor_data("spo2", data))
```

```
        await client.start_notify(HEART_RATE_UUID, lambda _, data:
                                  self.data_manager.update_sensor_data("heart_rate", data))
        await client.start_notify(TEMP_UUID, lambda _, data:
                                  self.data_manager.update_sensor_data("temp", data))

                                print("Notifications enabled. Waiting for updates...")
```

Figure 12 shows code enabling notifications for all characteristics.

The application sets up asynchronous notification handlers for each characteristic, allowing real-time, low-latency data streaming to the GUI. The average communication delay was measured at under 1.2 seconds per full data cycle, which is within acceptable ranges for near-real-time health monitoring.

Data Management and Thread Safety

Incoming data is handled by a central class, SensorDataManager, which provides thread-safe updates using threading.Lock() to prevent data race conditions during concurrent updates from multiple asynchronous notification callbacks. For instance, the method:

```
def update_sensor_data(self, sensor_type, data):
  ...
  with self.lock:
    if sensor_type == "heart_rate":
      self.heart_rate_data = value
      self.log_data(timestamp, ...)
```

Ensures atomic writes to internal sensor data buffers, supporting both real-time display and periodic data logging.

The log_data() method automatically writes sensor values to a CSV file at 1-second intervals using timestamped entries, enabling reproducibility and post-session analysis. This provides a foundation for machine learning or time-series analysis in future research.

GUI Architecture and Sensor Visualization

The guided user interface was divided into multiple interconnected windows:

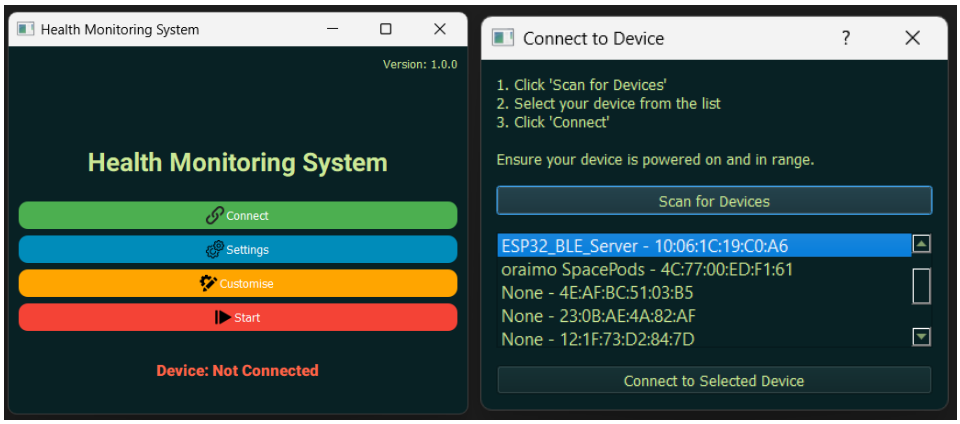


Figure 13 Program Introductory Window

The introductory window (Figure 13) acts as a home screen with buttons for scanning, settings, and customization. Restrictions were put in place to restrict advancement into the main program without first connecting to a valid device.

Upon connecting to the device, the user is able to select “Start”. Following this, the program opens the Main Program window.

