EDB4703: Systems Integration Design Project

EDB3023/EEB4063: Data and Computer Network

MAY 2023

FINAL REPORT

GROUP 3

PROJECT: Pflege Old Folks Health Monitor Wristband

NO	NAME	ID	RESPONSIBILITY (page)
1	Tan Yu Ting	19000505	21-29, 44-54, 71, 73
2	Nur Amira binti Ahmad Rezal	19000541	4- 7, 13, 28-36, 38-42, 44,56 – 63, 72
3	Hana Najwa Binti Abdul Aziz	19000437	8-20,36-38, 42-44,64-70, 73-75

Submitted to:

Dr. Fawnizu Azmadi B Hussin

Ir. Dr. Huzein Fahmi B Hawari

Table of Contents

1. BACKGROUND OF PROJECT

1.1. Introduction

The healthcare industry is actively utilising the usable Internet of Things (IoT) devices to enhance patient care and medical monitoring and optimise healthcare services. The worldwide IoT in the healthcare market was worth USD 252.1 billion in 2022 and is predicted to increase at a compound annual growth rate (CAGR) of 16.8% between 2023 and 2030 (Ubaid, 2023).

IoT devices are being integrated into healthcare operations in hospitals, clinics, nursing homes, and even patients' homes. Apple, Samsung, and Huawei have all produced various IoT devices for healthcare wearables, such as fitness trackers and smartwatches. They provide a variety of features that aid in health monitoring and wellness management. For example, it lets patients track their fitness levels and health issues in real-time. One of the Huawei smartwatches, which can monitor a variety of data like heart rate, blood oxygen levels, blood pressure, sleep quality, step count, and more, is seen in the image below. Clear, interactive graphs that display the sensor data allow consumers to simply and quickly stay informed (Huawei, n.d).

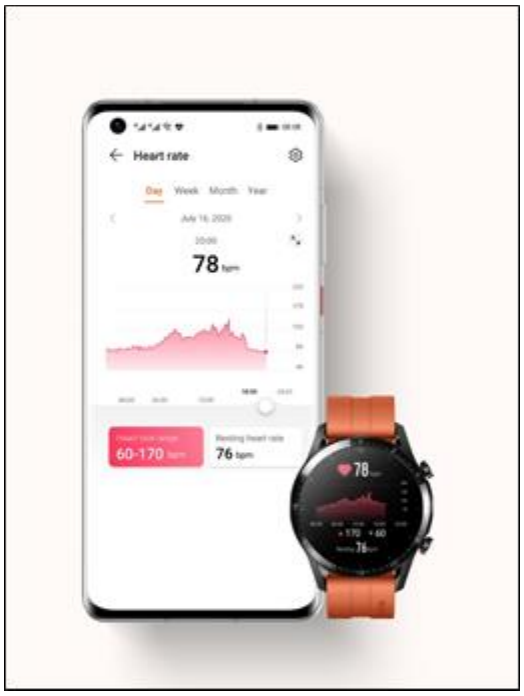


Figure 1.1: Huawei Smartwatch

Pflege Old Folks Health Monitor Wristband is a project that focuses on developing a health monitoring device that will be used in an elderly home to track old folks' health. Pflege is a German word that translates as "care." This device is designed to display and care for old folks' health. As people age, they may suffer various health issues, including chronic illnesses. Pflege is often called IoP (Internet of People) since it focuses on enabling and connecting people to communicate online (Derek, 2016). It entails utilising patient portals or mobile health applications to access their medical records. It allows them to contact doctors and other healthcare professionals, set appointments, and get individualised health information. IoT, on the other hand, describes a network of interconnected physical things, sensors, and gadgets that can speak to one another and share information via the Internet without human interaction.

Devices for health monitoring continuously check vital indicators, including heart rate, blood pressure, and oxygen levels, enabling early diagnosis of health problems and prompt treatment. For elderly persons who live alone or have restricted access to medical facilities, Pflege will provide remote health monitoring. Remote access to the data allows carers and medical experts to spot any alarming changes in a patient's health state. Other than that, the system will continuously monitor the patient's body temperature and heart rate, which will also look for patient falls. Falling increases the risk of serious injuries and problems in older persons. As a result, Pflege's fall detection features ensure a timely reaction by immediately alerting carers or emergency services in the case of a fall.

The system must pinpoint the patient's position within the nursing home to respond to any emergency. In addition to the areas where the system can monitor the patient's health state, the system should offer a user-friendly interface for analysing the patient's health status. Data about the patient's health status is first gathered using sensors, and after that, the data is communicated to a local host through WiFi and further visualised for analysis. The information gathered by the health monitor devices can offer important insights into older persons' health patterns and symptoms when they see their healthcare providers.

1.2. Literature Review

The elderly in Malaysia are those who are 60 years of age or older, according to the Malaysian Administrative Modernization And Management Planning Unit (n.d.). 2.83 million people, or 9.13% of Malaysia's 31 million people, are above the age of one (Department of Statistics, Malaysia, 2015). According to estimates, impairment prevalence increases with age, and more than 46% of adults 60 years and older have a disability. People may have numerous health issues and chronic diseases as they become older. As a result, it is crucial to use and demand health services as the population ages. Older persons' physical and cognitive abilities decline with age, impacting their capacity to live independently or carry out everyday tasks.

Chronic diseases, including diabetes, arthritis, and heart disease, disproportionately negatively impact older persons. Almost 80% of people have two or more chronic conditions, and approximately 95% have at least one (Susan, 2022). Every year, unintentional fall injuries affect three million persons 65 and older who need treatment in emergency rooms (Centre for Disease Control and Prevention, n.d.). Chronic illnesses can make it difficult for a person to carry out everyday tasks, reduce their level of independence, and necessitate institutional care, in-home carers, or other long-term support services.

In order to continually monitor and manage patients' health and well-being, a healthcare monitoring system is a vast and interconnected network of technology, tools, and software (Abdulmalek et al., 2022). IoT technology efficiently addresses the problems networks, public and private sector enterprises, and governmental organisations confront worldwide (Mohammad et al., 2022). A sensor in this health monitoring system will record the patient's health status. It is quicker, more inexpensive, and smaller in size. This device can gauge the body's temperature, heart rate, and oxygen saturation level before displaying the data online. Given that most older persons have at least one chronic illness that requires treatment, healthcare service providers must understand the needs of older adults in terms of monitoring their health (Institute of Medicine (US) Committee on the Future Health Treatment Workforce for Older Americans, 2008).

The ECG monitor is one of the healthcare monitoring systems being used. An electrocardiogram (ECG) is a test that measures the time and intensity of the electrical signals that cause the heart to beat (Take an ECG with the ECG app for Apple Watch, 2022). The ECG monitor on the Apple Watch works by measuring electrical signals across the user's wrist using built-in electrodes on the back of the watch and the digital crown (the button on the side). Users launch the specific ECG app on their Apple Watch to take an ECG reading and put their fingers on the digital crown for around 30 seconds (Amanda, 2023).

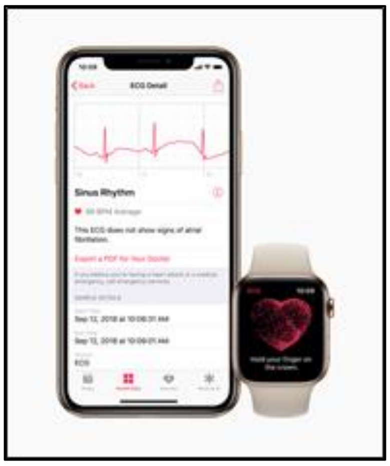


Figure 1.2.1: Apple Smartwatch and ECG Status Monitoring

Another example is blood pressure monitors (Amanda, 2023), worn on the right or left arm and precisely provide a blood pressure measurement to the user's phone. The Qardioarm Blood Pressure Monitor is one of the most well-known related products; it is the best on the market and has over 2K 5-star ratings.

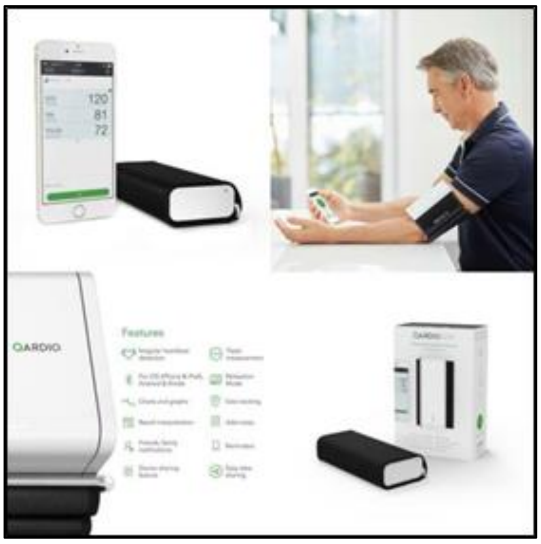


Figure 1.2.2 The Qardioarm Blood Pressure Monitor

2. PROBLEM STATEMENT & OBJECTIVES

2.1. Problem Statement

This project is mainly about creating a health monitoring system to be used in an old folks' home to monitor our users, the old folks' (patients') health condition. The challenge of continuously monitoring the health of numerous elderly residents presents itself to carers in an old folks' home. Manual monitoring can take a lot of time and may cause delays in spotting health problems. In order to help carers efficiently track the health status of elderly residents, there is a need for a trustworthy and automated health monitoring system.

In an old folks' home, there is a need for a portable and practical device for the ease and convenience of the elderly residents. They frequently have mobility issues, making heavy monitoring equipment uncomfortable for them. It is essential to create a wristband-based monitoring system that is light, discrete, and comfortable to wear so that they can move about without restriction.

For the purpose of identifying anomalies or irregularities that could point to potential health problems of the patients, continuous and accurate heart rate monitoring is crucial. The system will consistently monitor the patient's heart rate and body temperature, at the same time able to detect if the patient falls down.

The difficulty lies in creating a wristband equipped with a trustworthy heart rate monitoring sensor and algorithm that delivers reliable, accurate heart rate measurements. Consistent body temperature monitoring is essential for spotting fevers and other health changes early on. The challenge is to incorporate a temperature sensor that can precisely gauge body temperature and offer continuous monitoring into the wristband design.

The old folks who fall can sustain serious injuries that call for immediate medical attention. However, it is difficult for the caretakers to be altered as the old folks may not be under supervision at all times. Hence why the accelerometer and gyroscope sensor has to implement a fall detection algorithm that will reliably identify falls and send timely alerts to caretakers.

Besides, it is crucial for the system to detect the patient's location precisely within the old folk's home to cope with any emergencies. The difficulty is to create an indoor positioning system that can locate the patient precisely using the wristband's sensors or other features of the Old Folks' home.

Not only the parts where the system can monitor the patient's health condition, the system should provide a user-friendly interface on the analysis of the patient's health condition. It offers up-to-date health information and alerts when necessary is required by caretakers. Creating an intuitive and simple-to-use interface that allows carers to access and monitor patient health data efficiently and quickly identify any health concerns is a challenge.

After the patient's health condition is collected by using sensors as data, the data will then be transmitted to cloud through WiFi, and the data will then further visualized for better analysis. By storing health data in the cloud, it is possible for authorised personnel to have centralised access to it, analyse it easily, and track patients' health over time. To transmit and store data between the wristband and a cloud server, a secure and dependable communication link must be established.

Pflege Old Folks Health Monitor aims to consistently check the patient's heart rate and body temperature, detect falls, determine the patient's indoor location, and transmit all health-related data to the cloud for record-keeping. The system should be designed to ease the workload of caretakers in monitoring patients' health status and conditions efficiently and effectively. The final product should be a comprehensive healthcare monitoring system for old folks in an Old Folk's home which will significantly improve the care provided to them, enhance early detection of health issues, and streamline the caregiving process for the caregivers.

2.2. Project Objectives

A thorough overview of a functioning Health Monitor System is given by the analytic dashboard, which operate in real-time. This makes it possible for the caretakers to recognise problems with the old folks' health status and promptly attend to them. In this project, our team's prototype, Pflege Old Folks Health Monitor Wristband aims to achieve the following objectives:

- To check the patient's heart rate consistently.
  - The dashboard gives caretakers access to a continuous heart rate monitoring of the old folks. The design should be a reliable and non-invasive heart rate monitoring system that can consistently and accurately track the patient's heart rate without causing discomfort to the patient.

- To check the patient's body temperature consistently.
  - Continuous body temperature monitoring is visualised. Using appropriate sensors and hardware, a temperature monitoring mechanism that ensures precise and continuous measurement of the patient's body temperature without impeding their daily activities is implemented.

- To detect whether the patient falls down.
  - An algorithm is developed that can reliably detect falls and distinguish them from normal movements to promptly alert caretakers in case of emergencies by incorporating a fall detection mechanism that is sensitive and responsive to emergency situations.

- To detect the actual location of the patient inside the Old Folk's home (indoor).
  - Using available resources, an indoor positioning system is implemented that can accurately determine the patient's location within the Old Folk's home without relying on GPS signals.

- To transmit patient's health condition to the cloud and save records.
    - By implementing Data and Computer Network course, a Cloud Data Transmission and Storage is implemented. A secure and efficient communication protocol is used to transmit the patient's health data to a cloud server for storage, analysis, and easy accessibility by authorized healthcare personnel.

- To ease the old folks' caretakers in monitoring patients' overall status/condition
    - The visualising design must provide a convenient, intuitive, and user-friendly interface for caretakers to monitor and access patient data efficiently, enabling quick identification of any abnormal health conditions anytime anywhere in real-time.

- To provide a portable, practical device for the ease and convenience of the user.
    - The health monitoring device needs to be portable so that it can be worn by the elderly at all times without causing discomfort or hindrance to their daily activities, easy to use with adjustable sizes to fit comfortably on different individuals.

3. PROJECT MANAGEMENT PLANNING

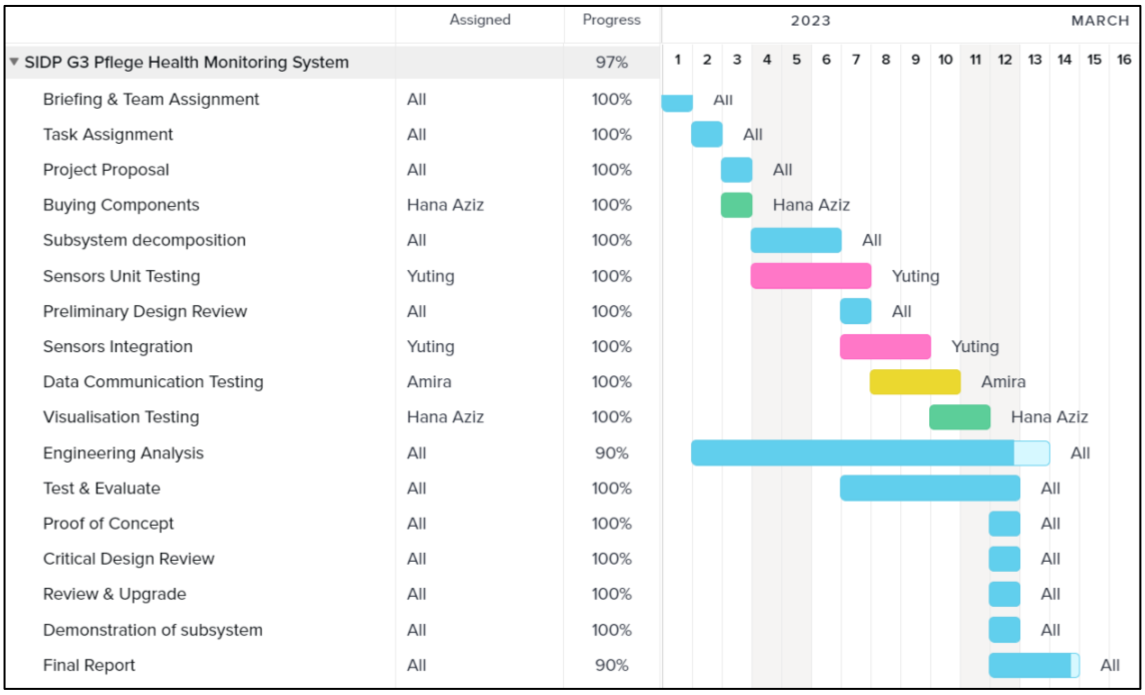
3.1 Task Listing and Distribution

Team Members	Task Distribution
Tan Yu Ting	• Lead and organize weekly meetings. • Arrange meetings with the team. • Handle project design concept. • Identify suitable electronic components. • Justify fabrication choices. • Unit testing on the sensors to check for functionalities. • Calibrate each sensor according to real-life cases. • Verify the function of sensor based on several test case scenarios. • Integrated testing of all sensors to achieve health monitoring. • Research and experiment on different methods for indoor positioning. • Mapping for indoor positioning. • Check and optimize codes. • Prototype building • Handle project report. • Lead the editing video presentation • Suggest recommendations to improve the weaknesses of the product.
Nur Amira binti Ahmad Rezal	• Conduct project background • Handle literature review. • Identify suitable tools or software for data transmission and communication • Handle communication technology

	• Provide the specification data sheet for the communication module selected • System design and optimization • Prototype building • Mapping for indoor positioning. • Configure ESP32 Microcontroller • Responsible in extract data from Arduino.IDE or devices to Thinger.io • Handle project report. • Optimize code to extract data from Arduino.IDE or devices to Thinger.io. • Suggest recommendations to improve the weaknesses of the data communication. • Responsible of FMEA sheet for data and communication subsystem. • Elaborate the relation of product data communication with Data and Communication Network theory.
Hana Najwa binti Abdul Aziz	• Team Leader • Assists in project design and material selection. • Handle the financial related item in the project. • Responsible for the project capital and operational cost considerations • Research items and prices for construction of prototype. • Handle electronic components soldering. • Prototype building • Handle project management planning. • Schedule meetings in Teams.

	• Record the meetings and take note of the meeting details. • Optimize codes. • Mapping for indoor positioning. • Produce block diagram and project work flowchart • Handle communication technology and elaborate communication issues • Handle data visualization and data monitoring in Cloud service and Mobile Apps • In charge of dashboard design • Handle project report.

3.2 Gantt Chart

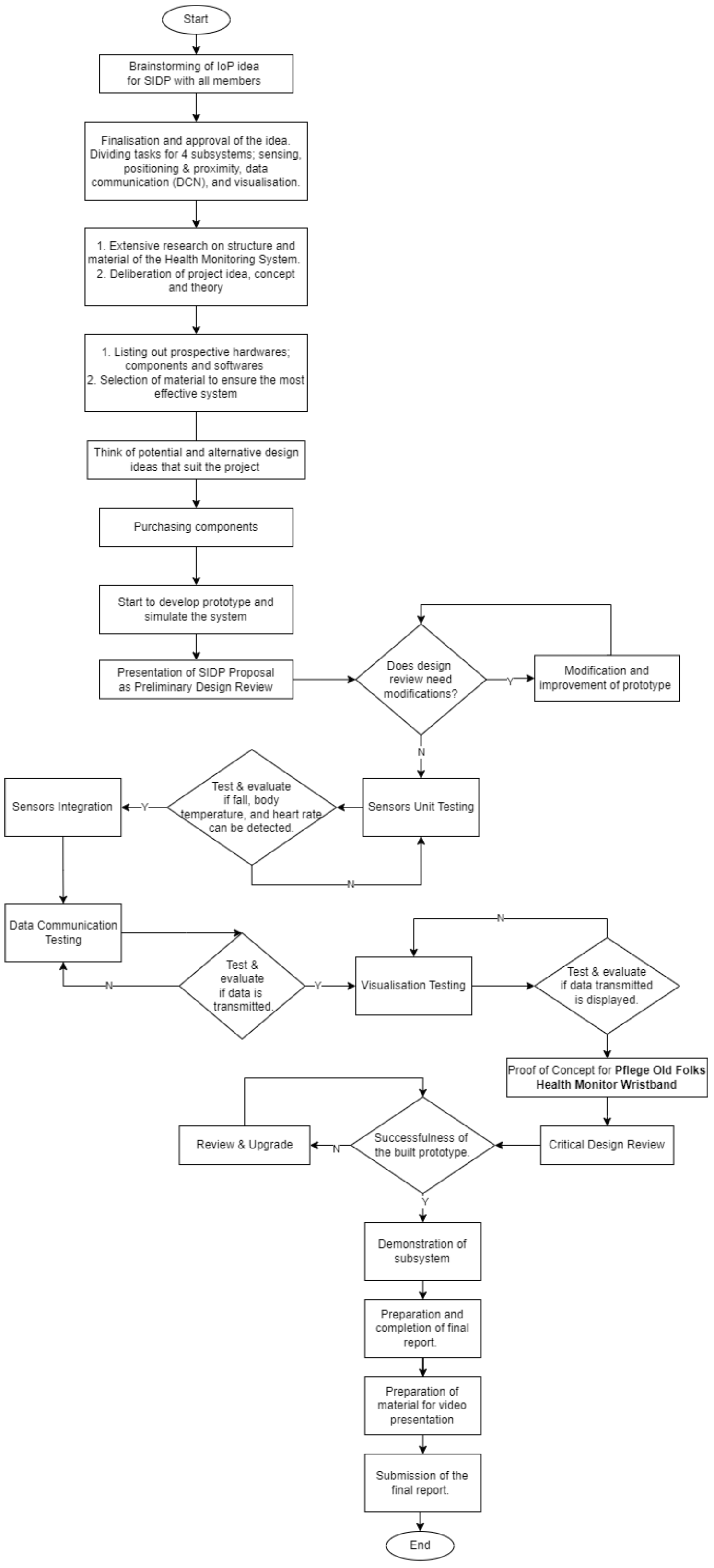


	Assigned	Progress
▾ SIDP G3 Pflege Health Monitoring System		97%
Briefing & Team Assignment	All	100%
Task Assignment	All	100%
Project Proposal	All	100%
Buying Components	Hana Aziz	100%
Subsystem decomposition	All	100%
Sensors Unit Testing	Yuting	100%
Preliminary Design Review	All	100%
Sensors Integration	Yuting	100%
Data Communication Testing	Amira	100%
Visualisation Testing	Hana Aziz	100%
Engineering Analysis	All	90%
Test & Evaluate	All	100%
Proof of Concept	All	100%
Critical Design Review	All	100%
Review & Upgrade	All	100%
Demonstration of subsystem	All	100%
Final Report	All	90%