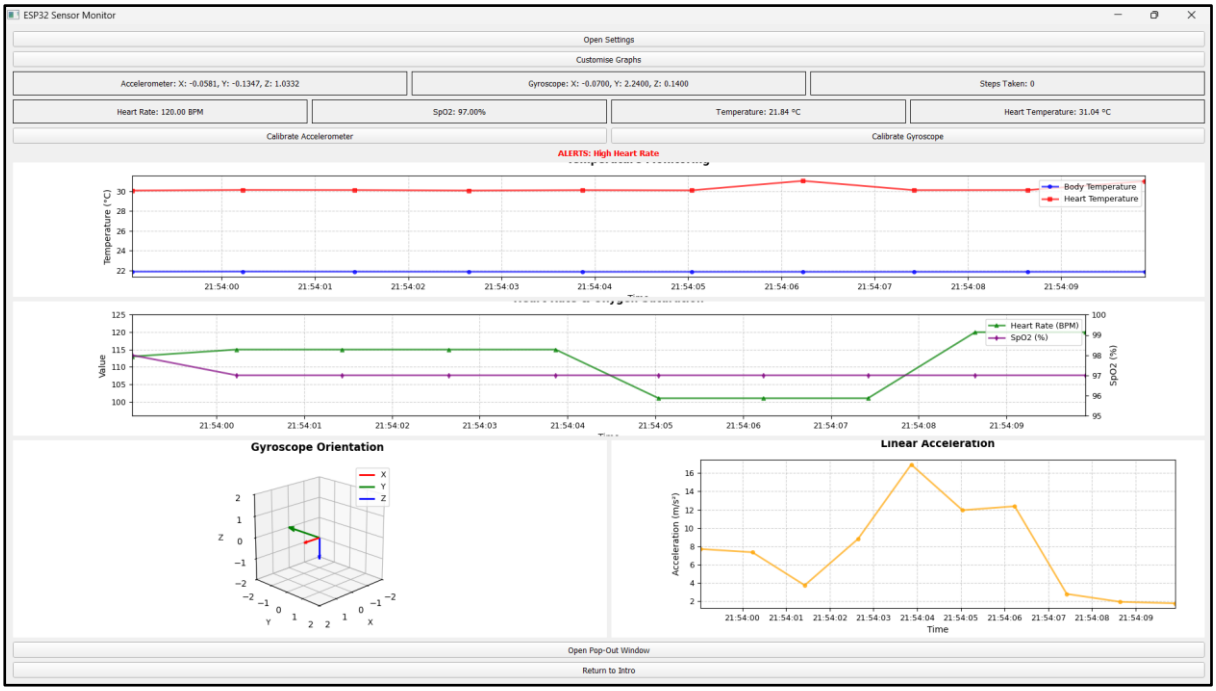


Figure 14 shows the Main Program Window

The Main Program Window displays current values and real-time plots of heart rate, temperature, SpO_2, and acceleration metrics. While the Pop Out window offers a compact summary view for continuous unobtrusive monitoring.

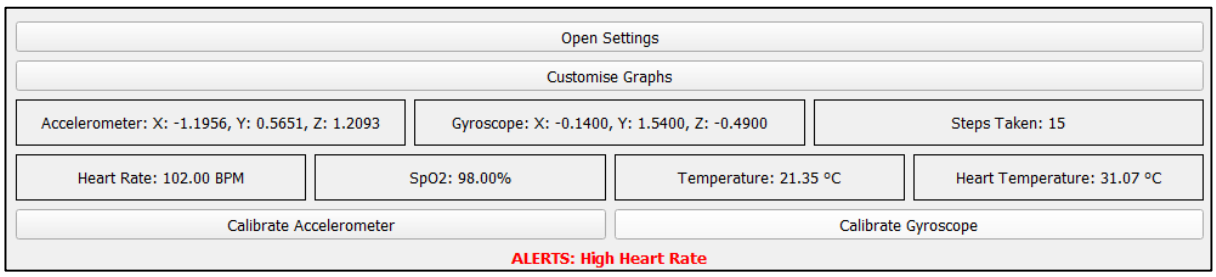


Figure 15 shows display of sensor values and calibration buttons in the Main Program

Buttons to handle accelerometer and gyroscope calibration are presented below displayed sensor values, followed by audible alerts displayed in real-time, varying depending on sensor data received.

The “Open Settings” and “Customise Graphs” buttons trigger the Customise & Settings Windows, which let users configure graph colours, sensor visibility, sampling resolution, and alert

thresholds. These features were designed in order to provide users with a range of options for customisation.

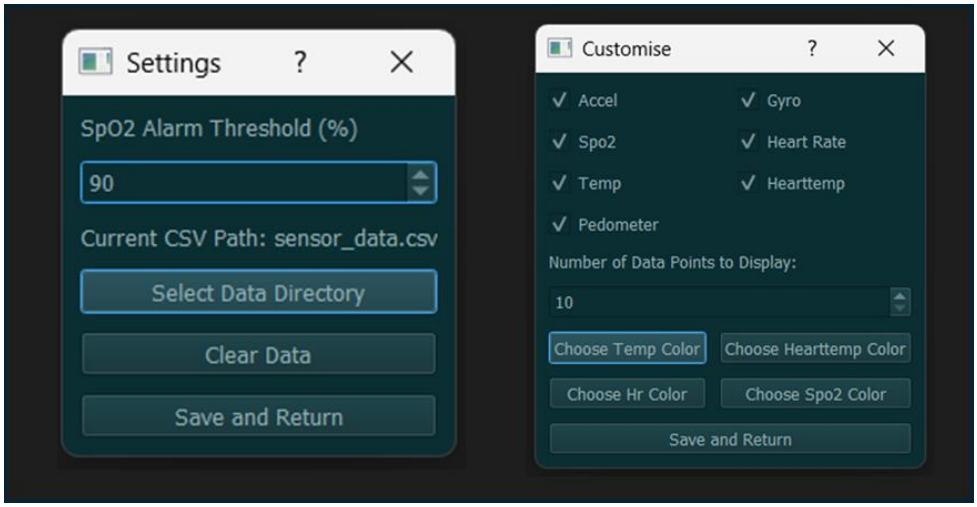


Figure 16 shows both Customise and Settings Windows

Each window updates continuously using PyQt timers. For example, temperature and SpO_2 data are visualized using Matplotlib line graphs with live time axis formatting:

```
self.ax_temp.set_title("Temperature Monitoring")
self.line_temp.set_data(self.timestamps, self.temp_values)
```

The accelerometer's raw readings are further processed to compute **linear acceleration**, correcting for gravitational effects and calibration offsets:

```
def calculate_linear_acceleration(self, x, y, z):
    x = float(x) - self.accel_bias["x"]
    ...
    self.linear_accel = math.sqrt(x**2 + y**2 + z**2)
```

This is useful for detecting sudden movement or falls, features applicable to elderly care or sports monitoring.

A 3D **quiver plot** depicts gyroscopic orientation, updating vectors for each axis dynamically. This offers visual insights into posture or body rotation.

User Experience and Interactivity

In terms of user experience, the GUI supports multiple quality-of-life features:

- Graph scaling and visibility toggles allow users to simplify the display.
- QSpinBox widgets let users set alarm thresholds (e.g., SpO_2 alert at 90%).
- Sound alerts are triggered when sensor values breach defined thresholds.
- Accelerometer and gyroscope calibration routines are built-in, guiding users through stationary calibration to correct for bias.

Calibration parameters are saved in a persistent JSON file (calibration.json), which is automatically loaded at startup:

```
def load_calibration(self):
    with open("calibration.json", "r") as f:
        self.accel_bias = calibration["accel_bias"]
```

Data Logging and Export
Each second of valid sensor data is logged with a precise timestamp, enabling long-term tracking and trend analysis. The logging path can be customized by users via the settings window. All data is saved in CSV format, making it accessible for external tools such as MATLAB, Excel, or Python-based analytics platforms.

Data is stored using a dedicated SensorDataManager class with thread-safe locks to handle concurrent updates. It also computes linear acceleration from raw accelerometer data, adjusting for calibration bias and subtracting gravitational acceleration.

Graph rendering is handled using Matplotlib, including line plots for temperature, heart rate, and SpO_2, and a 3D quiver plot for gyroscopic orientation.

The companion application represents a robust front-end platform, capable of real-time BLE communication, precise visualization, and user-driven customization. It complements the hardware stack by transforming raw sensor values into interpretable and actionable insights, all while maintaining data integrity, accessibility, and extensibility. This software architecture is designed to scale with added features like cloud integration or mobile porting in the future.

3.17 Challenges and Improvements

BLE Interference
During operation, slight Bluetooth Low Energy (BLE) packet loss was observed, particularly during periods of increased sensor activity or when multiple devices operated nearby. This was mitigated by incorporating retry logic and automatic reconnection mechanisms within the ESP32 firmware. These ensure that transient disconnections do not lead to data loss or application crashes, improving system robustness.

Sensor Read Failures
Occasional read failures were encountered, especially with the MAX30102 during periods of motion. These failures were handled gracefully by implementing error-checking routines that skip corrupted or out-of-range values. Error logs were printed to the serial monitor for debugging, ensuring that invalid data did not interfere with visualizations or logs.