Figure 3.2.1: Gantt Chart of Fabricating Pflege Old Folks Health Monitor Wristband

4 METHODOLOGY

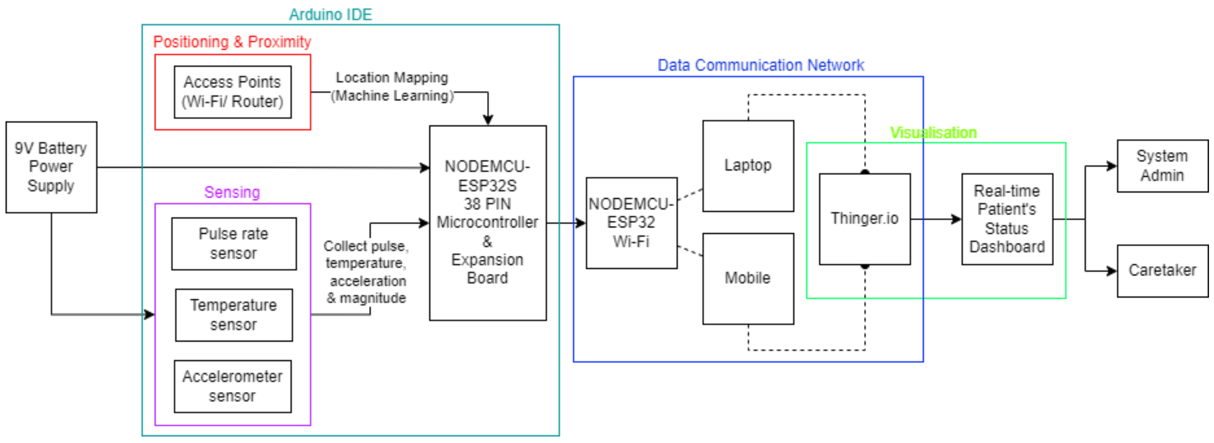
4.1 Project Workflow



Start
Brainstorming of IoP idea for SIDP with all members
Finalisation and approval of the idea. Dividing tasks for 4 subsystems; sensing, positioning & proximity, data communication (DCN), and visualisation.
1. Extensive research on structure and material of the Health Monitoring System.
2. Deliberation of project idea, concept and theory
1. Listing out prospective hardwares; components and softwares
2. Selection of material to ensure the most effective system
Think of potential and alternative design ideas that suit the project
Purchasing components
Start to develop prototype and simulate the system
Presentation of SIDP Proposal as Preliminary Design Review
Does design review need modifications?
Y
Modification and improvement of prototype
N
Sensors Unit Testing
Test & evaluate if fall, body temperature, and heart rate can be detected.
Y
N
Sensors Integration
Data Communication Testing
Test & evaluate if data is transmitted.
N
Y
Visualisation Testing
Test & evaluate if data transmitted is displayed.
N
Proof of Concept for Pflege Old Folks Health Monitor Wristband
Critical Design Review
Successfulness of the built prototype.
N
Review & Upgrade
Y
Demonstration of subsystem
Preparation and completion of final report.
Preparation of material for video presentation
Submission of the final report.
End

Figure 4.1: Project Flowchart

4.2 Block Diagram



Arduino IDE
Positioning & Proximity
Access Points (Wi-Fi/ Router)
Location Mapping (Machine Learning)
9V Battery Power Supply
Sensing
Pulse rate sensor
Temperature sensor
Accelerometer sensor
Collect pulse, temperature, acceleration & magnitude
NODEMCU-ESP32S 38 PIN Microcontroller & Expansion Board
Data Communication Network
NODEMCU-ESP32 Wi-Fi
Laptop
Mobile
Visualisation
Thinger.io
Real-time Patient's Status Dashboard
System Admin
Caretaker

Figure 4.2: Block Diagram of Pflege Subsystems Design

4.3 Design Concept

The design has been decided upon discussion. Sufficient research has also been carried out to ensure that cost-material effective materials and components are used during the prototype's the building process. Besides, in case we need assistance there will be many experienced lab technicians that are willing to assist and guide us during the fabrication process.

The building of prototype started in week 10. In constructing a fully functional prototype, we will keep work on testing the functionality, make further amendments and resolve any technical or design issues. Labs with functional and sufficient equipment are provided by UTP throughout the process.

4.3.1 Proposed Concept

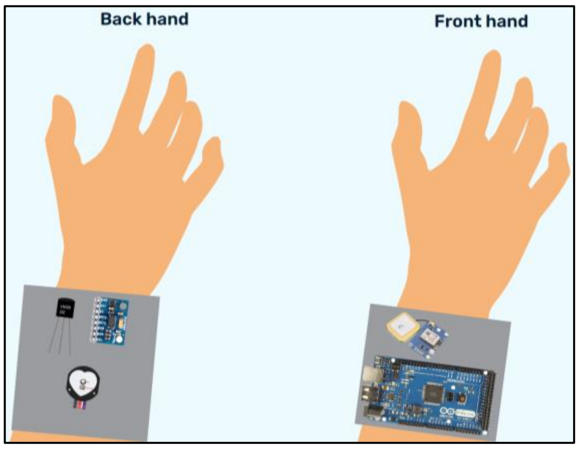


Figure 4.3.1.1: Initial Pflege Proposed Design Concept

Pflege Old Folks Health Monitor Wristband designed by our team will be placed in a covered rectangular plastic case containing the microcontroller and its baseboard along with all the accelerometer sensor and buzzer. The prototype will be portable and allows the patient to bring along with them whenever and wherever they are, as the prototype is worn on the patient’s wrist. There will also be a temperature sensor that can detect the body temperature of the patient, so that the caretaker can continuously monitor and keep track of.

We aim to dissolve any inconveniences for our users in handling the prototype by making it attached to a stretchable wristband that can be removed and worn easily. The wristband design should be comfortable to wear for extended periods, avoiding any irritation or pressure points that might lead to discomfort or skin issues. If the user needs to immediately take it off, they can just slip the wristband off. Due to the plethora of electronic components, direct contact with water must be avoided.

After discussion, Pflege Old Folks Health Monitor Wristband will not be very convenient if it comes with a larger power supply to operate like a power bank, thus, the prototype is designed for it to be supplied to direct current (9V battery plugged into a DC Jack as it regulates the input voltage). Given that some elderly residents may have limited technological experience, the device should be straightforward to use. Our prototype allows flexibility according to the needs of the user, where they do not necessarily have to interact with Pflege Old Folks Health Monitor Wristband for it to operate as it functions just by being connected to a power supply.

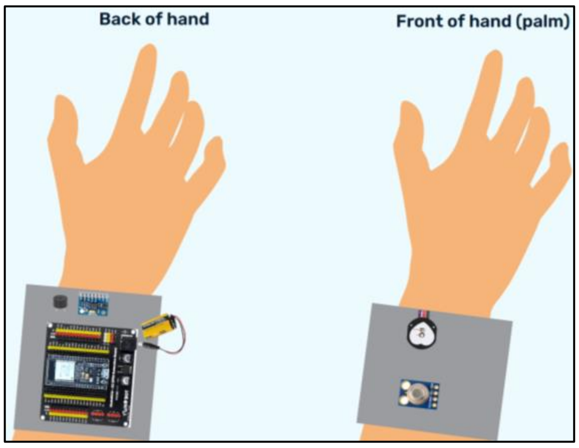


Figure 4.3.1.2: Finalised Pflege Proposed Design Concept

4.3.2 Final Prototype

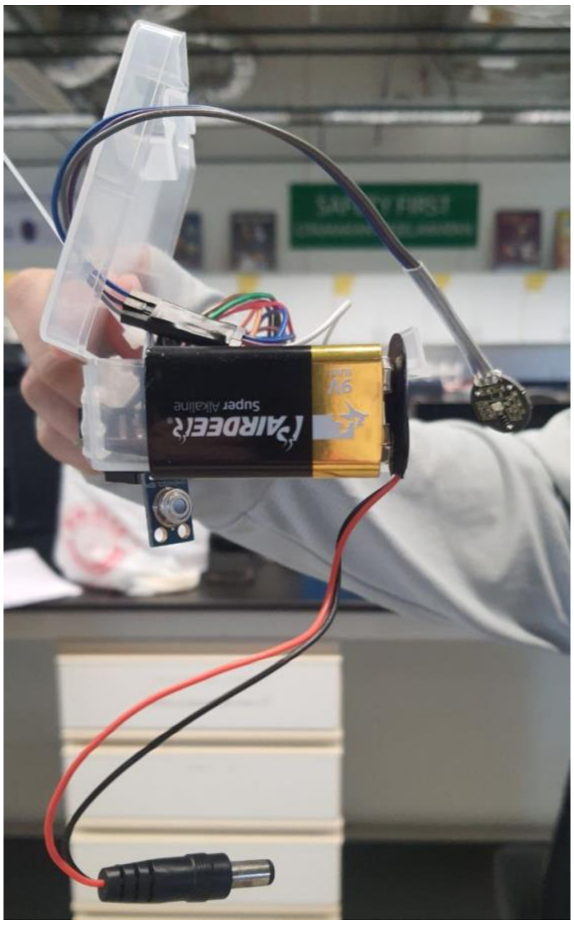


Figure 4.3.2.1: Side-view of Pflege Prototype

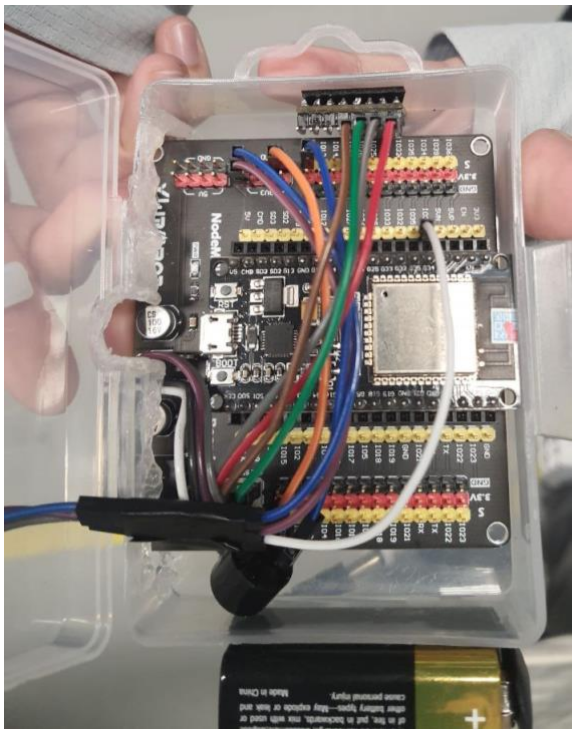


Figure 4.3.2.2: Plan View of Pflege Prototype

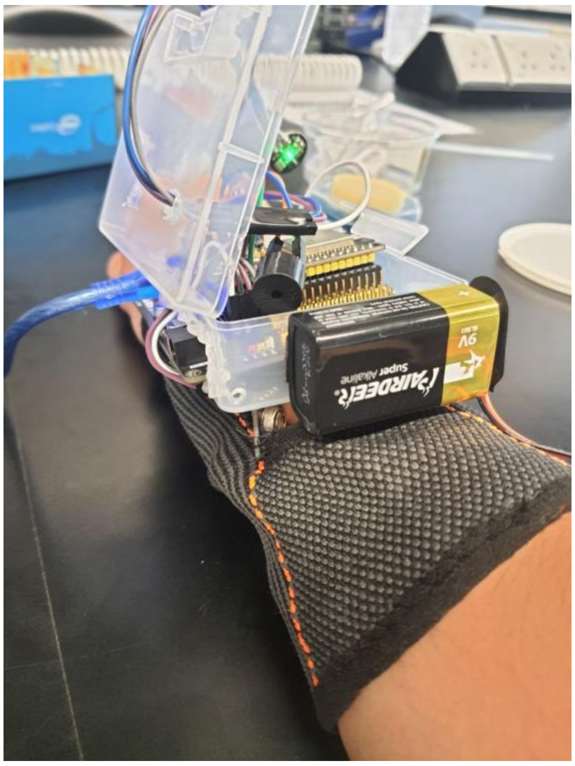


Figure 4.3.2.3: Pflege Prototype Worn on Wrist

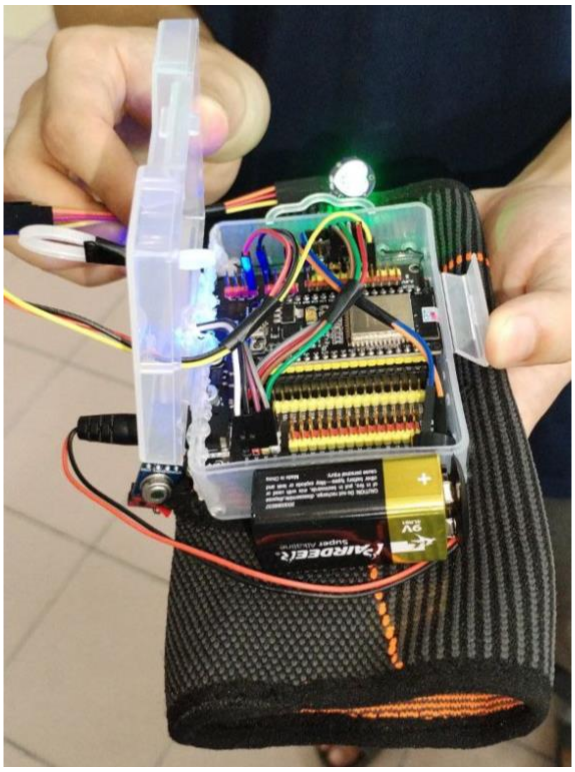


Figure 4.3.2.4: Overview of Pflege Prototype

4.3.3 Components Used

Components	Type of Components Used
Microcontroller	NODEMCU-ESP32S 38 PIN
Microcontroller Expansion	Base Board for NODEMCU-ESP32S 38 PIN
Temperature Sensor	MLX90614 Non-Contact Infrared Temperature Sensor
Pulse Sensor	XD-58C Pulse Sensor
Accelerometer	GY-521 MPU6050 6DOF Accelerometer

Table 1: Type of Components Used

Components	Pins Configuration
Temperature Sensor	3.3V, GND, GPIO34 (ADC6)
Accelerometer	3.3V, GND, SCL, SDA
Pulse Sensor	3.3V, GND, SCL, SDA
Buzzer	GPIO13, GND
9V Battery	Base Board DC Power Jack

Table 2: Components and Their Pins Configuration

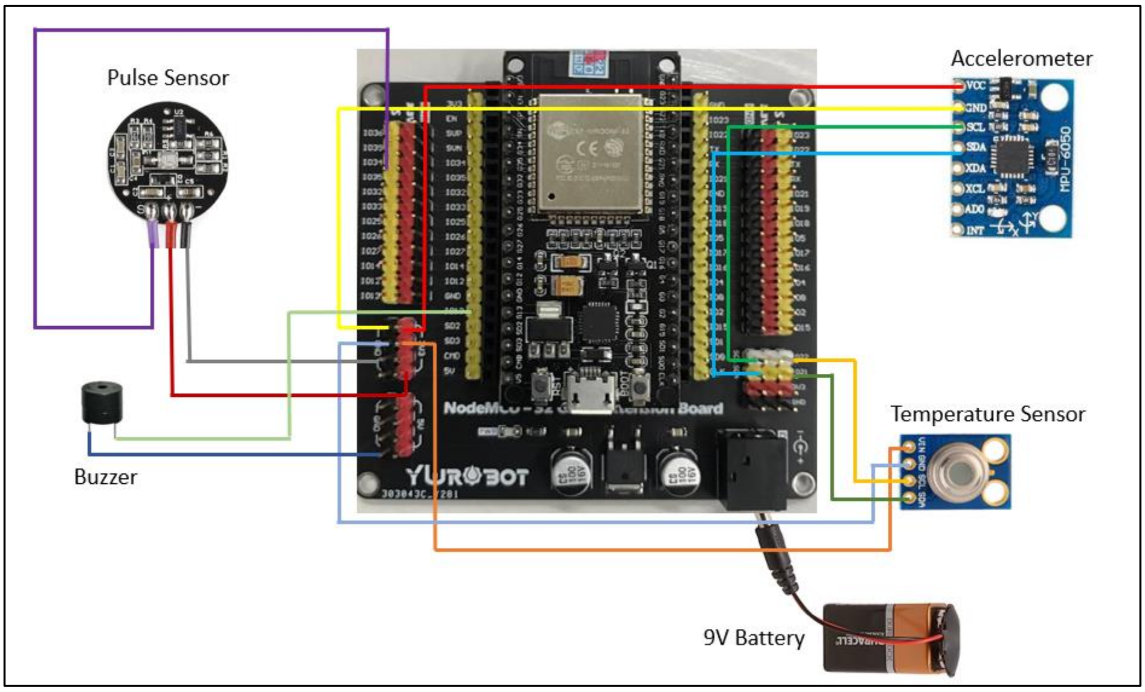


Figure 4.3.3: Circuit Diagram of Pflege Old Folks Health Monitor Wristband Prototype

- NODEMCU-ESP32S 38 PIN

  Microcontroller ESP32, is a highly versatile and powerful microcontroller development board designed and manufactured by Espressif Systems. It serves as an ideal platform for a wide range of electronic projects and Internet of Things (IoT) applications. The benefits of ESP32 is listed as below:

1. ESP32 Microcontroller Chip

   - The board is equipped with a dual-core Xtensa LX6 processor, capable of running at speeds of up to 240 MHz. This dual-core architecture enables simultaneous multitasking, making it suitable for handling complex tasks.

2. Wireless Connectivity

   - One of the key highlights of the ESP32 is its built-in wireless connectivity. It supports both Wi-Fi 802.11 b/g/n and Bluetooth 4.2 (and later versions), allowing seamless internet connection and wireless communication with other devices. Additionally, the ESP32 offers enough memory, including around 520KB of SRAM for data storage and 448KB of ROM for firmware.

3. GPIO Pins on ESP32

   - The board also provides a variety of Input/Output (I/O) pins which allows the connection of sensors, actuators, and other components. These pins support multiple communication protocols such as digital input/output, PWM, I2C, SPI, and UART. Furthermore, the ESP32 includes an Analog-to-Digital Converter (ADC) that can read data from analogue sensors.

4. Power Supply Options

    - Powering the ESP32 is straightforward, as it can be done through a micro-USB port or an external power supply. Some boards are equipped with LiPo battery charging circuits, making them suitable for portable and battery-powered applications.

5. Programming Interface

    - The ESP32 can be programmed using popular environments like Arduino and MicroPython. Its compatibility with multiple programming languages allows developers of all levels to work with the board easily. Additionally, the board offers various programming interfaces, such as UART, JTAG, and SWD, for debugging and development purposes.

Therefore, with its cost-effectiveness, ease of use, and extensive capabilities, the ESP32 has gained widespread popularity among hobbyists, makers, and professional developers for creating innovative IoT and embedded projects. Whether for building simple Wi-Fi-enabled sensors or more complex home automation systems and IoT applications, the ESP32 stands as an excellent choice, offering a powerful and flexible platform to bring creative ideas to life.

In the project, we have used the NODEMCU-ESP32S (38 pin) where it is a specific variant of the ESP32 development board, following the popular NodeMCU form factor. This board has gained significant popularity among hobbyists, makers, and beginners in the realm of IoT and embedded systems due to its user-friendly design and versatile capabilities.

- Base Board/ Expansion Board for NODEMCU-ESP32S

Besides, we also implement the base board for the NODEMCU-ESP32S where the base board is a valuable hardware accessory that complements the ESP32 development board, providing numerous enhancements and features.

1. Pin Compatibility & Enhanced Connectivity

   - The base board typically follows a form factor that matches the NodeMCU-ESP32S development board, ensuring a perfect fit and alignment. It has the same pin configuration which allow the NODEMCU-ESP32S board to be easily plugged into the base board. This compatibility ensures seamless integration, making it more straightforward to connect and use additional components.

   - The base board extends the number of GPIO pins available to the user. This expansion enables the connection of more sensors, actuators, and other peripherals to the ESP32 board. Having more GPIO pins at hand allows for more extensive and complex project designs without the need for external multiplexing or additional hardware.

   - Depending on the specific base board, it provides additional communication interfaces, such as I2C, SPI, or UART to facilitate the connection of various external modules or sensors. For example, the pins for SCL and SDA increased by using base board and this helps to cope with our project's needs.

2. Regulate Power Supply

   - The base board may include a regulated power supply circuit to ensure stable and reliable power delivery to the ESP32 board and connected components. This can be especially useful when dealing with multiple peripherals that require different power levels. For example, our group used 9V battery to power up the ESP32 boards where this is impossible without the help of the base board since the 9V battery is supplying too high power to the ESP32 boards and might burn it.

- MLX90614 Non-Contact Infrared Temperature Sensor

  The MLX90614 is a non-contact infrared temperature sensor developed by Melexis N.V. It is widely used in various applications where precise and non-contact temperature measurement is required. The sensor operates based on the principle of infrared radiation emitted by objects, allowing it to measure the temperature of an object without physically touching it.

1. Non-Contact Measurement

   - The MLX90614 is a non-contact infrared temperature sensor, meaning it can measure the temperature of an object without physical contact. This feature is particularly advantageous for measuring body temperature, as it eliminates the need for direct contact with the individual being measured, reducing the risk of cross-contamination, and improving hygiene, especially in medical settings.

2. Accurate and Fast Response Time

   - The MLX90614 offers high accuracy and resolution in temperature measurement. It can provide temperature readings with a resolution of up to 0.02°C and accuracy within ±0.2°C in the specific temperature range. This level of precision ensures reliable and consistent body temperature readings, critical for accurate diagnosis and monitoring of medical conditions.

   - The MLX90614 has a fast response time, providing real-time temperature measurements in a matter of milliseconds. This quick response time is beneficial for quickly detecting changes in body temperature and is particularly crucial in medical emergencies and situations where immediate temperature assessment is required.

- XD-58C Pulse Sensor

  The XD-58C Pulse Sensor is designed specifically for detecting a person's heart rate and pulse. It utilizes an optical sensing technique called photoplethysmography (PPG) to capture variations in blood volume in the peripheral blood vessels, which occur with each heartbeat.

  1. Photoplethysmography (PPG) Technology

     - The XD-58C Pulse Sensor uses PPG technology, which involves shining a light through the skin and detecting changes in light absorption caused by blood flow. As the heart pumps, the volume of blood in the blood vessels changes, leading to variations in light absorption. The sensor measures these changes, providing data that can be used to calculate the heart rate.

  2. Non-Invasive Measurement

     - The XD-58C Pulse Sensor is non-invasive, meaning it does not require any penetration of the skin or contact with the bloodstream. It can be placed on the skin's surface without causing discomfort to the user, making it suitable for continuous and comfortable heart rate monitoring.

- GY-521 MPU6050 6DOF Accelerometer

  The GY-521 MPU6050 6DOF (Degrees of Freedom) accelerometer is well-suited for fall detection due to its advanced motion sensing capabilities and compact form factor. Fall detection is a critical application in healthcare and wearable devices, aiming to identify and alert caregivers or emergency services when a person experiences a sudden fall, especially in vulnerable populations like the elderly in our project.

1. Combination of Accelerometer and Gyroscope

   - The MPU6050 integrates both an accelerometer and a gyroscope into a single chip. The accelerometer measures linear acceleration along three axes (X, Y, and Z), while the gyroscope detects angular velocity around those same axes. This combination of motion sensors allows the device to capture a comprehensive picture of movement, making it effective for fall detection.

2. DMP (Digital Motion Processor) Capability

   - The MPU6050 includes a Digital Motion Processor (DMP) that offloads some of the sensor fusion and processing tasks. The DMP can process the raw sensor data to provide accurate orientation and motion data, simplifying the development of fall detection algorithms and reducing the computational burden on the main processing unit.

4.3.4 Sensors Details and Testing

Before we start using the sensors in our project, it is important for us to do some research on the details of the sensors first. The most important step is to study the manufacturer's datasheet and application notes for the sensor as these documents provide detailed information about the sensor's operation, recommended usage, and best practices for obtaining accurate measurements. Datasheets will also include information crucial for integration such as electrical characteristics, pin configurations, and communication protocols. However, we still need to verify the information provided in the datasheet through real-world testing and experimentation on the sensors. This is because sometimes the datasheet values may vary slightly based on external factors and thus, it is crucial for us to validate the sensor's performance in our project.

1. XD-58C Pulse Sensor

Details	**Results**
Sensitivity	LED peak wavelength of 515nm
Operating Temperature	-40°C to +85°C
Power Requirements	Works with 3V or 5V of power supply.
Libraries Used in Arduino IDE	-
Calibration	No calibration required.
Communication	Analog Output
Environmental Considerations	Can be influenced by environmental factors such as temperature, humidity, and ambient light.

Table 3: XD-58C Pulse Sensor Specifications

2. MLX90614 Non-Contact Infrared Temperature Sensor

Details	**Results**
Sensitivity	Detect object temperature from -70°C to +380°C
Accuracy	MLX90614 offers a standard accuracy of ±0.5˚C around room temperatures.
Operating Temperature	-40°C to +125°C
Power Requirements	The MLX90614 operates on 3.3V power.
Libraries Used in Arduino IDE	Adafruit MLX90614 Library
Calibration	No calibration required.
Communication	I2C Protocol
Environmental Considerations	Can be influenced by environmental factors such as temperature, humidity, and ambient light.

Table 4: MLX90614 Non-Contact Infrared Temperature Sensor Specifications

3. GY-521 MPU6050 6DOF Accelerometer\

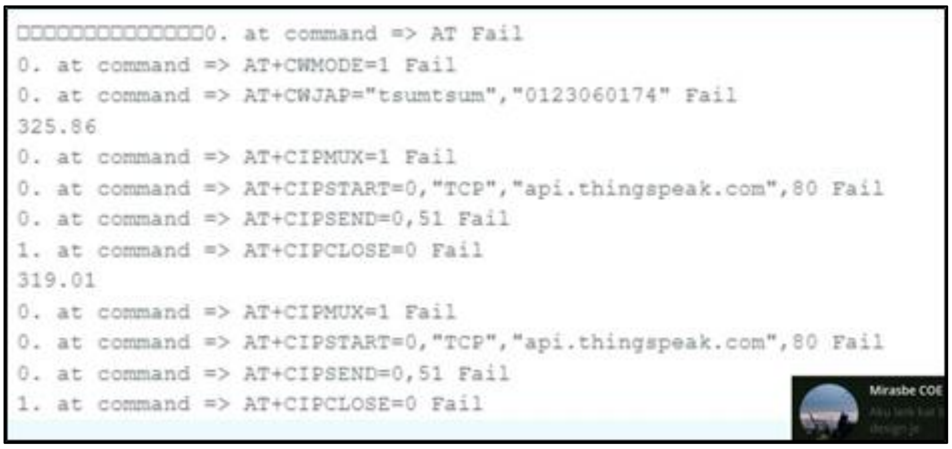
Details	**Results**
Gyroscope Sensitivity	Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec.
Accelerometer Sensitivity	Digital-output triple-axis accelerometer with a programmable full-scale range of ±2g, ±4g, ±8g and ±16g.
Operating Temperature	-40°C to +105°C
Power Requirements	VDD supply voltage range of 2.375V-3.46V
Libraries Used in Arduino IDE	MPU6050 Library
Calibration	No calibration required.
Communication	I2C Protocol
Environmental Considerations	Can be influenced by environmental factors such as temperature, humidity, and vibrations.

Table 5: GY-521 MPU6050 6DOF Accelerometer Specifications

4.3.5 Visualisation Platform

A visualisation platform aims to provide a complete and user-friendly interface that displays the patient's current temperature, pulse, indoor and outdoor position, and positioning in a visually appealing and readily accessible manner. These platforms are intended to collect, evaluate, and display different healthcare-related data to assist carers in making informed decisions and improving overall healthcare outcomes.

The visualisation platform is critical in transforming raw data into usable insights, supporting data-driven decision-making, improving patient care, and eventually contributing to the overall development of the healthcare system. Thingspeak, an Internet of Things (IoT) platform and web service that allows developers to gather, analyse, and display data from IoT devices and sensors, is the first visualisation being tested. ThingSpeak communicates data from IoT devices and sensors using HTTP or MQTT protocols. The platform may also contain tools for visualising collected data through configurable charts, graphs, and gauges.



```
□□□□□□□□□□□□□□□0. at command => AT Fail
0. at command => AT+CWMODE=1 Fail
0. at command => AT+CWJAP="tsumtsum","0123060174" Fail
325.86
0. at command => AT+CIPMUX=1 Fail
0. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 Fail
0. at command => AT+CIPSEND=0,51 Fail
1. at command => AT+CIPCLOSE=0 Fail
319.01
0. at command => AT+CIPMUX=1 Fail
0. at command => AT+CIPSTART=0,"TCP","api.thingspeak.com",80 Fail
0. at command => AT+CIPSEND=0,51 Fail
1. at command => AT+CIPCLOSE=0 Fail
```

Figure 4.3.5.1: ThingSpeak Transmission Failure

The failure to connect the esp8266 to Thingspeak due to a hardware flaw that is generating connectivity problems is depicted in the figure above. In order to proceed with setting up the WiFi connection to Thingspeak, the group decides to switch to an ESP32 microcontroller. The graph below depicts the temperature and pulse sensor values extracted from the Arduino IDE and sent to ThingSpeak through ESP32.

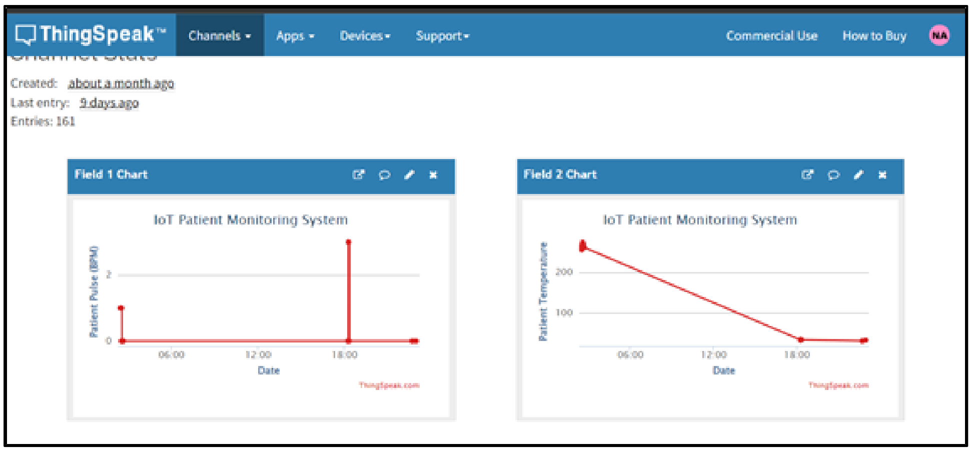


Figure 4.3.5.2: Temperature and Pulse Sensor Visualised in ThingSpeak

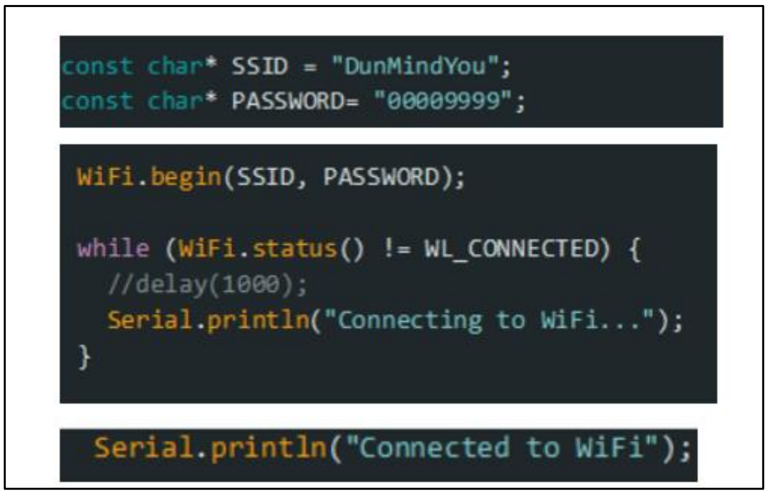
However, there is a limitation in utilizing the ThingSpeak visualization where it is unable to display a Text and outdoor positioning of patient. Thus, our group proceed with Thinger.io which is a visualisation tool used by carers to examine patient healthcare data. It is an open-source Internet of Things (IoT) platform allowing developers to create and manage IoT apps and projects. Thinger.io provides cloud infrastructure for IoT devices to connect and share data safely.

thinger.io

Figure 4.3.5.2: Thinger.io

To extract raw data from an Arduino.ide and send it to Thinger.io using an ESP32 is as shown below:

1. An account on Thinger.io is created to set up a project.
2. The Arduino sketch is written to collect raw data from sensors, set up WiFi, and connect to the internet.

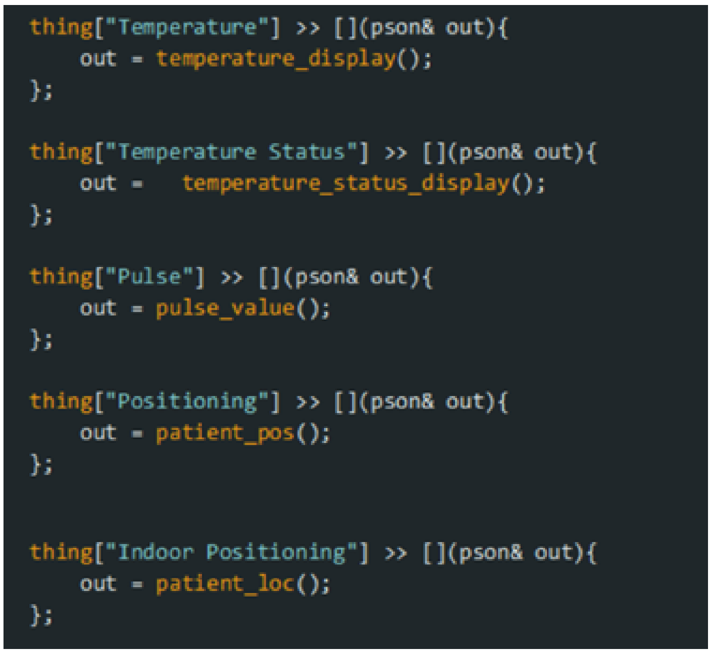


```
const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";
```

```
WiFi.begin(SSID, PASSWORD);

while (WiFi.status() != WL_CONNECTED) {
  //delay(1000);
  Serial.println("Connecting to WiFi...");
}
```

```
Serial.println("Connected to WiFi");
```

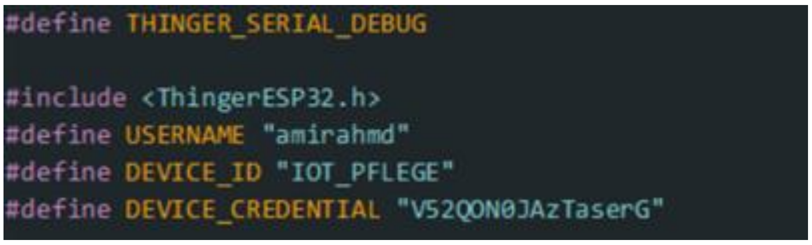
Figure 4.3.5.3: WiFi Setup in the Arduino sketch



```
thing["Temperature"] >> [](pson& out){
    out = temperature_display();
};

thing["Temperature Status"] >> [](pson& out){
    out =   temperature_status_display();
};

thing["Pulse"] >> [](pson& out){
    out = pulse_value();
};

thing["Positioning"] >> [](pson& out){
    out = patient_pos();
};


thing["Indoor Positioning"] >> [](pson& out){
    out = patient_loc();
};
```

Figure 4.3.5.4: Arduino code that allow Thinger.io to read a sensor value using the operator ">>"

3. The Thinger.io library is included at the beginning of the code, where Thinger.io is set up with the account credentials, such as username, device ID, and device credentials, as shown in the figure below.



```
#define THINGER_SERIAL_DEBUG

#include <ThingerESP32.h>
#define USERNAME "amirahmd"
#define DEVICE_ID "IOT_PFLEGE"
#define DEVICE_CREDENTIAL "V52QON0JAzTaserG"
```

Figure 4.3.5.5: Thinger.io device credentials

4. The code is compiled and uploaded to the ESP32 board.
5. Once the device is on and the ESP32 is connected to WiFi, the Thinger.io account can be accessed, and the data sent by the ESP32 can be viewed and visualized in real time.

Thinger.io allows users to select from various widget types to display in their dashboard, as illustrated in the picture below. In the dashboard, the time series chart widget displays the patient's body temperature and pulse value, while the text/value widget displays the patient's current location, positioning, and temperature status. The Google Maps widget is also utilised to determine the current position of the patient's IoT device.

4.4 IOT System Architecture / Data Transmission

4.4.1 Controllers / Sensors And Type Of Communication

The ESP32 Wi-Fi module is used as the connection between the board and the Thinger.io website. It supports Wi-Fi 802.11 b/g/n and Bluetooth 4.2 (and later versions), allowing you to connect to the internet and communicate wirelessly with other devices. Wi-Fi is the connection mode because it provides more data transmission than Bluetooth for quicker communication. It allows huge files to be exchanged rapidly, smoothly collaborate, and speedier decision-making. Cloud computing relies on rapid data transmission speeds to access and upload data to cloud servers. Higher speeds increase the responsiveness and accessibility of cloud-based apps and services. Aside from having a greater range than Bluetooth (15 metres), it can cover wider regions, enabling dependable internet connectivity throughout homes, businesses, or outdoor spaces without extra access points or repeaters. Wi-Fi with a longer range may penetrate walls, ceilings, and obstructions more efficiently, delivering a steady and constant connection in different building sections.

4.4.2 Disadvantages / Advantages Compared To Other Communication Type

Comparison	Wi-Fi	Bluetooth
Data Transfer Speed	Higher data transfer rates of 2.4 or 5GHz are available.	Lower data transfer rates which is 24 Mbps.
Application	Data transmission that is fast and continuous, such as video streaming or massive file transfers.	Smaller data payloads or data that is not time-sensitive, such as operating smart home devices,
Range	More expanded range of 15 metres, reaching several hundred feet in open settings.	Shorter range, often up to 30-100 feet
Power Consumption	During data transmission, the power consumption ranges from 2 to 20 watts.	One watt of electricity consumption is used.

Table 6: Comparison between Wi-Fi and Bluetooth.

4.4.3 Communication Technology

1. IEEE Specification:

   Complies with the IEEE 802.11 family of standards, supporting various Wi-Fi modes like 802.11b/g/n. Mobile phone Wi-Fi hotspots typically follow the IEEE 802.11 family of standards, which includes various Wi-Fi standards like 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, and 802.11ax.

2. Communication Technology:

   Mobile Phone Wi-Fi Hotspot

3. Range Limit:

   A mobile phone Wi-Fi hotspot has an effective range between 19 metres to over 90 metres. The range limit of ESP32 Wi-Fi module can vary depending on environmental conditions and obstacles, but in general, it has a range of up to hundreds of metres in open spaces.

4. Transfer Rate:

   The maximum data transfer rate of ESP32 Wi-Fi is 150Mbps in 802.11n mode. The maximum data transfer rate of a mobile phone Wi-Fi hotspot depends on the specific Wi-Fi standard it supports. The transfer rate can range from a few Mbps up to several hundred Mbps.

5. Protocol:

   The communication protocol used in a mobile phone Wi-Fi hotspot is typically TCP (Transmission Control Protocol), which is a reliable and connection-oriented protocol for data transmission over networks like ESP32 Wi-Fi.

6. Power Consumption:

The ESP32 Wi-Fi module is designed to be power-efficient. In active mode, its current consumption typically ranges from 95mA to 240mA, depending on the Wi-Fi mode and signal strength. The power consumption of a mobile phone Wi-Fi hotspot varies depending on factors such as the device's battery capacity, Wi-Fi standard used, and the number of connected devices. On average, the power consumption can range from tens to a couple of hundred milliamps (mA) while actively providing Wi-Fi hotspot functionality.

4.4.4 Communication Issue

1. Impairment:

Impairment refers to the potential degradation of the Wi-Fi signal due to physical obstacles present in the environment. As the ESP32 communicates with thinger.io over a Wi-Fi network, physical barriers like walls, buildings, or other obstructions can attenuate the Wi-Fi signal. This attenuation weakens the strength of the Wi-Fi connection between the ESP32 and the access point or router, leading to weaker connections or even signal loss. To mitigate impairment, it is essential to ensure that the ESP32 is placed in an optimal location with minimal obstructions to maintain a stable and reliable Wi-Fi connection.

2. Security:

Security issues arise due to the potential vulnerability of the Wi-Fi network to unauthorized access, eavesdropping, and data breaches. ESP32 devices communicate with thinger.io over Wi-Fi, and if the network is not adequately secured, malicious individuals may attempt to gain unauthorized access to the network and its connected devices, including the ESP32. This unauthorized access could lead to potential data breaches, compromising sensitive information transmitted between the ESP32 and the thinger.io platform. To address security concerns, it is crucial to implement robust security measures such as using strong encryption protocols.