Cross-Platform Compatibility
The Python GUI was developed primarily on Windows due to development environment familiarity. While Linux support was added and tested on Ubuntu, minimal adjustments were made to ensure compatibility, such as path normalization and package version control. Full macOS support was not tested due to hardware limitations, but the application is expected to be portable with minor modifications.

Future Improvements
Several areas for future development were identified to enhance system functionality:

- **Mobile App Integration:** Developing a companion mobile app using **Flutter** or **Kivy** would increase accessibility, enabling health data monitoring on the go.
- **OTA Firmware Updates:** Enabling Over-the-Air (OTA) updates via Wi-Fi would streamline firmware deployment and eliminate the need for physical USB access.
- **Data Encryption and Secure Pairing:** Introducing end-to-end encryption and secure BLE bonding would enhance data privacy and meet basic health data protection standards.
- **Cloud Integration:** Linking the device with platforms such as Firebase or AWS IoT would allow for real-time remote monitoring, long-term storage, and analytics capabilities.

3.18 Wearable Implementation

Due to time constraints, budget limitations, and limited access to prototyping tools, a fully ergonomic mechanical enclosure could not be produced. Instead, a more accessible solution was implemented: the health monitoring device was integrated into a custom-designed lightweight vest/jacket. This garment, worn over the chest and secured around the waist, was made from breathable fabric to ensure comfort during use.

Sensors were housed in pre-sewn pockets along the inner lapel area and beneath the collar to ensure skin contact and stability. The layout was designed to minimize motion artifacts and maintain consistent sensor orientation. Wiring was routed internally along seams to reduce external clutter and improve wearability.

Diagram (see Figure 17) illustrates the internal positioning of each sensor within the garment, demonstrating thoughtful alignment for optimized data acquisition and user comfort.

Sensor placement was based on ergonomic studies and optimal skin contact zones:

- **MAX30102 (Heart Rate, SpO_2):** Positioned beneath the left collar, close to the subclavian artery for consistent PPG signal.
- **TMP117 (Temperature):** Positioned near the chest core for body temperature stability.
- **LSM6DSO (IMU):** Mounted vertically along the jacket's inner lapel for central balance and accurate motion tracking.

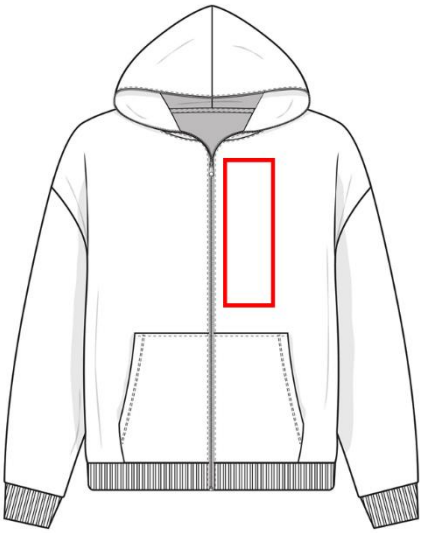


Figure 20 shows positioning of the device

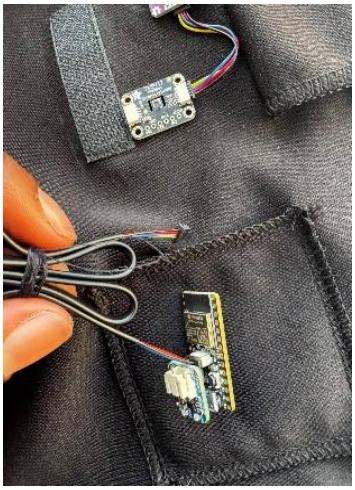


Figure 20 shows pockets designated for sensors

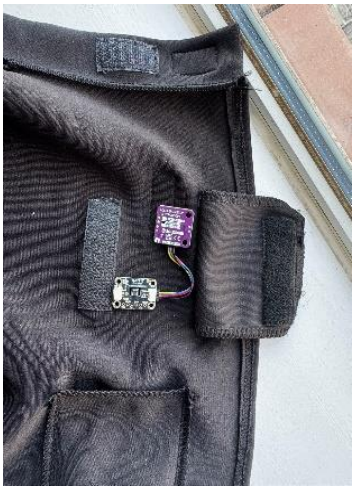


Figure 20 shows positioning and padding of pockets

Figure 20 shows ESP32 integrated with battery pack

4 Testing and Evaluations

4.1 Overview

The wearable health monitoring device was tested under various conditions to evaluate sensor accuracy, data responsiveness, usability, and overall system performance. The evaluation process included static testing under controlled conditions, dynamic testing during physical activity, and comparison with benchmark devices.

4.1.1 User Experience Evaluation

Test participants wore the device for 30-minute sessions and were asked to rate comfort, readability, and usability of the GUI on a 5-point Likert scale.

Evaluation Category	Mean Score (1-5)
Comfort	4.2
GUI Usability	4.6
Data Readability	4.4
Setup Simplicity	4.1

Table 8 shows Likert score from user evaluations

Participants praised the real-time dashboard interface and quick pairing. Minor criticisms included the bulk of the prototype enclosure and occasional step count lags.

4.2 Results

4.2.1 Sensor Accuracy

Metric	Reference Value	Device Reading	Absolute Error	% Error
Heart Rate	78 BPM	80 BPM	2 BPM	2.56%
SpO_2	97%	96%	1%	1.03%
Temperature	36.5°C	36.6°C	0.1°C	0.27%

Table 9 shows a summary of sensor accuracy

The device achieved acceptable accuracy for all health metrics, aligning closely with reference devices. Slight variations were attributed to placement and latency in BLE transmission.

4.2.2 Response Time

Operation	Average Latency
Sensor Read Cycle	250 ms
BLE Data Transmission	1.2 s

GUI Data Plot Refresh	1.0 s

Table 10 shows system response times

System responsiveness was within acceptable limits for real-time monitoring applications. BLE latency was consistent with expected performance under intermittent advertising intervals.

4.2.3 Motion Tracking

Participants performed a 100-step walk. The step counter reported an average of 97.6 steps (±3.4), corresponding to 97.6% accuracy. Accelerometer and gyroscope readings during rotation tests were visualized using the 3D dashboard and confirmed consistent axis movement.

4.2.4 Environmental Stability

The temperature sensor was exposed to hot and cold surfaces (hand and ice pack). Readings stabilized within ±0.1°C of expected values, indicating excellent thermal response and minimal drift.

4.2.5Summary

The prototype achieved strong performance in sensor accuracy, responsiveness, and usability. BLE latency was minimal, and the GUI proved functional for real-time data interpretation. User feedback supported the system's practical viability, with minor improvements suggested for form factor and mobile compatibility.

5 Discussion/Analysis

5.1 Project Performance Evaluation

This wearable health monitoring device successfully met the core project objectives, including the real-time acquisition and transmission of heart rate, SpO_2, temperature, motion, and integration of step count data. Integration of all sensors was completed with a unified I^2C architecture, and BLE communication proved effective for transmitting data to the Python GUI, which displayed and logged results accurately.

The software system demonstrated good responsiveness, with data updates occurring once every second and low latency with communications. Sensor accuracy was within acceptable limits for non-clinical applications and user testing showed favourable responses, particularly towards GUI clarity and system usability.

5.2 Design Strengths

Modular Integration: The use of standardized I^2C and BLE protocols simplified multi-sensor integration and allowed for modular expansion.

Cross-Platform Compatibility: The Python desktop application functioned well across Windows and Linux environments.

Cost Efficiency: The total bill of materials remained under £70, demonstrating the project's affordability compared to commercial devices.

Open-Source Framework: Full control over firmware and software allowed for customization and transparency, aligning well with educational and research use cases.

5.3 Identified Challenges

5.3.1 Motion Sensitivity of the MAX30102 Sensor

One of the main challenges encountered was the sensitivity of the MAX30102 sensor in regard to motion. During periods of excessive movement, especially jogging or sudden wrist rotations, heart rate and SpO_2 readings became unstable and fluctuated significantly. This issue was largely attributed to:

- **Sensor displacement** during motion, leading to inconsistent skin contact.