3. Transmission Delay:

One of the primary reasons for transmission delay is signal propagation. As the ESP32 sends data to the thinger.io platform over a Wi-Fi network, the data packets take time to propagate through the network, which can result in delays. Additionally, interference from other wireless devices operating in the same frequency band and network congestion due to a high number of connected devices can further contribute to transmission delays. The delay experienced by data packets as they travel through the network is known as latency. High latency can lead to slower data transfer rates and increased response times. Optimizing the Wi-Fi network and selecting less congested Wi-Fi channels can minimize transmission delay and latency,

4.4.5 Specification Data Sheet

Please refer appendix.

4.4.6 System design and optimization

Thinger.io allows users to select from various widget types to display in their dashboard, as illustrated in the picture below. In the dashboard, the time series chart widget displays the patient's body temperature and pulse value, while the text/value widget displays the patient's current location, positioning, and temperature status. The Google Maps widget is also utilised to determine the current position of the patient's IoT device. The dashboard's constraint is the latency in showing data received from the IoT device, which might take up to 7 seconds or more. Signal propagation is one of the key causes of transmission lag. Delays may occur when the ESP32 transmits data to the thinger.io platform via a Wi-Fi network because data packets need time to spread throughout the network.

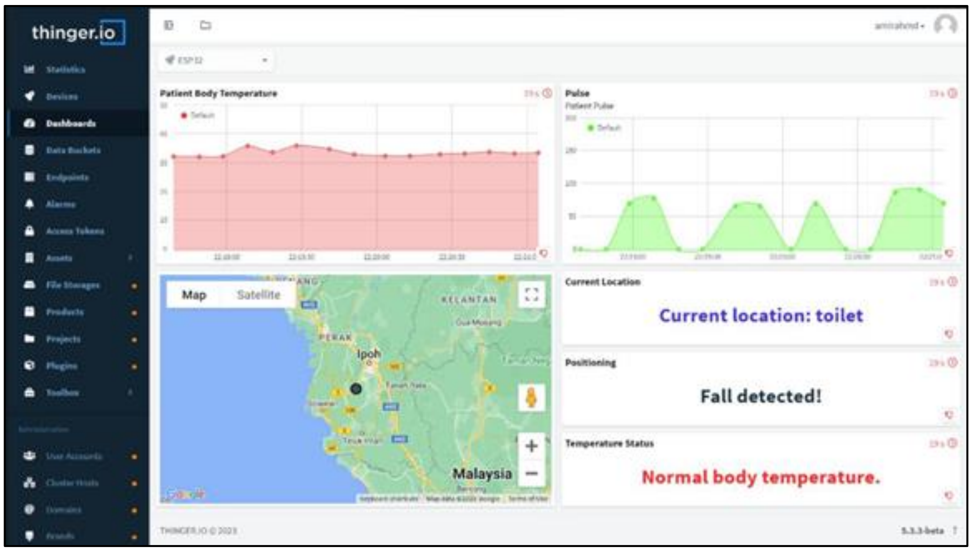


Figure 4.4.6: Dashboard of Thinger.io

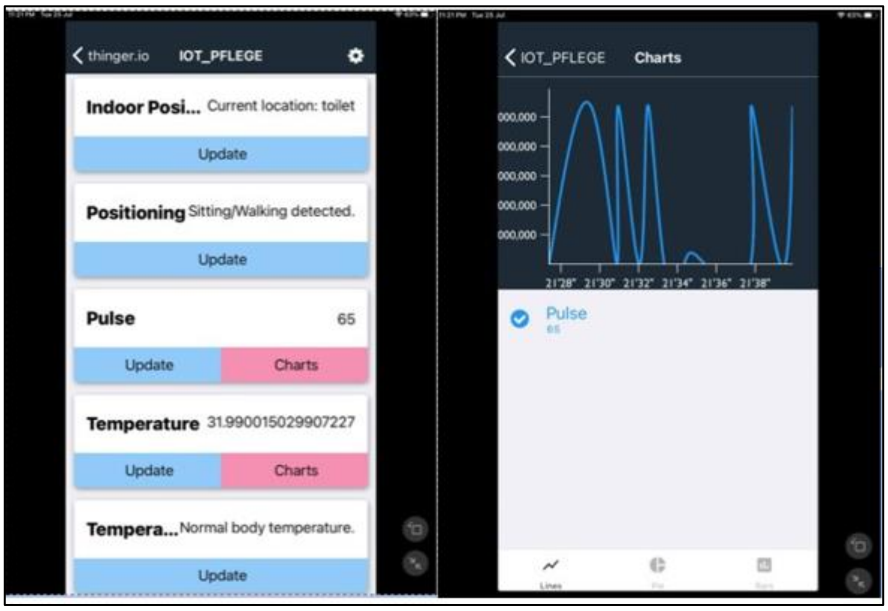

thinger.io
IOT_PFLEGE
Indoor Posi... Current location: toilet
Update
Positioning Sitting/Walking detected.
Update
Pulse 65
Update
Charts
Temperature 31.990015029907227
Update
Charts
Tempera... Normal body temperature.
Update
IOT_PFLEGE
Charts
21'28" 21'30" 21'32" 21'34" 21'36" 21'38"
Pulse
65

Figure 4.4.7: Interface of Thinger.io Mobile Application

Thinger.io also offers a mobile app that allows users to handle various platform aspects on their smartphone. The image above depicts how the interface appears on a smartphone. The app is available for download from Google Play and the App Store. It is possible to access the user account of the web console using this programme; all the devices registered automatically will be imported. It is important to note that by inputting the user credentials, user may also define the web domain of the instance you are working with, giving access to accounts of private Thinger.io instances.

The dashboard's constraint is the latency in showing data received from the IoT device, which might take up to 7 seconds or more. Apart from that, the restriction is that it takes a while for the mobile app to connect with the user account of the web interface. Signal propagation is one of the key causes of transmission lag. Delays may occur when the ESP32 transmits data to the thinger.io platform via a Wi-Fi network because data packets need time to spread throughout the network.

4.4.7 Data analytical method

Thinger.io also allows users to examine device statuses such as sent and received data in bytes, IP addresses, live data transmission from IoT devices, and daily data transmission. This functionality will allow caretakers or developers to examine the daily data transfer flow.

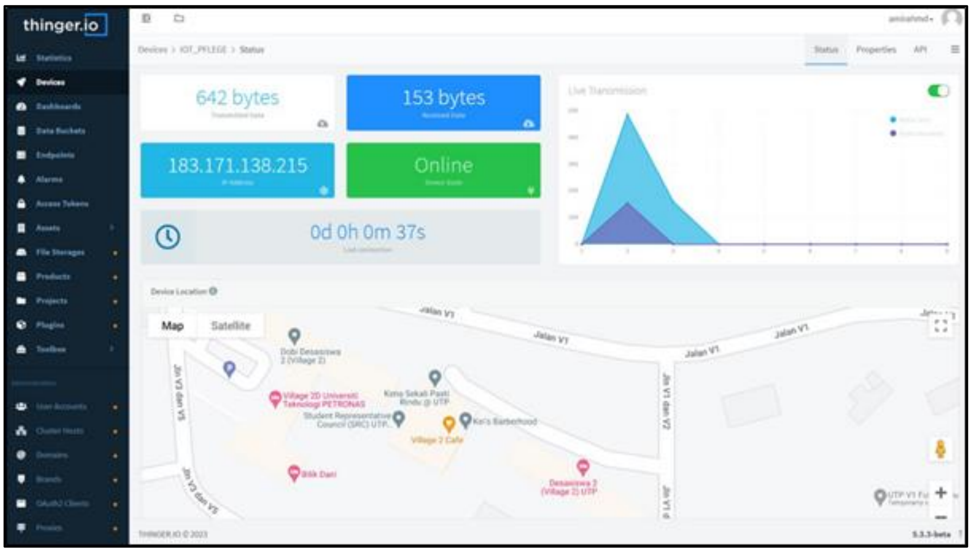


Figure 4.4.7.1: Device Status "Online"

The status of the IoT device can be changed to "Online" and “Offline” depending whether the IoT device is connected to the power supply and WiFi or not. The dashboard also displays the real-time transmission of received and transmitted data. "Transmitted data" and "received data" in Thinger.io reflect information sent from and received by an IoT device linked to the Thinger.io platform.

Aside from that, the dashboard displays the IP address of the connected IoT device transmitting data to Thinger.io. Each IoT device or client that connects to Thinger.io is issued a unique IP address, which acts as the device's network identity.

"Live transmission" on the Thinger.io dashboard denotes real-time data streaming or updates from connected IoT devices or sensors to the Thinger.io platform. It refers to the continuous flow of data from devices to the cloud, which provides users with current information and insights. The dashboard's "live transmission" function allows users to watch and monitor this data as it is received.

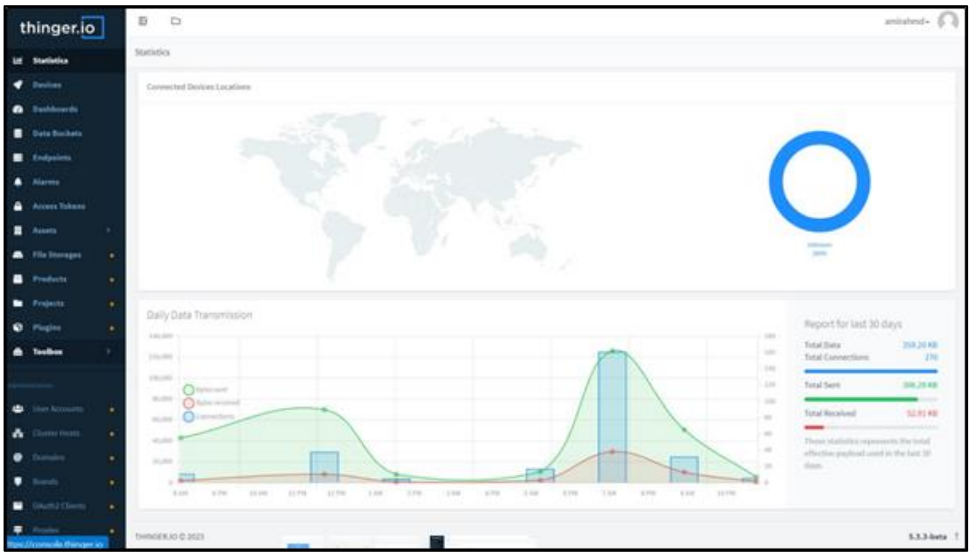


Figure 4.4.7.2: Device Status Daily Data Transmission

The "daily data transmission" on the Thinger.io dashboard as shown in figure above refers to the total quantity of data transferred by an IoT device or sensor to the Thinger.io platform in a 24-hour period. It reflects the total amount of data delivered from the device to the cloud in a single day.

4.5 Visualization

Visualisation is in which the dashboard should be created to offer an understandable and comprehensible visual representation of the data. Caretakers should be able to choose the data they want to observe on the old folks and how they want to view it through the customisation of the display.

User access and security should be taken into consideration when designing the dashboard for a certain patient. To guarantee that only caretakers see the data they are ratified to view, it should only be accessible to authorised users and offer granular access control. It should be simple to use, intuitive, and continuously updated in response to sensors input based on the patient’s condition.

By providing real-time visibility into key metrics and allowing analysation of information in real-time, it can be the game changer in to improving patient care. A visualisation platform's goal is to offer a comprehensive and user-friendly interface that shows the patient's current temperature, heart rate, indoor and outdoor position, and falling status in an aesthetically pleasing and easily accessible way. Platform thinger.io is designed to gather, assess, and present various healthcare-related data to help caretakers make better decisions and enhance overall healthcare outcomes.

4.6 Procedures

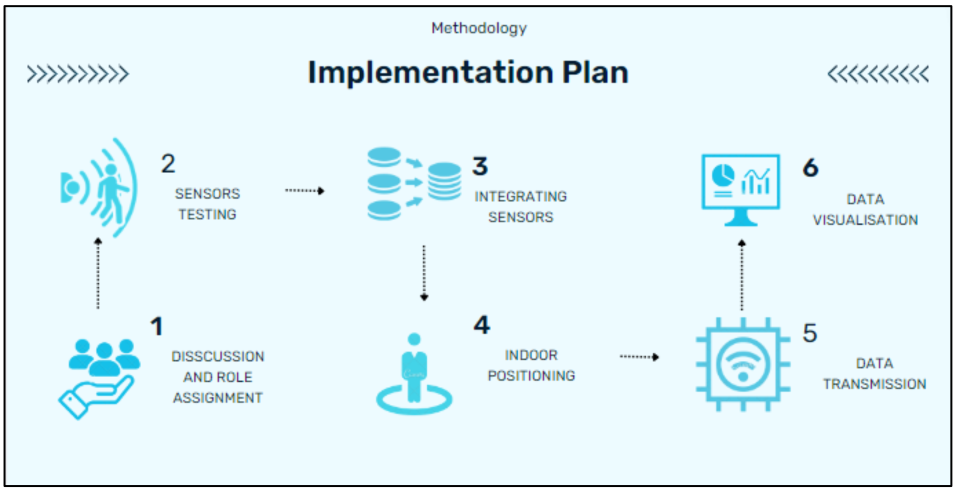


Figure 4.6: Implementation Plan of Pflege Old Folks Health Monitor Wristband Overview

Pflege Old Folks Health Monitor Wristband was effectively developed to provide continuous health monitoring, fall detection, indoor positioning, and cloud-based data visualisation. The following procedures were used to make effective Pflege Old Folks Health Monitor Wristband:

1. Discussion and role assignment
   - The project started with a discussion among the team members to plan and outline the development process of the Pflege Old Folks Health Monitor Wristband. During this phase, roles and responsibilities were assigned to each team member based on the existing subsystems. The team discussed the objectives, requirements, and challenges of the project, ensuring everyone was on the same page regarding the project's scope and goals.

2. Sensors unit testing
   - Each unit sensor, pulse sensor and temperature sensor, were thoroughly tested in isolation to ensure their accuracy and reliability. Unit testing helped identify and rectify any issues with the sensors before integrating them into the wristband. The team ensured that each sensor provided accurate and consistent readings, as these readings would form the foundation for monitoring the old folks' health conditions.

3. Integrating sensors

   - Once the sensors passed the unit testing phase, the team proceeded to integrate them into the wristband design. The goal was to create a compact and wearable wristband that could fit all the required sensors comfortably. Careful integration ensured that the wristband's design remained non-intrusive, comfortable, and practical for the patients to wear on a daily basis.

4. Indoor positioning
    - To achieve indoor positioning, the team explored various technologies and methods, such as Bluetooth beacon, triangulation, and Wi-Fi-based positioning systems. The chosen approach Esp32 WiFi with Machine Learning was then implemented and tested within the mapped Microprocessor Lab located in Block 23, Level 2 of UTP (representing Old Folk's home environment) to determine its accuracy and reliability in tracking the patient's indoor location. The indoor positioning system was designed to complement the fall detection feature and enhance the overall safety of the patients.

5. Data transmission
    - To enable real-time monitoring and record-keeping, a robust data transmission system was established. The team selected an appropriate communication protocol, such as Wi-Fi, and integrated it into the wristband's design. This allowed the health data collected from the sensors to be transmitted securely to a cloud server for analysis. The data transmission system was optimized for efficiency and reliability to ensure continuous and uninterrupted data flow.

6. Data visualisation
    - To present the collected health data in a user-friendly and actionable format, the team developed a data visualisation system. This system processed and analysed the data transmitted to using thinger.io and generated visual representations, such as graphs and charts, on a dashboard. Caretakers could access this dashboard to monitor the patients' health status in real-time.

4.7 FMEA

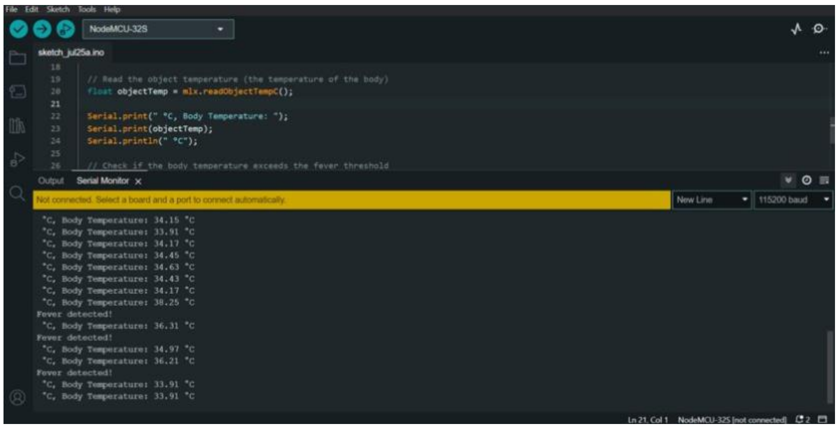
Please refer appendix.

5 RESULTS AND DISCUSSION

5.1 Subsystem 1 – Health Condition Sensing

5.1.1 Temperature Sensor

5.1.1.1 Result


File Edit Sketch Tools Help
NodeMCU-32S
sketch_jul25a.ino
Output
Serial Monitor
Not connected. Select a board and a port to connect automatically.
New Line
115200 baud
Ln 21, Col 1 NodeMCU-32S [not connected]

```
// Read the object temperature (the temperature of the body)
float objectTemp = mlx.readObjectTempC();

Serial.print(" °C, Body Temperature: ");
Serial.print(objectTemp);
Serial.println(" °C");

// Check if the body temperature exceeds the fever threshold
```

```
 °C, Body Temperature: 34.15 °C
 °C, Body Temperature: 33.91 °C
 °C, Body Temperature: 34.17 °C
 °C, Body Temperature: 34.45 °C
 °C, Body Temperature: 34.63 °C
 °C, Body Temperature: 34.43 °C
 °C, Body Temperature: 34.17 °C
 °C, Body Temperature: 38.25 °C
Fever detected!
 °C, Body Temperature: 36.31 °C
Fever detected!
 °C, Body Temperature: 34.97 °C
 °C, Body Temperature: 36.21 °C
Fever detected!
 °C, Body Temperature: 33.91 °C
 °C, Body Temperature: 33.91 °C
```

Figure 5.1.1.1: Temperature Sensor Serial Monitor Result in Arduino IDE

The code to detect body temperature begins by including two Arduino libraries which are the Wire.h and Adafruit_MLX90614.h. The Wire.h library is used for I2C communication while the Adafruit_MLX90614.h library provides functions to interface with the MLX90614 Non-Contact Infrared Temperature Sensor. By using the Adafruit_MLX90614 library, an instance of the Adafruit_MLX90614 class is created with the name "mlx" where this instance will be used to interact with the MLX90614 sensor. Besides that, we also defined a constant named "feverThreshold" which represents the temperature threshold for fever detection. The threshold can be adjusted according to specific requirements and medical guidelines where in our product, it is set to 36.5 degrees Celsius.

In the setup() function, the code initializes the serial communication with a baud rate of 115200, which allows us to communicate with the sensor data and view it on the Serial Monitor of the Arduino IDE. We also call the function mlx.begin() to initialize the MLX90614 sensor.

While in the loop() function, it is the main part of the code where the temperature measurements and fever detection are performed repeatedly. Within the loop, the code reads the object temperature which is body temperature in our product using the function in Adafruit_MLX90614 library which is mlx.readObjectTempC(). The code will then display the body temperature and check whether the body temperature exceeds the fever threshold defined earlier. If the body temperature is equal to or higher than the threshold value, the code prints "Fever detected!" to the Serial Monitor to indicate that the person is having high body temperature so that further actions like alerting the user or activating an alarm can be implemented.

5.1.1.2 Discussion

The MLX90614 is a non-contact infrared thermometer that can measure both ambient temperature and object (body) temperature based on the infrared radiation emitted by the object. The non-contact nature of the MLX90614 sensor makes it a suitable choice for body temperature detection, especially in scenarios where contact-based thermometers may not be desirable due to hygiene concerns or when rapid temperature screening is required for many individuals. We also experimented with some other temperature sensors such as the LM35 and MCP9700 and eventually decide to select MLX90614 due to its performance such as stability and more accurate data provided.

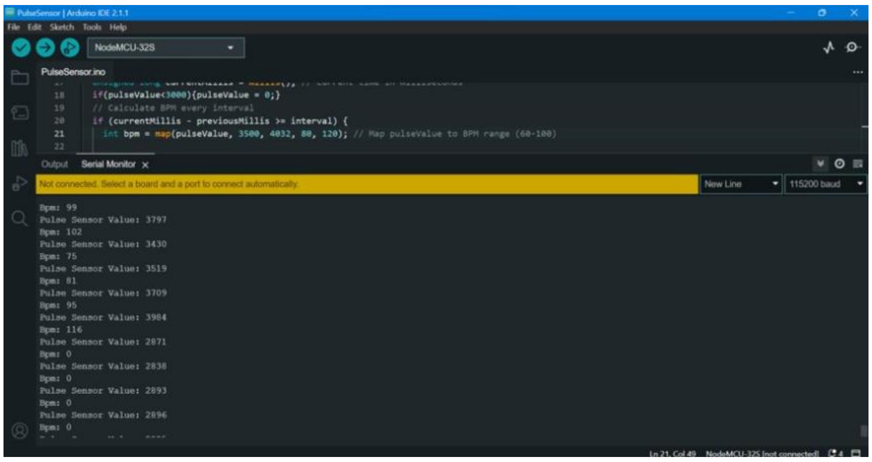
The Wire.h library in the Arduino IDE provides functions to handle the I2C (Inter-Integrated Circuit) communication protocol where it enables the ESP32 microcontroller and any connected devices to act as either a master or slave devices in an I2C communication network following master-slave configuration. The I2C protocol is a widely used serial communication protocol that allows multiple devices to communicate with each other using just two wires which are the SDA for data and SCL for clock. It is used for communication between various devices, such as sensors, displays, EEPROMs, and other microcontrollers.

The body temperature is directly collected using the function from Adafruit_ MLX90614 library which is readObjectTempC() and verify that the reading obtained is quite accurate for body temperature detection. Thus, we do not need do make any further calibration on the temperature sensor. To further verify that the results obtained from the temperature sensor is accurate, we can use another temperature sensor, or some other devices used for detecting body temperature for checking whether the values obtained are similar.

However, it is essential to note that the MLX90614 sensor measures the surface temperature of the object which is our skin in this case rather than the internal body temperature. Therefore, the detected temperature may not exactly match the core body temperature measured through other methods like oral or rectal thermometers. The threshold for fever detection should be carefully calibrated based on medical guidelines and the use case to ensure accurate results. Thus, we select the value 36.5 degree Celsius for fever threshold but not the actual thermometer readings used to indicate fever where the body temperature is 37.8 degree Celsius or higher for oral temperature and 37.2 degree Celsius or higher for armpit temperature.

5.1.2 Pulse Sensor

5.1.2.1 Result



```
  if(pulseValue<3000){pulseValue = 0;}
  // Calculate BPM every interval
  if (currentMillis - previousMillis >= interval) {
    int bpm = map(pulseValue, 3500, 4032, 80, 120); // Map pulseValue to BPM range (60-100)
```

Not connected. Select a board and a port to connect automatically.

```
Bpm: 99
Pulse Sensor Value: 3797
Bpm: 102
Pulse Sensor Value: 3430
Bpm: 75
Pulse Sensor Value: 3519
Bpm: 81
Pulse Sensor Value: 3709
Bpm: 95
Pulse Sensor Value: 3984
Bpm: 116
Pulse Sensor Value: 2871
Bpm: 0
Pulse Sensor Value: 2838
Bpm: 0
Pulse Sensor Value: 2893
Bpm: 0
Pulse Sensor Value: 2896
Bpm: 0
```

Figure 5.1.2.1: Pulse Sensor Serial Monitor Result in Arduino IDE

Unlike the other two sensors that use I2C protocol for communication, the pulse sensor uses the GPIO pin for communication and send the analogue data to the ESP32 microcontroller. However, the ESP32 does not have pins specially for Analog data input as in Arduino board and thus the GPIO34 is used as it also represents ADC6. Therefore, the first step for the code used to detect heart rate using pulse sensor is to define the GPIO pin number which GPIO 34 in this case to which the output of the pulse sensor is connected. Then, we define several variables such as the variable “pulseValue” to store the raw value read from the pulse sensor, the variable “previousMillis” to store the time in milliseconds when the last BPM calculation was performed, and the variable “interval” used to update interval in milliseconds for BPM calculation.

In the setup function that will be called once when the microcontroller starts, we initialize the serial communication and sets the pulsePin which is GPIO34 as an input so that the ESP32 microcontroller can read the analogue voltage from the pulse sensor.

Then, the loop function is executed repeatedly after the setup function. In the loop function, it reads the analogue voltage from the pulse sensor connected to the pulsePin using analogRead function and the values returned by the pulse sensor is expected to be in between 0 and 4095 for ESP32's 12-bit ADC corresponding to the voltage level.

During the calibration process of the pulse sensor, we found that the pulse sensor might produce a baseline value even when there's no heartbeat detected such as the value for pulseValue from 1900 to 2000. Therefore, the code sets the pulseValue to 0 if it falls below a threshold which is 2000 in this case after calibration to avoid displaying incorrect BPM values.

The code will also check if the time since the last BPM calculation (previousMillis) is greater than or equal to the defined interval which is 1000 milliseconds. If yes, the value of pulseValue variable is mapped from its original range which is 3000 to 4095 to the BPM range for human which is 60 to 130. This mapping is done to approximate the BPM value based on the pulse sensor reading and the BPM value is displayed.

5.1.2.2 Discussion

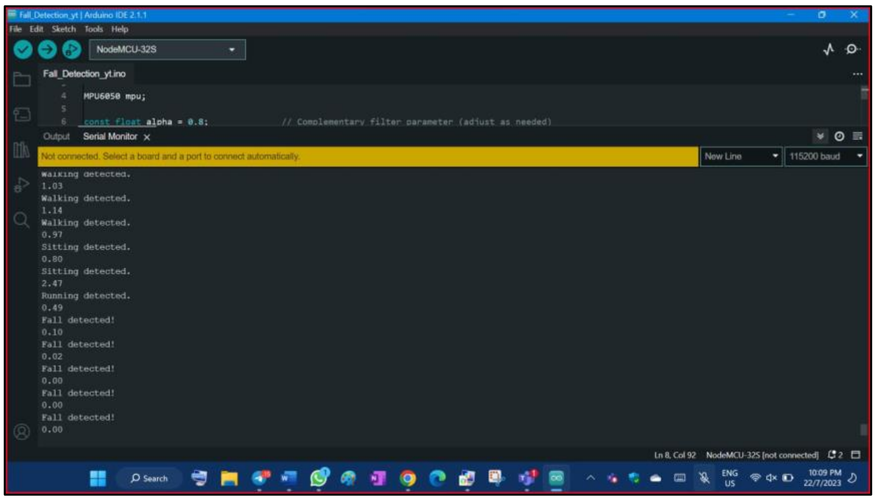
Heart rate monitoring is an essential aspect of healthcare monitoring as it provides valuable insights into an individual's cardiovascular health and physical activity levels. In our code, we utilize the analogue readings from the pulse sensor to calculate and display the heart rate (BPM) after multiple times of calibration on the pulse sensor readings. However, no matter how many times of calibration have been done, the effectiveness of monitoring heart rate is still depending on the accuracy and consistency of the pulse sensor's raw readings and the reliability of the ESP32's analogue-to-digital converter (ADC).

The benefit of using the pulse sensor XD-58C is due to its low-cost solution where the use of an ESP32 microcontroller makes the heart rate detection system cost-effective and accessible for various applications. We can also customizable the threshold value on our own through calibration process based on the real-life condition for the purpose of filtering baseline noise, providing some level of adaptability to different environmental conditions.

However, the results obtained will not be as accurate as expected due to the basic linear mapping algorithm used to approximate BPM values as it might not be the most accurate method for heart rate calculation and more advanced signal processing algorithms are required for precise and reliable results. Besides that, the reliability of the pulse sensor can be influenced by external noise and environmental conditions and thus it is not able to achieve the critical applications or medical-grade heart rate monitoring.

5.1.3 Accelerometer Sensor

5.1.3.1 Result


Fall_Detection_yt | Arduino IDE 2.1.1
File Edit Sketch Tools Help
NodeMCU-32S
Fall_Detection_yt.ino
Output
Serial Monitor
New Line
115200 baud
Ln 8, Col 92 NodeMCU-32S [not connected]

```
MPU6050 mpu;

const float alpha = 0.8;           // Complementary filter parameter (adjust as needed)
```

Not connected. Select a board and a port to connect automatically.

```
Walking detected.
1.03
Walking detected.
1.14
Walking detected.
0.97
Sitting detected.
0.80
Sitting detected.
2.47
Running detected.
0.49
Fall detected!
0.10
Fall detected!
0.02
Fall detected!
0.00
Fall detected!
0.00
Fall detected!
0.00
```

Figure 5.1.3.1: Accelerometer Sensor Serial Monitor Result in Arduino IDE

To write the program for fall detection by using accelerometer MPU6050, first we need to include the libraries required where the code starts by including the Wire.h library which is used for I2C communication and the MPU6050.h library which is used to interface with the MPU6050 accelerometer and gyroscope sensor.

Then, several important constants are defined to fine-tune the fall detection algorithm. The variable alpha is a parameter for the complementary filter used to fuse accelerometer and gyroscope data. The complementary filter is a common method for improving the accuracy of orientation measurements. For the threshold values such as the thresholdFall, thresholdSitting, thresholdWalking, and thresholdRunning variables, they were used to detect different activities based on the magnitude of the acceleration vector. These values can be adjusted according to specific use cases and sensitivity requirements. Lastly, the variable lastAccelMagnitude that store the previous magnitude of the acceleration vector, which will be used for the complementary filter calculation is initialized to 0.

Inside the loop function, the accelerometer and gyroscope data are read from the MPU6050 sensor using the getMotion6() function for the purpose of retrieving sensor data. The raw accelerometer values are then converted to floating-point values (float). Then, the code applies a complementary filter to combine the gravity component from the accelerometer with the orientation data from the gyroscope where the accelMagnitude variable represents the magnitude of the acceleration vector after applying the filter. If the value of accelMagnitude is less than the threshold values, it detects respective activities based on the different types of threshold values.

Thus, the threshold value takes a significant role in fall detection, and we have to calibrate the threshold values slowly based on real-life cases so that the program can detect the activity of a person accurately.

5.1.3.2 Discussion

The fall detection algorithm using the MPU6050 sensor presented in the code demonstrates a simple yet effective approach to detect different activities based on the magnitude of the acceleration vector. This section discusses the key aspects of the algorithm, its potential strengths, limitations, and possible areas for improvement.

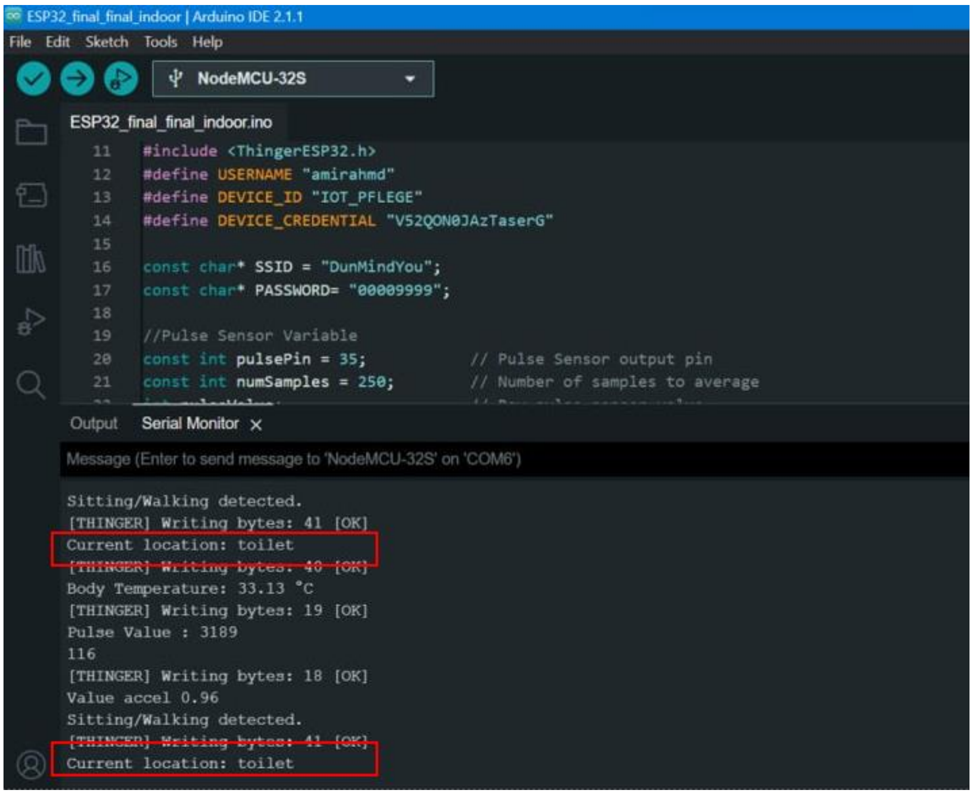
The algorithm utilizes a complementary filter to fuse data from the accelerometer and gyroscope, improving the accuracy of orientation measurements. By combining the gravity component from the accelerometer and orientation data from the gyroscope, the algorithm can effectively eliminate noise and drift in the accelerometer readings. The choice of using a complementary filter is appropriate for motion tracking applications like fall detection, where real-time and accurate orientation data are crucial. Other than complementary filtering, the algorithm also uses different threshold values such as thresholdFall and thresholdSitting to classify activities based on the magnitude of the acceleration vector. Threshold-based detection is a simple and straightforward method that works well in scenarios where the activities have distinct ranges of acceleration magnitudes. It enables the system to identify falls, sitting, walking, and running, making it suitable for basic activity monitoring applications.

One of the notable strengths of the algorithm is its real-time responsiveness. The code continuously reads sensor data and processes it to detect activities on-the-fly. This capability makes it suitable for applications where prompt detection and response are essential, especially in fall detection scenarios, where immediate action may be critical.

However, the accelerometer sensor can only cope with limited fall detection scenarios as the algorithm only focuses on detecting falls based on acceleration magnitude. The real-world fall scenarios can be much more complex and may involve specific patterns in the acceleration data or other sensor information.

5.2 Subsystem 2 – Indoor Positioning

5.2.1 Result



ESP32_final_final_indoor | Arduino IDE 2.1.1
File Edit Sketch Tools Help
NodeMCU-32S
ESP32_final_final_indoor.ino
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU-32S' on 'COM6')

```
#include <ThingerESP32.h>
#define USERNAME "amirahmd"
#define DEVICE_ID "IOT_PFLEGE"
#define DEVICE_CREDENTIAL "V52QON0JAzTaserG"

const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";

//Pulse Sensor Variable
const int pulsePin = 35;           // Pulse Sensor output pin
const int numSamples = 250;        // Number of samples to average
```

```
Sitting/Walking detected.
[THINGER] Writing bytes: 41 [OK]
Current location: toilet
[THINGER] Writing bytes: 40 [OK]
Body Temperature: 33.13 °C
[THINGER] Writing bytes: 19 [OK]
Pulse Value : 3189
116
[THINGER] Writing bytes: 18 [OK]
Value accel 0.96
Sitting/Walking detected.
[THINGER] Writing bytes: 41 [OK]
Current location: toilet
```

Figure 5.2.1.18: Indoor Positioning Serial Monitor Result in Arduino IDE

The process to achieve indoor positioning based on Wi-Fi is as indicated by the flow chart below. Several Arduino libraries such as the “WiFi.h”, “eloquent.h” and “eloquent/networking/wifi/WifiScanner.h” are required in order to achieve indoor positioning.

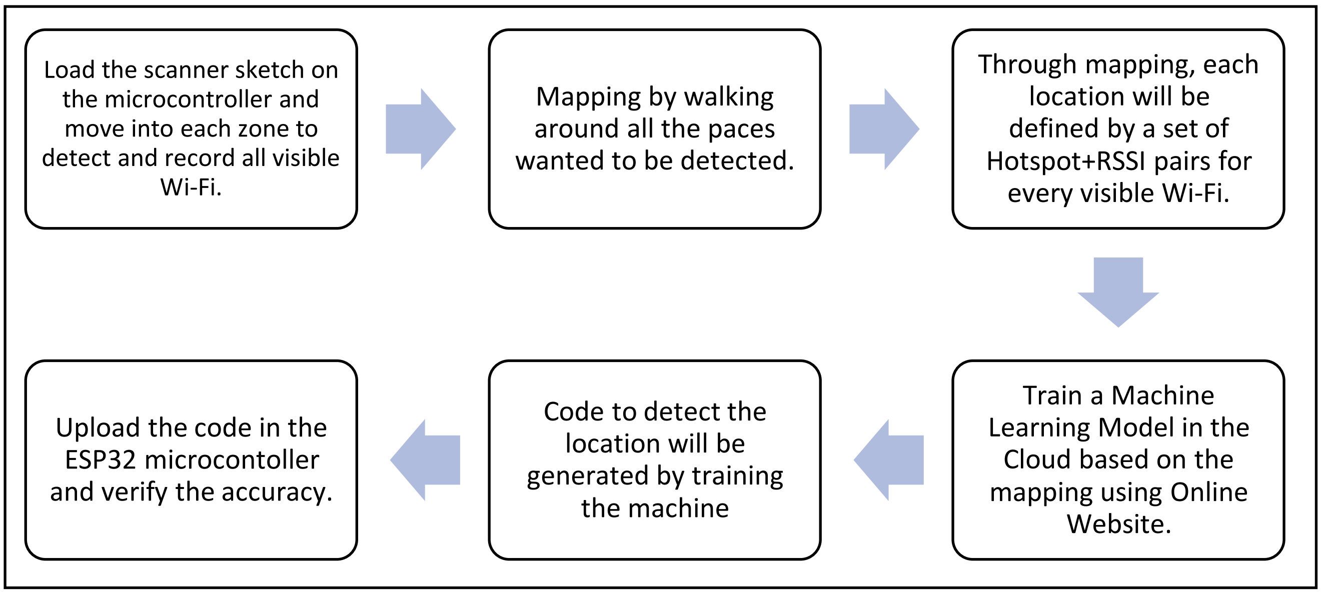

Load the scanner sketch on the microcontroller and move into each zone to detect and record all visible Wi-Fi.
Mapping by walking around all the paces wanted to be detected.
Through mapping, each location will be defined by a set of Hotspot+RSSI pairs for every visible Wi-Fi.
Train a Machine Learning Model in the Cloud based on the mapping using Online Website.
Code to detect the location will be generated by training the machine
Upload the code in the ESP32 microcontoller and verify the accuracy.

Figure 5.2.1.2: Flowchart of Indoor Positioning

For every location that we wanted to recognize and map it, we need to position ourselves in the centre and enter the name for the location. Then, we can start to move around by going to the corners or walking along the walls for the specific location and lastly go back to the centre point. For each point we mapped, we will be obtaining the and for every visible Wi-Fi for example "toilet: {"TP-LINK_9118":-56,"AnchorAP1":-63,"EnergisingFutures":-64,"AnchorAP3":-69,"EnergisingFutures":-72,"NasiKandaq-2.4Ghz":-73,"EnergisingFutures":-81,"LEARNING HUB 05":-91}". We should record at least mapped 10-15 point for each location for the indoor positioning to be accurate. The diagram below shows the process of mapping the location where the actual location is the Microprocessor Lab located in Block 23, Level 2 of UTP and we customize the area of Microprocessor Lab to three locations for indoor positioning which are kitchen, room, and toilet.

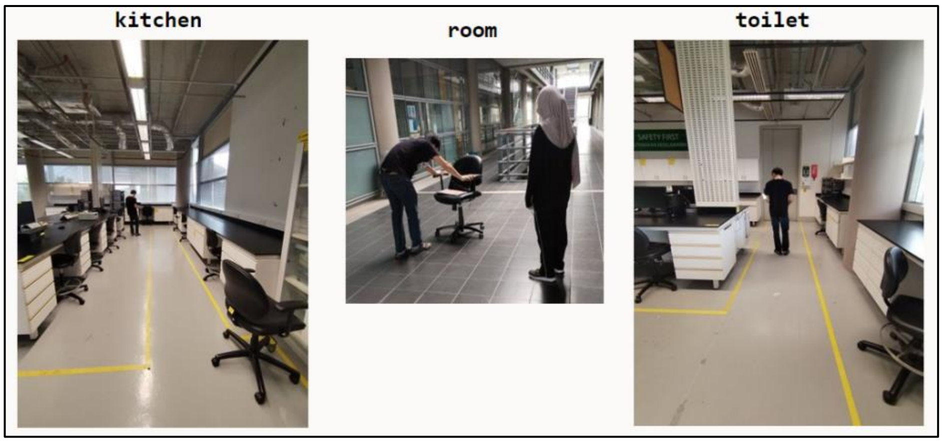


Figure 5.2.1.39: Mapping Locations for Old Folks Home

5.2.2 Discussion

To further explain on the mechanism of the indoor positioning, it is based on the ability of the ESP32 microcontroller to detect all the available Wi-Fi which will act as the access points. Therefore, we need several hotspots that will emit radio signal from fixed location to implement this indoor positioning system. The devices that might help will be the repeater or router where the function of a repeater is to extend the range of a network signal, while a router function to manage the network traffic and enables communication between devices within the network. Then, the microcontroller will hear the strength of these signals and use a Machine Learning model to detect in which location it currently is.

The purpose of why Wi-Fi signal is selected for our indoor positioning system instead of using the other type of radio waves such as Bluetooth, ZigBee and Ultra-wideband is due to its widespread availability, high data transmission rates, and the extensive infrastructure of Wi-Fi access points in indoor environments which allows accurate and reliable location tracking of devices.

The indoor positioning system relies on several factors that affect its accuracy and performance:

1. Type of Radio
   - The specific Wi-Fi radio frequency used for example either the Wi-Fi of 2.4GHz or 5GHz can influence the signal's propagation characteristics and penetration through obstacles.

2. Number of Access Points
    - The more access points available in the area, the higher the density of reference points for positioning, which leads to better accuracy.

3. Proximity to Access Points
    - The closer the access points to the device being located, the stronger signals can be provided by the access points and thus better location estimation.
    - The location of the Access Points should also be fixed so that the strength of Wi-Fi at each location mapped is constant.