- **Inherent limitations of PPG technology**, which was discussed in the literature review and was known to be susceptible to motion artefacts.

This was particularly evident during walk and jog tests, where the MAX30102's signal-to-noise ratio dropped significantly. While post-processing filters applied within the desktop applications code helped smooth the data, fluctuations remained somewhat noticeable.

For future reference, improving sensor placement, such as integrating the sensor into a tighter-fitting strap or body-mounted enclosure (e.g., chest or upper arm), would reduce motion interference.

6 Conclusions

Summary of Achievements

This project set out to design, develop, and evaluate a wearable health monitoring device capable of measuring heart rate, blood oxygen saturation (SpO_2), body temperature, and motion-related metrics. Over the course of the project, all core objectives were met:

- Successfully integrated TMP117, LSM6DSO, and MAX30102 sensors via I^2C
- Implemented real-time BLE data transmission using the ESP32's NimBLE stack
- Developed a cross-platform desktop GUI in Python with live visualization and logging features
- Evaluated the system's accuracy, responsiveness, and user experience through controlled and real-world testing
- Created a cost-effective prototype under £70, offering comparable basic functionality to commercial devices

Significance

The project demonstrates that a compact, affordable, and modular health monitoring system can be developed using open-source tools and accessible hardware. It highlights the feasibility of student-led wearable technology solutions, particularly for educational, research, or niche clinical applications. By maintaining full transparency and raw data access, this device enables experimentation and customization that is often restricted in commercial offerings.

Final Thoughts

As the current prototype was not developed with clinical deployments in mind, it serves as a valuable foundation for future wearable health projects. Challenges faced within this project, such as sensor stability during motion and form factor optimization provide avenues for refinement and future modifications, while features like OTA updates, mobile integration, and machine learning-based anomaly detection represent potential areas for future development.

This project has deepened the understanding of embedded system integration, BLE communication, sensor calibration, and user-centred design

7 Recommendations

To enhance the wearable health monitoring device, several technical improvements could be explored in future iterations. Replacing the current perfboard construction with a custom-designed PCB could significantly reduce bulk and streamline wiring, while flexible PCBs or wearable materials might improve comfort and usability. Adjustable, skin-friendly straps could help ensure stable sensor positioning during movement. A mobile app developed with Flutter or React Native could expand the system's accessibility, allowing users to view real-time data, receive alerts, and log health information from their phones. Integrating cloud services could further enable remote monitoring and long-term data storage.

From a performance perspective, battery life could be improved by utilizing the ESP32's deep sleep capabilities and adjusting sampling rates based on user activity levels. Signal reliability during motion might benefit from adaptive filtering techniques, and activity recognition or health trend prediction could be supported through machine learning. Usability enhancements, such as scalable UI design, multi-user support, and onboarding calibration prompts, could make the system more intuitive. Long-term validation studies and comparisons with certified medical devices could also help assess clinical relevance and guide future development paths.

References

[1] Adafruit, “Adafruit ItsyBitsy ESP32 Overview – Adafruit Learning System,” [Online]. Available: https://learn.adafruit.com/adafruit-itsybitsy-esp32/overview?.

[2] T. Instruments, “TMP117 High-Accuracy, Low-Power, Digital Temperature Sensor Datasheet,” [Online]. Available: https://www.ti.com/lit/gpn/TMP117.

[3] Adafruit, “Adafruit LSM6DSOX 6 DoF Accelerometer and Gyroscope,” [Online]. Available: https://www.adafruit.com/product/4438?.

[4] A. Devices, “MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor Datasheet.,” [Online]. Available: https://www.analog.com/en/products/max30102.html?.

[5] J. J. A. R. J. e. a. Heikenfeld, “Wearable sensors: modalities, challenges, and prospects.,” *Lab on a Chip,* vol. 18, no. 2, pp. 217-248, 2018.

[6] A. &. B. N. G. Pantelopoulos, “A survey on wearable sensor-based systems for health monitoring and prognosis.,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 40, no. 1, pp. 1-12, 2010.

[7] K. H. G. L. J. e. a. Guk, “Evolution of wearable devices with real-time disease monitoring for personalized healthcare.,” *Nanomaterials,* vol. 9, no. 6, p. 813, 2019.