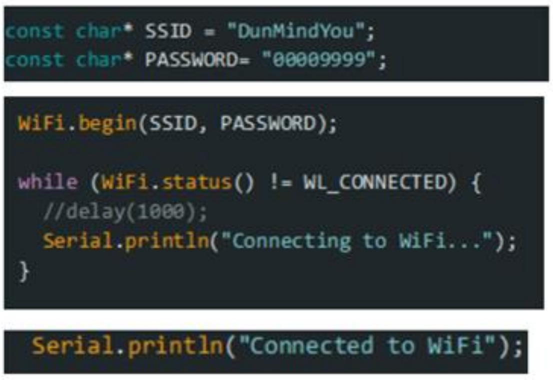
4. Spatial Distribution of Access Points
    - The strategic placement of access points throughout the indoor space can optimize signal coverage, minimizing signal blind spots and improving positioning accuracy.

5. Obstacles and Interference
    - Physical obstacles such as walls, furniture, and other can attenuate and reflect Wi-Fi signals which potentially leads to signal fluctuations and inaccuracies in location estimation.

5.3 Subsystem 3 - Data Transmission and Interface

5.3.1 Wi-Fi Connection Setup

Result



```
const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";
```

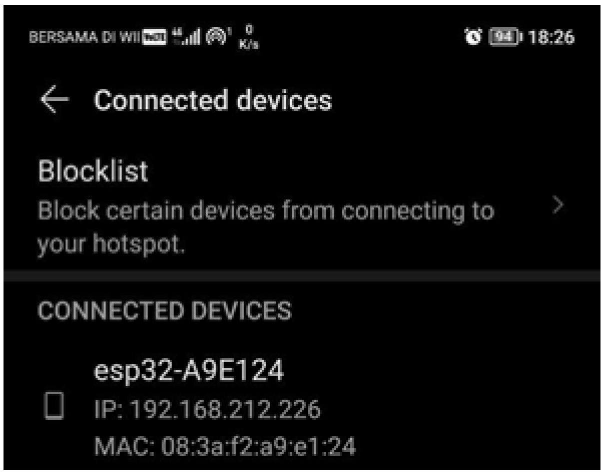
```
WiFi.begin(SSID, PASSWORD);

while (WiFi.status() != WL_CONNECTED) {
  //delay(1000);
  Serial.println("Connecting to WiFi...");
}
```

```
Serial.println("Connected to WiFi");
```

Figure 5.3.1.1: Wi-Fi Connection Setup

The figure above shows how the Wi-Fi is set up in the Arduino sketch. The Arduino library of "WiFi.h" must connect the ESP32 board with the selected Wi-Fi mobile hotspot. The SSID (service set identifier) and password of the selected Wi-Fi mobile hotspot are declared, which are "DunMindYou" and "00009999", respectively. In the void setup function, Wifi.begin() initiates a connection to a Wi-Fi network. Then, it checks the connection status using Wifi.status() and prints the "Connected to WiFi" when the connection is successful.

Once the connection is successful, the name of microcontroller ESP32 (esp32-A9E124) will appear in the list of connected devices of Wi-Fi mobile hotspots, as shown in the figure below.



BERSAMA DI WII 18:26

Connected devices

Blocklist

Block certain devices from connecting to your hotspot.

CONNECTED DEVICES

esp32-A9E124

IP: 192.168.212.226

MAC: 08:3a:f2:a9:e1:24

Figure 105.3.1.2: Connected Devices of Wi-Fi Mobile Hotspots

Discussion

A successful Wi-Fi connection is the main result of connecting an ESP32 to a Wi-Fi mobile hotspot. The ESP32 will connect if the credentials (SSID and password) are correct, and the Wi-Fi mobile hotspot is in range and accessible. Wi-Fi has a larger range of 15 metres and can reach hundreds of feet in open areas. If the Wi-Fi mobile hotspot is within range, the ESP32 may be put in various locations. However, the dependability of the Wi-Fi connection is influenced by the mobile hotspot's signal strength, network coverage, and user mobility. If the hotspot signal becomes weak or unreliable while travelling, the ESP32's connection may be disrupted.

One of the factors of weak Wifi signal strength is overlapping wireless networks from neighbouring devices in the immediate area, such as neighbouring companies or public hotspots, which might contribute to poor signal strength. Other wireless devices, such as cordless phones and Bluetooth, might also cause interference with a WiFi signal. Because these devices use the same frequency as WiFi signals, they create interference.

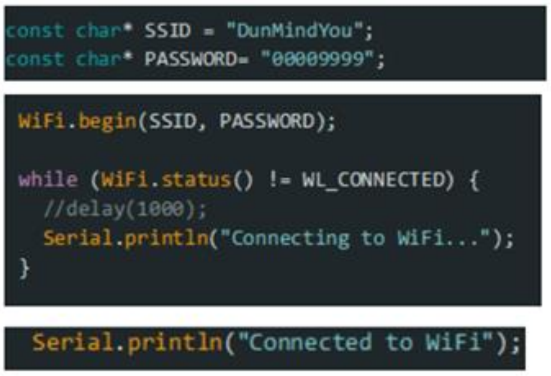
In terms of security, ensure the hotspot is safe and has a strong password. Using an unprotected hotspot may expose the ESP32 to unauthorised access or attacks. Attackers can intercept and analyse data packets from the exposed Wi-Fi network using packet sniffing tools. This enables them to obtain sensitive information such as login passwords, personal information, or financial information. Then, when the ESP32 is linked to a Wi-Fi mobile hotspot, it receives internet access via the mobile network. This allows the ESP32 to transmit and receive data via the internet, which is useful for various IoT and communication applications.

5.3.2 Extract data from IoT devices to Thinger.io

Result

To extract raw data from an Arduino.ide and send it to Thinger.io using an ESP32 is as shown below:

1. An account on Thinger.io is created to set up a project.
2. The Arduino sketch is written to collect raw data from sensors, set up WiFi, and connect to the internet. Figure below shows on how the WiFi is set up in the Arduino sketch.

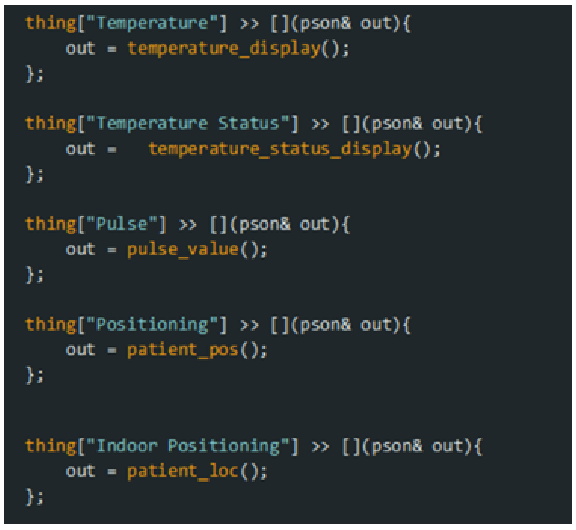


```
const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";
```

```
WiFi.begin(SSID, PASSWORD);

while (WiFi.status() != WL_CONNECTED) {
  //delay(1000);
  Serial.println("Connecting to WiFi...");
}
```

```
Serial.println("Connected to WiFi");
```

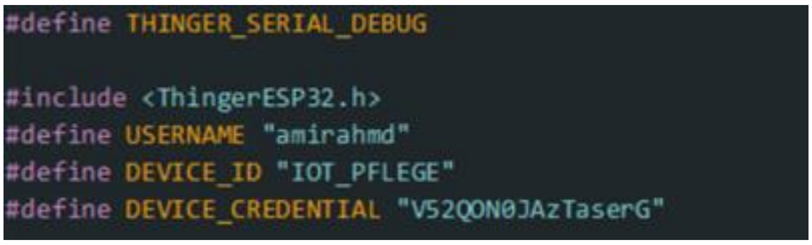
Figure 5.3.2.1: Code from Arduino that allow Thinger.io to read a sensor value using the operator “>>”.



```
thing["Temperature"] >> [](pson& out){
    out = temperature_display();
};

thing["Temperature Status"] >> [](pson& out){
    out =   temperature_status_display();
};

thing["Pulse"] >> [](pson& out){
    out = pulse_value();
};

thing["Positioning"] >> [](pson& out){
    out = patient_pos();
};


thing["Indoor Positioning"] >> [](pson& out){
    out = patient_loc();
};
```

Figure 5.3.2.2: Thinger.io Transmission

2. The Thinger.io library is included at the beginning of the code, where Thinger.io is set up with the account credentials, such as username, device ID, and device credentials, as shown in the figure below.



```
#define THINGER_SERIAL_DEBUG

#include <ThingerESP32.h>
#define USERNAME "amirahmd"
#define DEVICE_ID "IOT_PFLEGE"
#define DEVICE_CREDENTIAL "V52QON0JAzTaserG"
```

Figure 5.3.2.3: Thinger.io Credentials

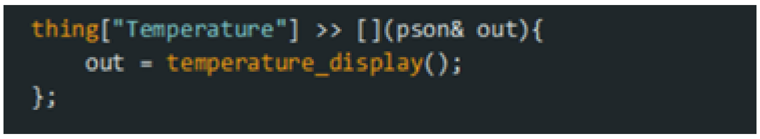
4. The code is compiled and uploaded to the ESP32 board.
5. Once the device is on and the ESP32 is connected to WiFi, the Thinger.io account can be accessed, and the data sent by the ESP32 can be viewed and visualized in real time.

Discussion

Several processes are involved in extracting data from IoT devices to Thinger.io using an ESP32, which enables the ESP32 to gather sensor data and communicate it to the Thinger.io cloud platform. IoT devices have a variety of sensors to collect environmental data. Several GPIO pins on the ESP32 may connect to various sensors. An Analogue Digital Converter (ADC) is, for instance, Pin 35, which is connected to the Pulse sensor. The ESP32 can transform analogue voltage signals from sensors into digital values that can be analysed and utilised in code thanks to the ADC.

The ESP32 receives sensor data and saves it in variables for further processing using the proper libraries or protocols. In term of Protocol, pin SCL (serial clock) and SDA(serial data)pin is used for the I2C protocol. Inter-Integrated Circuit. Allows the microcontroller to interact with other I2C-compatible devices in a networked environment.

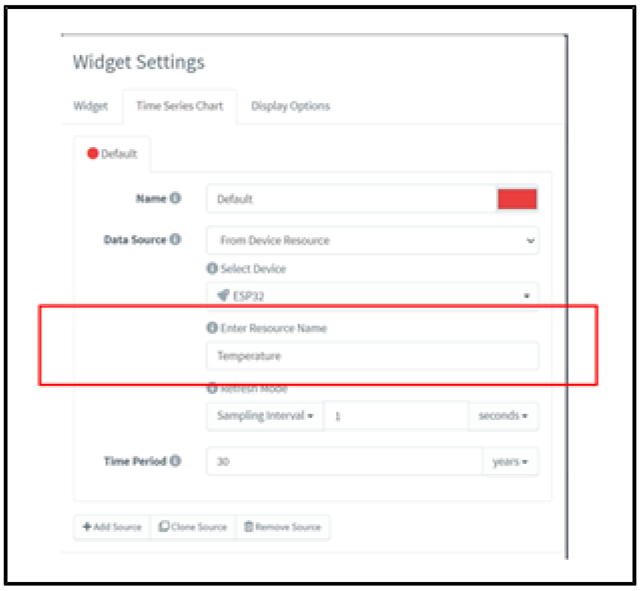
A Thinger.io account must be created on their platform to access the service. After creating a device, the user must obtain device credentials (Device ID, Device Credential). The ESP32 will be authenticated with the Thinger.io platform using these credentials. The ESP32 uses the initialised Thinger.io library to periodically communicate the collected sensor data to the Thinger.io platform. The operator ">>" is used to achieve the transmission of value from devices to Thinger.io. For example, code below shows that the "Temperature" resource is created on the Thinger.io platform using this line of code. The lambda function is called when the platform requests data for this resource, and it calls the "temperature_display()" method to get the temperature reading. The "out" object is then used to return the returned value to the platform.



```
thing["Temperature"] >> [](pson& out){
    out = temperature_display();
};
```

Figure 5.3.2.4: Syntax to send temperature value to Thinger.io

1. thing["Temperature"]**:** The "thing" in Thinger.io allows for providing resources (data endpoints) that the platform may access and control. "Temperature" refers to a particular resource containing temperature information in the Widget Settings as shown in figure below. Which will be selected to display only the temperature value received from the IoT devices.



Widget Settings

Widget | Time Series Chart | Display Options

Default

Name: Default

Data Source: From Device Resource

Select Device: ESP32

Enter Resource Name: Temperature

Refresh Mode: Sampling Interval 1 seconds

Time Period: 30 years

Figure 5.3.2.5: Selection to display Temperature value in Widget Setting

3. >>: This operator is used to construct an output handler for the supplied resource. When the platform requests data for this resource, it indicates the data that should be supplied.
4. [](pson& out): The "pson" data type, which manages data serialisation and transmission, is probably unique to the Thinger.io framework.
5. out = temperature_display()**:** Assigned to the "out" object, which will be returned to the Thinger.io platform as the data for the "Temperature" resource, is the obtained temperature value.

5.3.3 Thinger.io Data Transmission

Result and Discussion

Figures 5.3.3.1, 5.3.3.2 and 5.3.3.depict the interface connection device and data management where the caretaker may monitor and manage the data usage of their IoT device. This information is critical for assessing the IoT system's efficiency, optimising data utilisation, and ensuring that data plans and resources are effectively assigned to meet the application's demands.

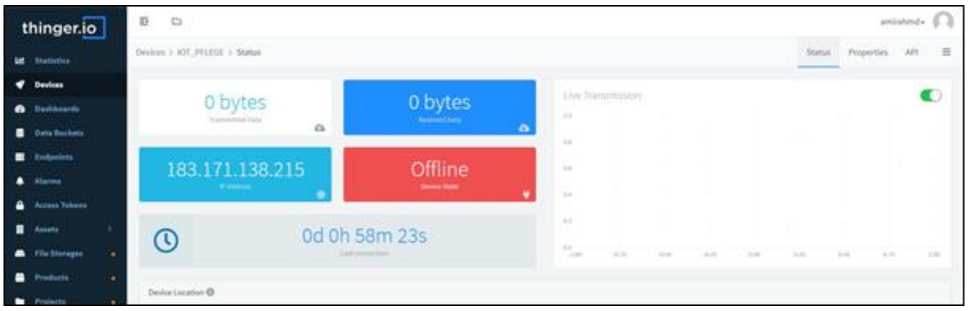


Figure 5.3.3.1: Status of device is offline, 0 bytes of transmitted data.

Figure 5.3.3.1 shows that the status of device is offline. This shows that the device is currently not connected to the targeted Wifi, or it is not connected to a power supply. Thus, Thinger.io unable to receive data from the device or send data to it. The live transmission of the received and transmitted data also did not appear.

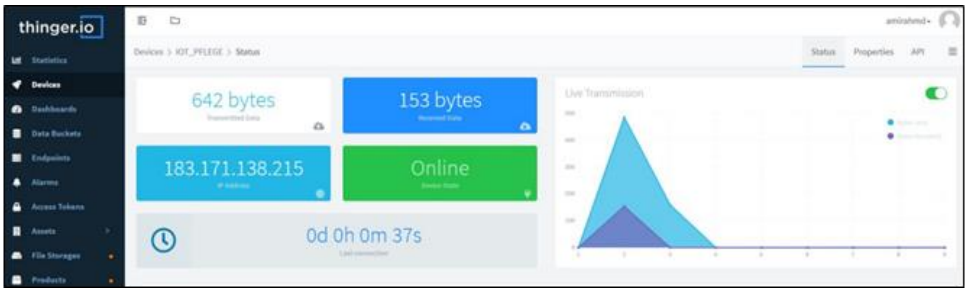


Figure 5.3.3.2: Status of device is online, 642 bytes of transmitted data and 153 bytes of received data.

The status of the IoT device is changed to "Online" once it is connected to the power supply and successfully connects to WiFi. Thinger.io can both receive and transmit data to the gadget. The dashboard also displays the real-time transmission of received and transmitted data. "Transmitted data" and "received data" in Thinger.io reflect information sent from and received by an IoT device linked to the Thinger.io platform. The presently transmitted data in the figure is 643 bytes, which relates to information such as the temperature sensor, pulse sensor, current position, current indoor location, and current temperature status transferred from the device to Thinger.io through a network. It entails delivering data from a source to a destination, which may be Thinger.io, capable of receiving and processing the data. Thus, transmitted data in the context of the Internet of Things (IoT) is data gathered by IoT devices or sensors and transferred to Thinger.io for storage, processing, analysis, and visualisation.

While the received data number is 153, as indicated on the dashboard, refers to the information or data that Thinger.io collects or receives from various IoT devices or sensors. Sensors on IoT devices collect data from patients, which is then transferred and received by Thinger.io for processing, analysis, storage, and action. The value of sent and received data might differ for various reasons. Some of the frequent reasons why sent and received data values may differ:

1. Communication protocols that govern information to the delivered data are frequently used when communicating between devices or systems. Overheads, footers, checksums, and other control information required for data integrity and reliable transmission are examples of overheads.
2. Network devices or servers may use data filtering or aggregation techniques to decrease the amount of data delivered over the network.

3. Due to network difficulties, interference, or congestion, data packets may be lost or damaged during transmission.
4. Communication protocols frequently include acknowledgement messages and handshakes between the sender and recipient to ensure trustworthy data transmission. The transmission data includes these control messages but not the received data.

Aside from that, the dashboard displays the IP address of the connected IoT device transmitting data to Thinger.io, which is 183.171.138.215. Each IoT device or client that connects to Thinger.io is issued a unique IP address, which acts as the device's network identity. An IP address, or Internet Protocol address, is a unique numerical designation provided to each device connected to a computer network that communicates using the Internet Protocol. It allows devices on a network to be identified and located. IP addresses are critical in allowing device connectivity and routing data packets over the internet.

When an IoT device connects to Thinger.io, it delivers data over the internet to the platform. The network assigns the gadget an IP address to ease communication. The device and platform utilise this IP address to transport data packets between the device and the Thinger.io server.

"Live transmission" on the Thinger.io dashboard denotes real-time data streaming or updates from connected IoT devices or sensors to the Thinger.io platform. It refers to the continuous flow of data from devices to the cloud, which provides users with current information and insights. The dashboard's "live transmission" function allows users to watch and monitor this data as it is received.

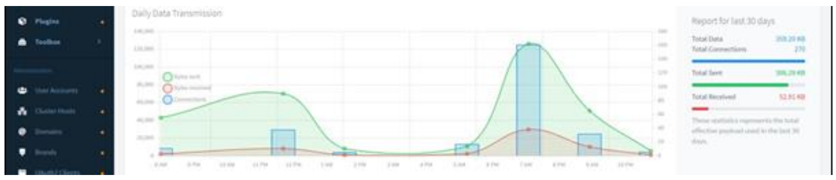


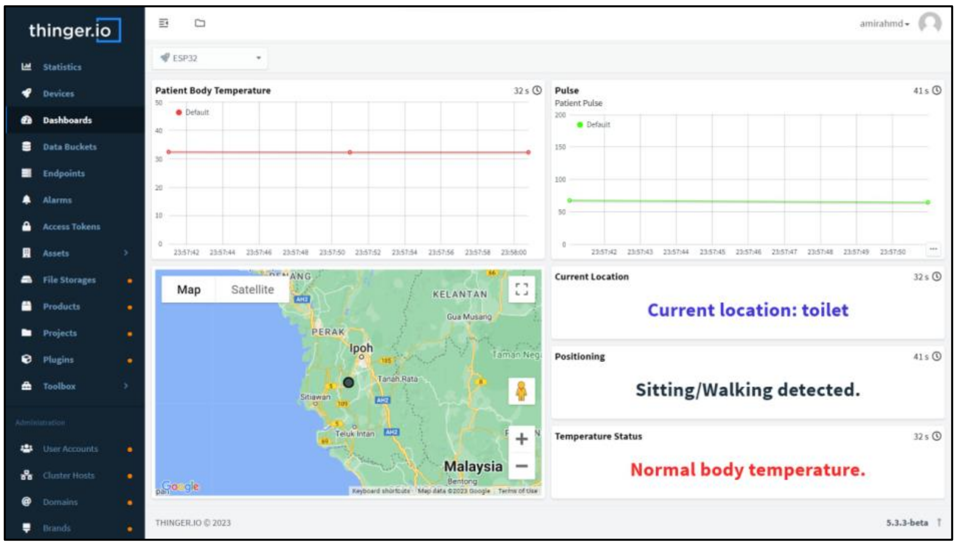
Figure 5.3.3.4: Daily Data Transmission

The "daily data transmission" on the Thinger.io dashboard as shown in figure above refers to the total quantity of data transferred by an IoT device or sensor to the Thinger.io platform in a 24-hour period. It reflects the total amount of data delivered from the device to the cloud in a single day.