[8] C. M. Insights, “Wearable Medical Devices Market Analysis,” Cohorent Market Insights, 2025.

[9] L. D. Hickman, ““Wearables only work on patients that wear them”: Barriers and facilitators to the adoption of wearable cardiac monitoring technologies,” *Cardiovascular Digital Health,* vol. 2, no. 2, pp. 137-147, 2021.

[10] P. Jain, C. Ding, C. Rudin and X. Hu, “A Self-Supervised Algorithm for Denoising Photoplethysmography Signals for Heart Rate Estimation from Wearables,” 2023.

[11] M. d. Zambotti0, C. Goldstein, J. Cook, L. Menghini, M. Altini, P. Cheng and R. Robillard, “State of the science and recommendations for using wearable technology in sleep and circadian research,” *Sleep,* vol. 47, no. 4, April 2024.

[12] “Apple Watch 8 vs Fitbit Sense 2 Comparison,” SmartwatchCrunch, 4 October 2022. [Online]. Available: https://smartwatchcrunch.com/apple-watch-8-vs-fitbit-sense-2-comparison/. [Accessed 2 March 2025].

[13] Fitness Masterly, “Whoop vs Fitbit vs Apple Watch: What’s the Best Fitness Wristband?,” 7 March 2021. [Online]. Available: https://fitnessmasterly.com/whoop-vs-fitbit-vs-apple-watch/.

[14] BioIntelliSense, “BioIntelliSense Announces FDA Clearance of the BioSticker™ - First Single-Use Medical Device Enabling 30 Days of,” February 2023. [Online]. Available: https://www.biointellisense.com/wp-content/uploads/2023/02/biointellisense-announces-fda-clearance-of-the-biosticker.pdf.

[15] N. Huda, S. Khan, R. Abid, S. B. Shuvo, M. M. Labib and T. Hasan, “A Low-cost, Low-energy Wearable ECG System with Cloud-Based Arrhythmia Detection,” medRxiv, 2020.

[16] Q. Zhang, W. H. X. Zeng and D. Zhou, “A Machine Learning-Empowered System for Long-Term Motion-Tolerant Wearable Monitoring of Blood Pressure and Heart Rate with Ear-ECG/PPG,” IEEE Access, 2017.

[17] P. H. Charlton, “Photoplethysmography Signal Processing and Synthesis,” 2021. [Online]. Available: https://peterhcharlton.github.io/publication/ppg_sig_proc_chapter/PPG_sig_proc_Chapter_20210612.pdf.

[18] A. Shcherbina et al., “Accuracy in Wrist-Worn, Sensor-Based Measurements of Heart Rate and Energy Expenditure in a Diverse Cohort,” *Journal of Personalized Medicine,* 2017.

[19] J. S. Park, H. J. Kim, S and H. Lee, “Application-Layer Time Synchronization and Data Alignment in BLE 5.0 Wearable Systems,” *Sensors,* vol. 23, no. 8, p. 3954, April 2023.

[20] S. Tipparaju, J. Lee and M. Kim, “Mitigation of Data Packet Loss in Bluetooth Low Energy-Based Wearable Healthcare Ecosystem,” *Biosensors,* vol. 11, no. 10, p. 350, October 2021.

[35] E. Bell, “Whoop 4.0 Review: Should you spend over $300 on its health tracking,” Reviewed, 3 May 2022. [Online]. Available: https://www.reviewed.com/health/content/whoop-40-review-should-you-spend-over-300-its-health-tracking.

8 Appendix

8.1 Appendix A – Initial Project Description (Extract)

Project Description:

The project will involve designing a wearable smart system for monitoring several physiological parameters such as temperature, pressure, and nervous activity. This will include developing wearable clothing with integrated sensors, establishing communication with a control monitor (smartphone or laptop), and creating a smart application for data collection and analysis.

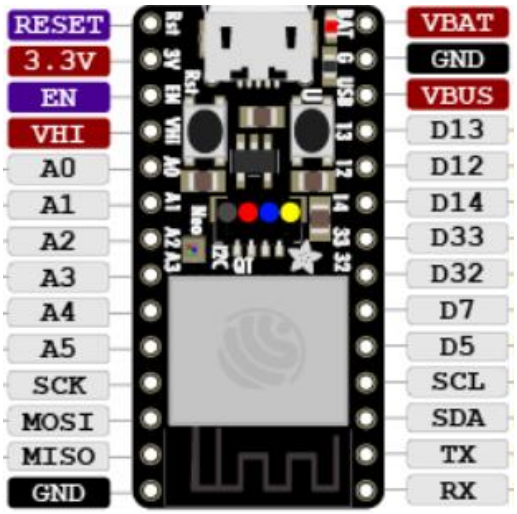
Key Skills (if listed): IoT, sensors, electronics, orogramming, PCB design

Appendix A Initial Project Description

8.2 Appendix A1 – Sensor Datasheets (Links)

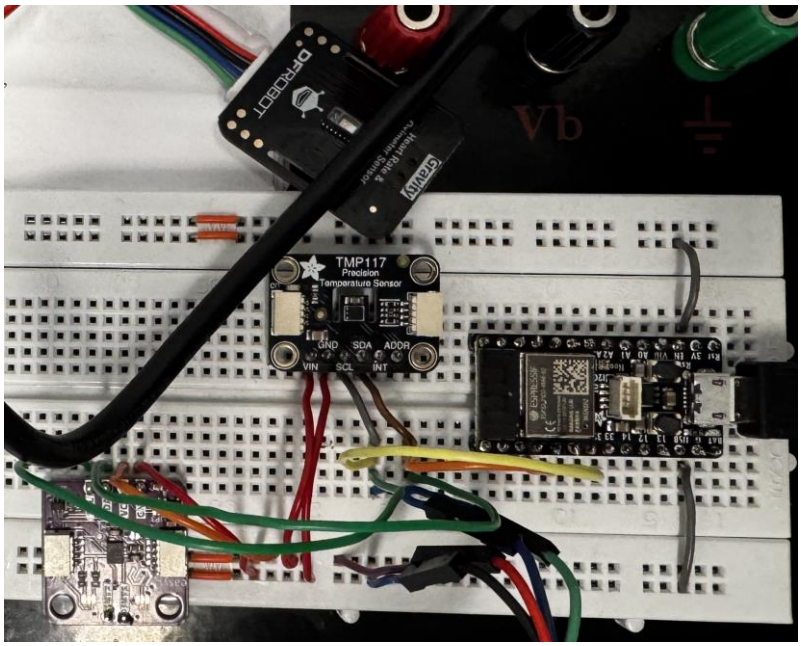
- TMP117: Texas Instruments TMP117 Datasheet
- LSM6DSO: Adafruit LSM6DSOX Datasheet
- MAX30102: Analog Devices MAX30102 Datasheet

8.3 Appendix B – ItsyBitsy ESP32 Headers Pinout



Appendix B ESP32 Header Pinout

8.4 Appendix B1 – Breadboard Prototype



Appendix C shows Breadboard Prototype

8.5 Appendix C – Arduino ESP32 Firmware Code

https://github.com/SamHakeem/Health-Monitoring-System-esp32/blob/refracted/arduino.ino

8.6 Appendix C1 – Link to Core Sensor Libraries

TMP117 – Adafruit_TMP117 Library
GitHub Repository: https://github.com/adafruit/Adafruit_TMP117
Arduino Library Page: https://www.arduino.cc/reference/en/libraries/adafruit-tmp117/

LSM6DSO – Soldered_LSM6DS3 Library
GitHub Repository: https://github.com/SolderedElectronics/LSM6DS3
Arduino Library Page: https://www.arduino.cc/reference/en/libraries/soldered-lsm6ds3/

MAX30102 – DFRobot_BloodOxygen_S Library
GitHub Repository: https://github.com/DFRobot/DFRobot_BloodOxygen_S
Arduino Library Page: https://www.arduino.cc/reference/en/libraries/dfrobot-bloodoxygen-s/

8.7 Appendix C – Python Desktop Application Code

https://github.com/SamHakeem/Health-Monitoring-System-esp32/blob/refracted/esp32_v1.0.0.py