5.4 Subsystem 4 - Visualisation

5.4.1 Result

5.4.1.1 Output of Combined Sensors

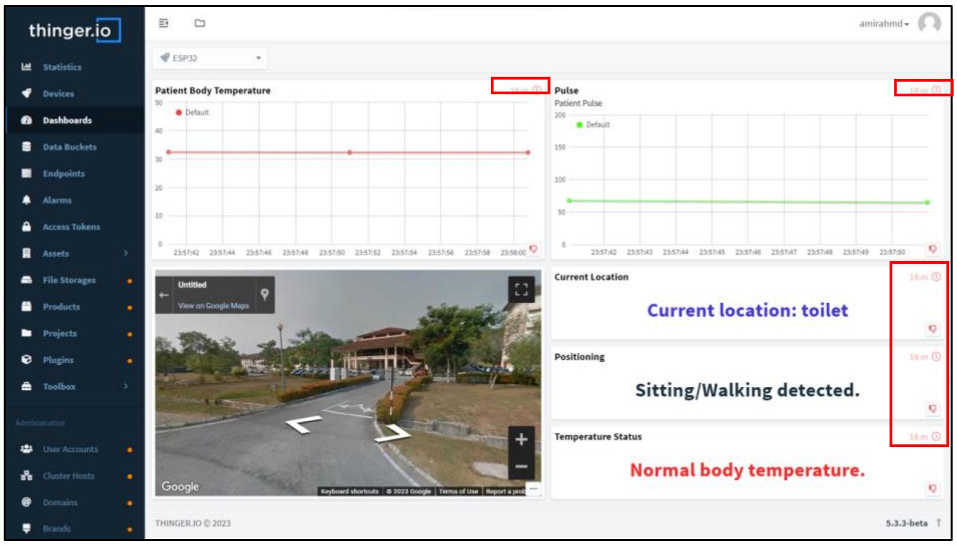


Current location: toilet

Sitting/Walking detected.

Normal body temperature.

Figure 5.4.1.1.1: Finalised Pflege Dashboard Web Application

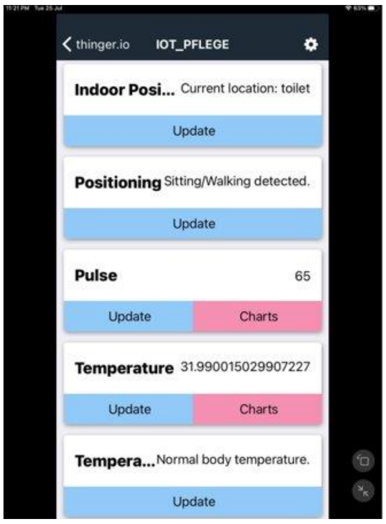


Current location: toilet

Sitting/Walking detected.

Normal body temperature.

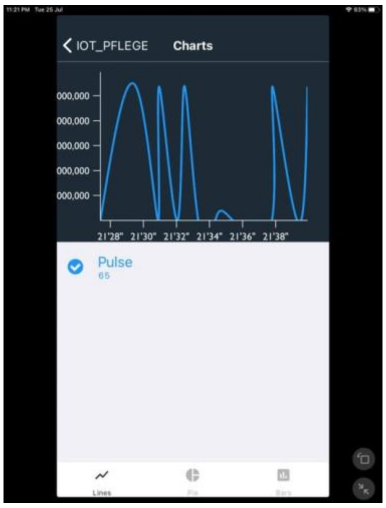
Figure 5.4.1.1.2: Street View shown on Pflege Dashboard and Timeout Detected



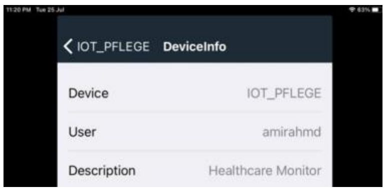
thinger.io IOT_PFLEGE

Indoor Posi...	Current location: toilet
Positioning	Sitting/Walking detected.
Pulse	65
Temperature	31.990015029907227
Tempera...	Normal body temperature.

Figure 5.4.1.1.3: Finalised Pflege Dashboard Mobile Application



5.4.1.1.4: Sample Pulse Rate Graph



IOT_PFLEGE DeviceInfo

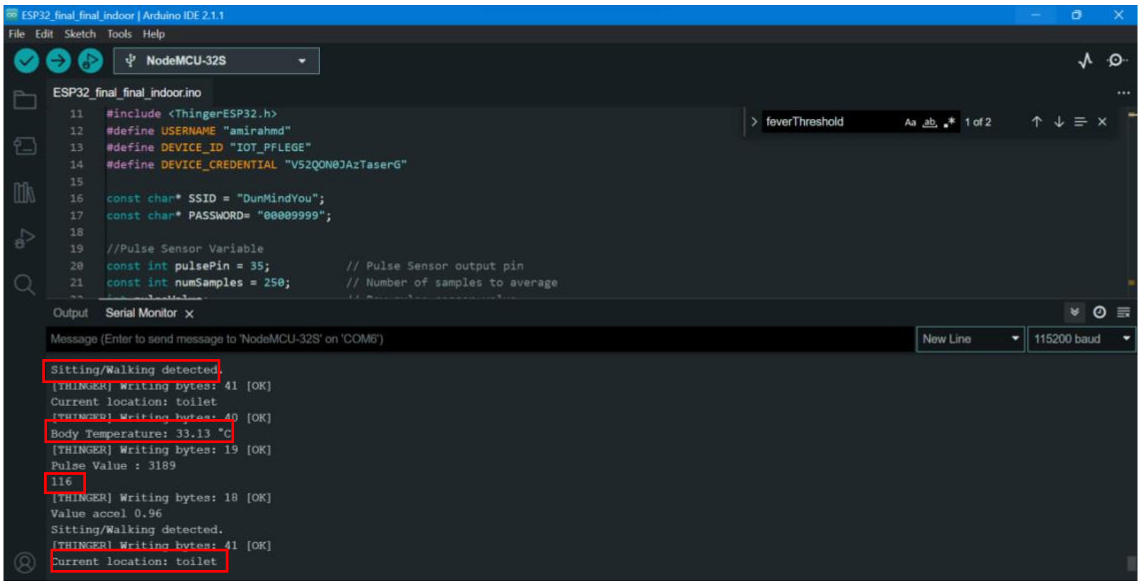
Device	IOT_PFLEGE
User	amirahmd
Description	Healthcare Monitor

5.4.1.1.5: Device Info on Thinger.io Mobile Application

5.4.1.1.6: V2 Foyer of V2 Café (Device is Located Indoor)

The above figures shows that the dashboard has been successfully built in both Thinger.io web-based and mobile application showing the indoor positioning, positioning, temperature, and heart rate. Figure 5.4.1.1.6 was taken when the prototype was located in an indoor building, on the first level V2 foyer which proved that it was the same place as shown in Google Maps in Figure 5.4.1.1.2.

5.4.1.2 Output in Arduino IDE Serial Monitor


ESP32_final_final_indoor | Arduino IDE 2.1.1
File Edit Sketch Tools Help
NodeMCU-32S
ESP32_final_final_indoor.ino
feverThreshold
1 of 2
Output
Serial Monitor
Message (Enter to send message to 'NodeMCU-32S' on 'COM6')
New Line
115200 baud

```
#include <ThingerESP32.h>
#define USERNAME "amirahmd"
#define DEVICE_ID "IOT_PFLEGE"
#define DEVICE_CREDENTIAL "V52QON0JAzTaserG"

const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";

//Pulse Sensor Variable
const int pulsePin = 35;          // Pulse Sensor output pin
const int numSamples = 250;       // Number of samples to average
```

```
Sitting/Walking detected.
[THINGER] Writing bytes: 41 [OK]
Current location: toilet
[THINGER] Writing bytes: 40 [OK]
Body Temperature: 33.13 °C
[THINGER] Writing bytes: 19 [OK]
Pulse Value : 3189
116
[THINGER] Writing bytes: 18 [OK]
Value accel 0.96
Sitting/Walking detected.
[THINGER] Writing bytes: 41 [OK]
Current location: toilet
```

Figure 5.4.1.2.1: Highlighted Outputs shown in the Serial Monitor

The higlighted outputs shown in the Serial Monitor of each sensor that has been integrated into the prototype. As seen, each sensor output, though accurate, is not being printed uniformly because of the indoor positioning algorithm. It is overlapped with “[THINGER] Writing bytes:…” too as data is being transmitted to Thinger.io while it is being written.

5.4.1.3 Testing

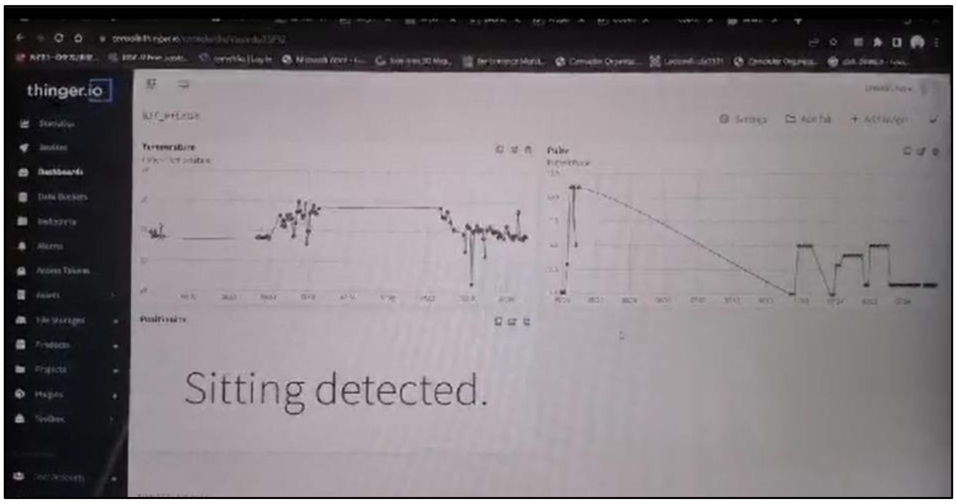


Figure 5.4.1.3.1: First Second Web-Based Dashboard Testing

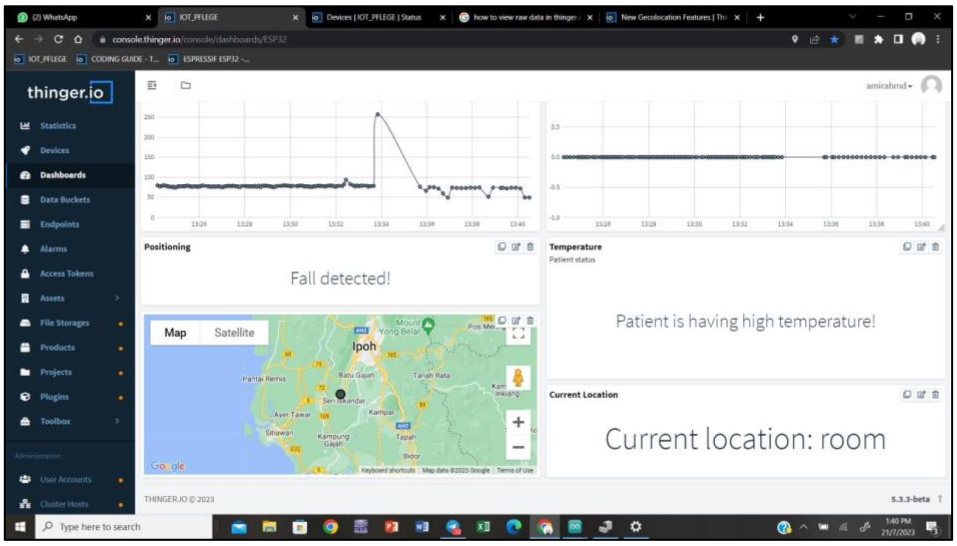


Figure 5.4.1.3.2: Second Web-Based Dashboard Testing

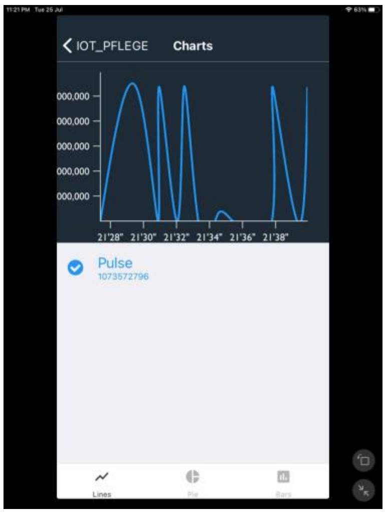


Figure 5.4.1.3.3: Mobile App Dashboard Testing

Thinger.io provides flexible and customizable features, allowing caretakers to configure notification settings according to their preferences. They can adjust the threshold values for each health parameter, defining what constitutes an alert or normal reading from the algorithm set in Arduino.ide. This customization ensures that the notification system is tailored to the specific health needs of each patient and minimizes false alarms.

In testing of building a Thinger.io dashboard, users just have to drag and drop the required widgets. There is currently no limit to how many widgets can be deployed on this free platform. After data transmission was successful, the first attempt in Figure 5.4.1.3.1 displayed only three widgets but they were not customised yet. There were two graph and one text message.

Then, as seen in Figure 5.4.1.3.2, more widgets had been added for body temperature, heart rate, three widgets in text format showing patient's current state (Positioning) where fall is detected, patient is having high temperature (Temperature) and the current location is room (Current Location). The text size and colour can be modified to easily differentiate between the text messages.

5.4.2 Discussion

By integrating Data Computer and Network using Thinger.io, the health monitoring system's dashboard, had been successfully built. The implementation of text message alerts with visual indicators simplifies and streamlines the monitoring process. Caretakers can efficiently track the health status of multiple patients at a glance, promptly identifying and addressing any health concerns. This real-time and responsive approach to health monitoring further supports the overall objective of providing high-quality and proactive care to the elderly residents in the Old Folk's home. Therefore, it is proven that humans do really need dashboards even with the advancements in AI and machine learning. Kelly (2020) stated that we are a visual society, which will never change, however without the ability to interact with data, manipulate and get answers, it will lead to frustration and is pointless.

Once the dashboard was created, only the admin can publish this dashboard and share it with the public. So, either the caretaker or the system administrator publishes the dashboard for everyone, permitting view for a particular user in a particular group. However, in this project, only an account is used to create the dashboard thus there is no shared link.

The temperature graph widget, named as “Patient Body Temperature”, Pulse allows the caretakers to monitor the current body temperature detected by MLX90614 Non-Contact Infrared temperature sensor. Once hovered, the actual value of the reading will be displayed. Whereas “Pulse” value comes from XD-58C Pulse Sensor.

Similarly, the other widgets’ values; “Current Location”, “Positioning”, “Temperature Status” are written based on the sensors listed in Table 3. Communication is delivered through the application as it does not only host the Thinger.io web application but also a Thinger.io mobile application which allows caretakers to view the same data visualisation in a slightly different representation. For example, to view a graph in the mobile app, users have to click "Charts", whereas clicking "Update" will refresh the last data received (if there is any delay). Data received will always be updated unless Pflege is disconnected from its power supply. Pflege data visualisation depends on Wi-Fi connection and data transmission. Once Pflege wants to connect, or after it has been disconnected it would have around seven seconds delay before the service status becomes "Offline" to "Online" (or vice versa) in the dashboard.

Figure 5.4.1.1.2 shows Thinger.io feature to show when the last data was received. Thinger.io provides various features for IoT data visualization and monitoring, but it does not have a built-in feature to show the exact timestamp of when the last data was received from a device or sensor. However, Thinger.io does offer real-time data visualisation through its dashboards and widgets, allowing users to monitor and visualize heart rate, temperature and falling status data coming from ESP32 and the sensors in real-time. The data is typically updated as it is received by the Thinger.io platform.

For devices equipped with GPS (Global Positioning System) sensors, the GPS data collection into Thinger.io device (ESP32) can be integrated. Thinger.io provides APIs to send data from ESP32 through Wi-Fi to the platform, allowing location-related information to be captured. This data can include latitude, longitude, and other relevant location attributes such as Live Street (Satellite Live View). In this project, external mapping services like Google Maps, was used. Google Maps mapping services provide APIs that allow display of geospatial data on interactive maps.

5.5 Limitation

1. Huge delay due to the complexity of indoor positioning algorithm

   The major limitation of our project is the delay exist in the program due to the complexity of the code which results in data inaccuracy. There are two parts in the program where one is for health monitoring sensing and the another is for indoor positioning. However, the algorithm for indoor positioning is too complex until the program needs to take more than 5 second just to detect the position of the patient. This causes the issue where the health monitoring sensing is not functioning when the program in the ESP32 microcontroller is figuring the actual position of the patient since the ESP32 microcontroller does not have the function of parallel processing to execute indoor positioning and health monitoring sensing at the same time.

   The worst case can happen due to this limitation is that the patient is experiencing fatal fall, but the system is not able to detect the fall when it is taking long time to locate the patient's indoor position. The caretaker will not be notified regarding the fall and thus timely first aid is unable to be carried out.

2. Sensor Accuracy

   There are several sensors that are lack of accuracy for critical health monitoring system such as the pulse sensor and accelerometer. One of the reasons is because the sensors we bought for our product is relatively cheaper due to the limited budget and therefore the sensors are having lower quality, simpler design architecture, limited signal processing capabilities, environmental sensitivity, and others. Another reason might because the simple algorithm is used for heart rate detection and fall detection which leads to insufficient data processing as the simple algorithms may not process the raw data from the sensors adequately. Health monitoring systems require sophisticated signal processing techniques to extract meaningful information from noisy sensor data. Thus, the measurements can be inaccurate without proper filtering and analysis. The simple algorithms may cause the lack of calibration process which is necessary to ensure accurate measurements as well.

6 CONCLUSION

6.1 Summary of Progress and Findings

The project aimed to create a health monitoring system for elderly residents in an Old Folk's home, addressing challenges in manually monitoring the health of the patients. The system was designed to provide a convenient and efficient solution to ease the workload of caretakers and enhance the overall healthcare management process.

To achieve this, the project activities focused on implementing various features in the health monitoring wristband. Continuous and accurate heart rate monitoring was incorporated, ensuring reliable heart rate measurements for identifying anomalies or irregularities. Additionally, the system consistently monitored body temperature to detect fevers or health changes early on. One critical aspect of the system was the implementation of a fall detection mechanism using accelerometer and gyroscope sensors. This feature reliably detected falls and promptly alerted caretakers in emergencies, ensuring timely medical attention for patients.

Furthermore, the project incorporated an indoor positioning system to determine the precise location of patients within the Old Folk's home. This capability improved emergency response times and the overall safety of the elderly residents. To ease the caretakers' task of monitoring patient health conditions, a user-friendly interface was developed. The interface provided caretakers with up-to-date health information and alerts when necessary, making it easy to access and monitor patient data efficiently.

Moreover, the project implemented cloud data transmission and communication, allowing authorized personnel to have centralized access to patient health data. The data stored in the cloud facilitated easy analysis and long-term tracking of patients' health conditions. The overall outcome of the project was a comprehensive healthcare monitoring system that consistently checked heart rate, body temperature, and fall events, while providing indoor location tracking and cloud-based record-keeping. The final product proved to be a practical and portable solution, ensuring the ease and convenience of elderly residents in the Old Folk's home.

By successfully addressing these objectives, the health monitoring system significantly improved the care provided to the old folks, enhancing early detection of health issues, and streamlining the caregiving process for caretakers. This innovative solution empowered caretakers with a powerful tool to ensure the well-being and safety of their, fostering a conducive and caring environment in the Old Folk's home.

6.2 Recommendations

1. Use two microcontrollers instead of one for the system.

   Since the major limitation is the huge delay from the complexity of indoor positioning algorithm which causes the malfunction of health monitoring sensing, we can split these two functions into two microcontrollers where each will be responsible on one of the functions. With this, the indoor positioning algorithm will no longer be affecting the health monitoring sensing and the health conditions of the patient such as the heart rate, body temperature and fall can be detected in real time without any delay.

2. Calibrate all the sensors in detail.

   The sensors will only function at their best accuracy with proper and detailed calibration. To ensure a sensor is well calibrated, firstly, we must understand the sensor specifications and the intended operating conditions. Next, we can use the calibration standards with known reference values to create a controlled environment for measurements and the sensor’s data is collected by taking multiple readings across the sensor's range. Then, we must analyse the data statistically by calculating calibration factors to align readings with true values. The step of applying corrections to adjust raw sensor outputs will help to generate more accurate and reliable data. Make sure to account for the environmental factors such as temperature and humidity that might affect the sensors from getting accurate data during the calibration. To proof-check the calibration, we can even validate the readings by retesting with the same standards of sensors.

3. Database Hosting

Thinger.io does not provide native database hosting for popular databases such as MySQL. It is primarily intended for the storing and visualisation of time-series data. A database is essential in a healthcare monitoring system for organising, storing, and retrieving patient-related data and health information. It provides a number of important advantages that add to the overall efficacy and efficiency of the healthcare monitoring process. For example, instant access to information enables speedier decision-making, particularly in urgent situations, and enables prompt answers to patient demands. Furthermore, healthcare databases may keep patient data for an extended period of time, ensuring that previous health information are easily accessible for future reference, study, or audits.

4. Add notification feature.

To enhance the health monitoring system's capabilities, it is recommended to implement a notification feature. The notifications can be sent via various communication channels, such as Telegram, SMS alerts, or email notifications, depending on the preferences of the caretakers and the available infrastructure. Additionally, the notifications can be configured with different levels of urgency, enabling caretakers to prioritize their response based on the severity of the health event. The notification feature will enable the system to send timely alerts to caretakers and healthcare personnel in case of critical health events or emergencies. When the system detects abnormal heart rate or body temperature readings, a fall event, or any other concerning health condition, it will trigger notifications to the designated caregivers' devices. The notification feature ensures that caretakers and healthcare staff can promptly respond to the elderly residents' health needs, providing timely medical attention or assistance when required. This feature adds an extra layer of safety and responsiveness to the health monitoring system, further improving the quality of care provided to the elderly residents.

5. Colour-coded widgets notification display on dashboard

Implement at notification alerts using thinger.io. The dashboard will continuously monitor the patient's heart rate, body temperature, fall events, and other health parameters. When any of these parameters indicate abnormal readings or a fall event is detected, the system will trigger a notification. Using thinger.io, the notification will be visually represented on the dashboard by changing the background colour of the corresponding widgets. When the patient's status is normal, the widget background will remain green, signifying that the health parameters are within acceptable ranges. However, in the event of an alert, such as an abnormal heart rate or body temperature reading or the detection of a fall, the widget background will change to red, indicating a critical health event. This visual representation of alerts on the dashboard will allow caretakers and healthcare personnel to quickly identify and respond to abnormal health conditions. The immediate change in colour draws attention to the affected patient's status, enabling swift action to address the situation.

REFERENCES

Abdulmalek, S., Nasir, A., Jabbar, W. A., Almuhaya, M. a. M., Bairagi, A. K., Khan, M. A., & Kee, S. (2022). IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review. *Healthcare*, *10*(10), 1993. https://doi.org/10.3390/healthcare10101993

Arduino and ESP32 WiFi indoor positioning. (n.d.). Eloquent Arduino. https://eloquentarduino.com/arduino-indoor-positioning/

Dosm. (n.d.). *Department of Statistics Malaysia*. https://www.dosm.gov.my/portal-main/home

Facts about falls | Fall Prevention | Injury Center | CDC. (n.d.). https://www.cdc.gov/falls/facts.html

HUAWEI. (n.d.). *HUAWEI Health - HUAWEI Global*. https://consumer.huawei.com/en/mobileservices/health/

Kelly, J. (2020, July 27). *Digital Business Analytics: Accelerating your dashboard journey*. Dynatrace News. https://www.dynatrace.com/news/blog/digital-business-analytics-accelerating-your-dashboard-journey

Khan, M. M., Alanazi, T. M., Albraikan, A. A., & Almalki, F. A. (2022). IoT-Based health monitoring system development and analysis. *Security and Communication Networks*, *2022*, 1–11. https://doi.org/10.1155/2022/9639195

MyGOV - The Government of Malaysia's Official Portal. (n.d.). https://www.malaysia.gov.my/portal/content/30740

National Academies Press (US). (2008). *Health status and health care service utilization*. Retooling for an Aging America - NCBI Bookshelf. https://www.ncbi.nlm.nih.gov/books/NBK215400/

Peterson, D. (2016, December 15). Internet of People (IoP): The next frontier for IoT. *IoT Agenda*. https://www.techtarget.com/iotagenda/blog/IoT-Agenda/Internet-of-People-IoP-The-next-frontier-for-IoT

Rpn, A. J. M. (2023). Top wearable medical devices used in healthcare. *HealthNews*. https://healthnews.com/family-health/healthy-living/wearable-medical-devices-used-in-healthcare/

Take an ECG with the ECG app on Apple Watch. (2022, September 12). Apple Support. https://support.apple.com/en-my/HT208955

The National Council on Aging. (n.d.). https://ncoa.org/article/the-inequities-in-the-cost-of-chronic-disease-why-it-matters-for-older-adults

The National Council on Aging. (n.d.-b). https://ncoa.org/article/get-the-facts-on-healthy-aging

APPENDIX

FMEA

DESIGN FAILURE MODE AND EFFECTS ANALYSIS (FMEA)

Product Name	Health Monitoring System	Team 3
Responsibility	Sensing and Positioning	

Prepared by: Tan Yu Ting
FMEA Date: 20/5/2023

Item	**Function**	**Potential Failure Mode**	**Potential Failure Effects**	**SEV**	**Potential Causes**	**OCC**	**Current Controls**	**DET**	**RPN**	**Action Recommended**	**Resp.**	**Action Taken**
What is the subsystem / component under study?	*What is the primary purpose or design intend of the subsystem / component?*	*In what ways that subsystem / component can potentially fail to meet the intended functions?*	*What is the consequences of the failure to the system or end user*	*How severe its effect to the system / end user?*	*What is the design deficiency that results in the failure modes?*	*How often does cause of FM occur?*	*What are the existing controls and procedures (preventive and detection) that prevent the cause or the Failure Mode?*	*How well can you detect the FM causes?*		*What are the actions for reducing the occurrence of the cause, or improving detection? Should have actions only on high RPNs or easy fixes.*	*Who's responsible for the recommended action?*	*What are the completed actions taken with the recalculated RPN? Be sure to include completion month/year.*
Power Supply to	NodeMCU ESP32S is an integrated micro-controller that is	Excessive power is supplied to	Applying excessive voltage or current to an ESP32 board can	7	Did not study the data sheet for	4	Preventive: Study the data sheet for	2	56			

NodeMCU ESP32S	responsible for executing the code and controlling the connected components such as the sensors.	the Arduino board.	cause damage to the components and potentially render the board unusable.		NodeMCU ESP32S.		NodeMCU ESP32S then notice the voltage and current supplied by the battery chosen and do accurate calculation on the power supply.					
					Wrong selection of battery to supply power for NodeMCU ESP32S.	4	Detection: Always be aware of the condition of the board whether it is overheated.	3	84	Make sure to start trying with battery that supplies lower voltage. Lack of power will only cause the Arduino to be unable to function correctly but will not burn the board.	The member who in charge of the sensing and positioning subsystem. (Tan Yu Ting)	Bought base board for NodeMCU ESP32S and 9V voltage is supplied through the base board DC Jack as it regulates the input voltage to suitable operating voltage.

Power Supply from ESP32 to the components such as sensors.	There are two voltage levels available in NodeMCU ESP32S 38Pin which are 3.3V and 5V and should be wisely chosen as voltage supplied to the components connected to it. The components will help in the health monitoring function.	The wrong voltage level is selected.	If voltage level selected is too high, it will cause damage to components due to overheating whereas there might be incorrect operation from the components if low voltage level is supplied.	7	Did not study the data sheet for all the components used, causing the wrong selection of voltage levels in ESP32 board.	5	Preventive: Study the data sheet for all components to ensure they operate on voltage level of 3.3V or 5V.	3	105	Make sure to study the data sheet of all components used so that the correct operating voltage level can be chosen.	The member who in charge of the sensing and positioning subsystem. (Tan Yu Ting)	Read all the components' datasheet and save the datasheets in technical folder to ease future referring.
							Detection: Aware on all the components connected to the ESP32 board to check for any overheating issues or check whether the sensors are functioning correctly.	2	70			

DESIGN FAILURE MODE AND EFFECTS ANALYSIS (FMEA)

Product Name	Health Monitoring System (Old Folks' Home)	Team 3
Responsibility	DCN & Visualization	

Prepared by: Hana Aziz
FMEA Date: 20/6/2023

Item	**Function**	**Potential Failure Mode**	**Potential Failure Effects**	**SEV**	**Potential Causes**	**OCC**	**Current Controls**	**DET**	**RPN**	**Action Recommended**	**Resp.**	**Action Taken**
What is the subsystem/component under study?	*What is the primary purpose or design intend of the subsystem/component?*	*In what ways that subsystem/component can potentially fail to meet the intended functions?*	*What is the consequences of the failure to the system or end-user*	*How severe its effect to the*	*What is the design deficiency that results in the failure modes?*	*How often does cause of FM*	*What are the existing controls and procedures (preventive and detection) that prevent the cause or the Failure Mode?*	*How well can you detect the*		*What are the actions for reducing the occurrence of the cause, or improving detection? Should have actions only on high RPNs or easy fixes.*	*Who's responsible for the recommended action?*	*What are the completed actions taken with the recalculated RPN? Be sure to include completion month/year.*
Thingspeak	Provide a platform for collecting, storing, and analyzing real-time Internet of Things (IoT) data.	Poor internet connectivity or server downtime.	May result in loss of data, inaccurate data analysis, and delayed decision-making.	5	This may include inadequate system redundancy, insufficient data validation, and poor error handling.	3	Regular system maintenance, network monitoring, data validation checks, and error reporting mechanisms.	7	105	Improving the design or manufacturing process, implementing additional quality control measures, providing additional training to personnel, or enhancing monitoring and reporting mechanisms.	The member who is in charge of the visualisation. (Hana)	Move to Favoriot.
Arduino IoT Cloud & Favoriot	Provide a platform for developers to create custom mobile applications for IoT projects,	Issues with connectivity, compatibility, or security prevent devices from	If devices are unable to communicate with the	6	Issues related to connectivity, compatibility, and	3	Regular testing and maintenance to identify and address potential issues before they become	8	144	Have a system in place for monitoring and analyzing	The member who is in charge of the	Move to Arduino IoT Cloud & Thinger.io

	including data visualization. Arduino IoT Cloud allows developers to create custom widgets and graphs to display data collected from connected devices, making it easier to monitor and analyze data in real-time	communicating with the platform or cause data to be compromised. There are limitations in the platform's capabilities - lack of support for specific protocols or devices, or if the platform experiences performance issues or downtime .	platform, or if data is compromise d, this could result in inaccurate or incomplete data, which could lead to incorrect decisions or actions being taken. If the platform experiences performance issues or downtime, this could result in delays or disruptions to critical processes or services.		security – not designed to support certain protocols or devices, could result in devices being unable to communicat e with the platform, which could lead to data being lost or inaccurate. If the platform experiences performance issues or downtime, this could result from inadequate infrastructur e or design, which could cause delays or disruptions to critical processes or services.		significant problems. Real-time alerts and notifications to notify users of potential issues as they occur.			data. This could include real-time alerts and notifications, regular testing and maintenance, and a system for tracking and analyzing data to identify potential issues before they become significant problems.	visualisatio n. (Hana	
Thinger.io	IoT platform to connect devices to the internet and manage them remot ely with a feature that allows to create custom dashboards for IoT devices, display text and	Thinger.io may fail to meet its intended functions to monitor in real time if there are issues with the internet connection, device compatibility, or if there are bugs or	If Thinger.io fails, it could result in the loss of data, decreased system efficiency, and in some	7	Issues with compatibilit y, scalability, security, and reliabilit y.	3	Regular software updates and bug fixes, compatibility testing with different devices and systems, data backup and recovery processes, security protocols such as encryption	5	10 5	Conducting regular risk assessments to identify potential failure modes, implementing additional security	The member who is in charge of the visualisatio n. (Hana	Read all the related documentati on and open tickets for references.

	monitor and analyse data in real-time.	glitches in the software.	cases, a complete system shutdown. This could lead to a loss of productivity and revenue for the end user.				and user authentication, and monitoring tools to detect and alert users to potential issues.			measures to prevent data loss or theft, improving compatibility with different devices and systems, and providing users with clear instructions and resources for troubleshooting issues.		

DESIGN FAILURE MODE AND
EFFECTS ANALYSIS (FMEA)

Product Name	Health Monitoring System (Old Folks' Home)	Team 3
Responsibility	DCN & Data Transmission	

Prepared by: Amira
FMEA Date: 20/6/2023

Item	Function	Potential Failure Mode	Potential Failure Effects	S E V	Potential Causes	O C C	Current Controls	D E T	R P N	Action Recommended	Resp.	Action Taken
What is the subsystem / component under study?	*What is the primary purpose or design intend of the subsystem / component?*	*In what ways that subsystem / component can potentially fail to meet the intended functions?*	*What is the consequences of the failure to the system or end user*	*How severe its effect to the system / end user?*	*What is the design deficiency that results in the failure modes?*	*How often does cause of FM occur?*	*What are the existing controls and procedures (preventive and detection) that prevent the cause or the Failure Mode?*	*How well can you detect the FM causes?*		*What are the actions for reducing the occurrence of the cause, or improving detection? Should have actions only on high RPNs or easy fixes.*	*Who's responsible for the recommended action?*	*What are the completed actions taken with the recalculated RPN? Be sure to include completion month/year.*
ESP8266 ESP01 WiFi Module	Allow devices to connect to WiFi networks. Enable the device to communicate with other devices or access the internet.	Overheating while connected.	Overheating will cause permanent damage to the module, unexpected resets, loss of data and system failure.	9	Excessive voltage consumption. Wrong selection of voltage when connected to the Arduino Mega	2	Ensure the module is connected at the correct power supply by referring to the data sheet.	3	54	Ensure the module is connected at the correct voltage supply.	The member who is in charge of the visualisation. (Hana)	Move to Favoriot.
		Defected module.	Unable to connect to selected WiFi mobile hotspot	9	Old devices which may have been used for many times. Device is not configured before used.	5	Ensure to use a new ESP8266 ESP01 WiFi Module.	4	180	Ensure that the used devices is configured before used.	The member who is in charge of the visualisation. (Hana	Move to Arduino IoT Cloud & Thinger.io

NodeMCU ESP32S	Allow devices to connect to WiFi networks. Enable the device to communicate with other devices or access the internet.	Defected devices	Unable to connect to selected WiFi mobile hotspot	9	Error in code where Wifi.h() is not declare Error in code where Wifi.status() is not declared to check the connection status.	2	Ensure that the Wifi.h and Wifi.status function is insert in the code to connect to WiFi mobile hotspot.	3	54	Ensure that the Wifi setup code is insert in the code.	The member who is in charge of the visualisation. (Hana	Read all the related documentation and open tickets for references.

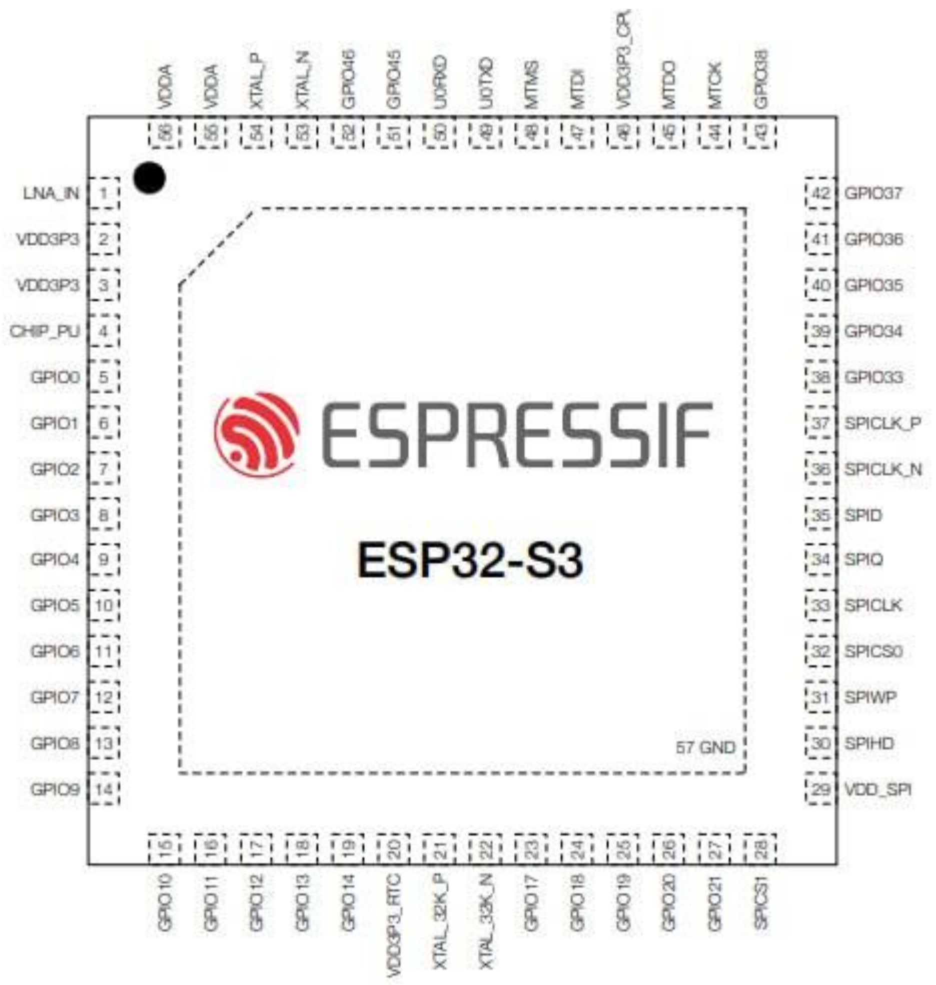
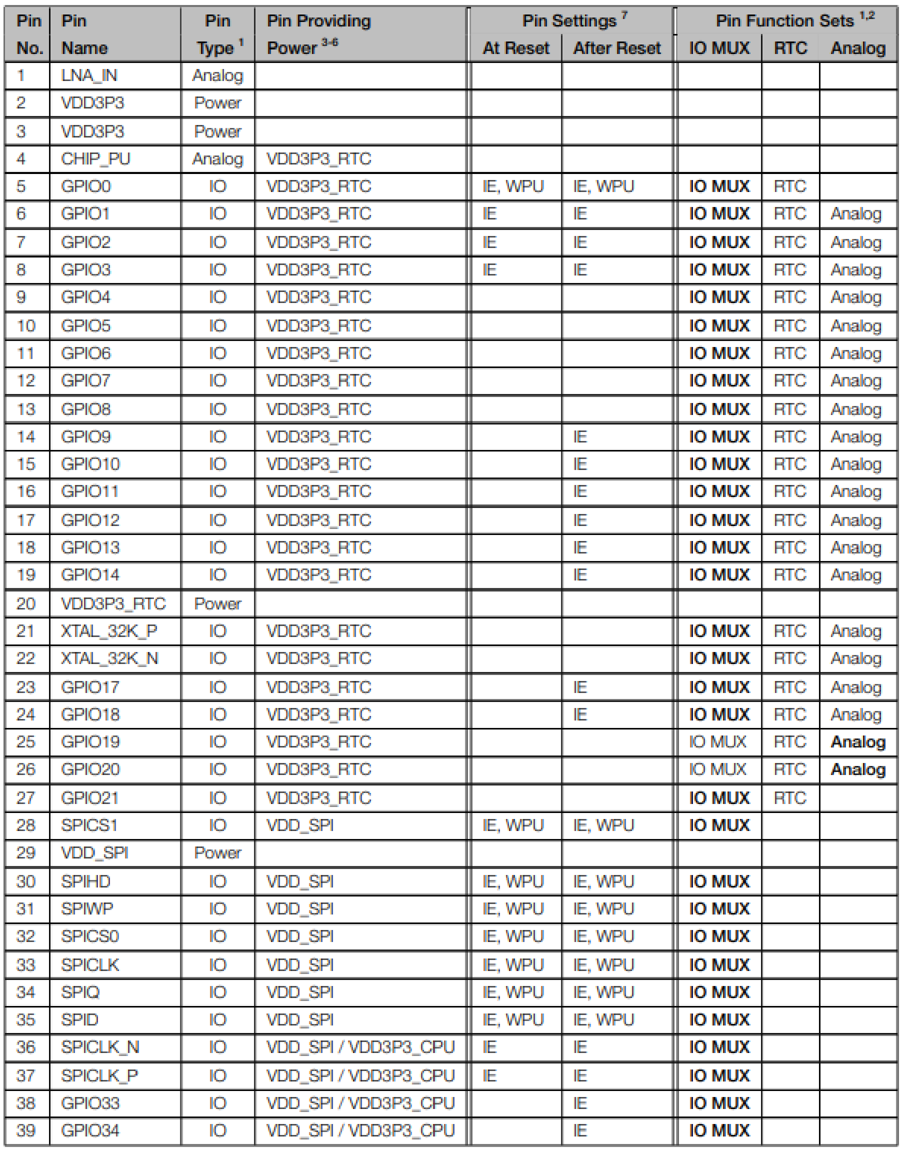
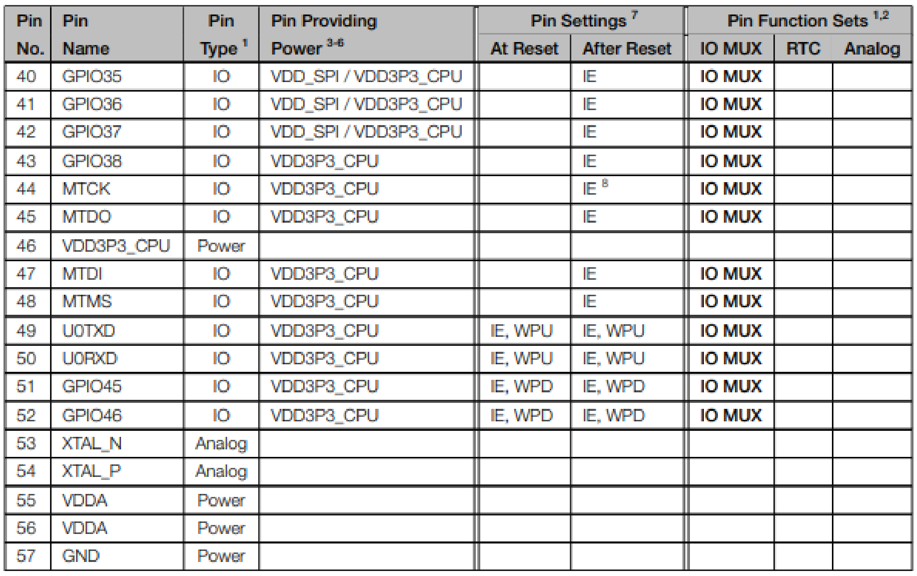


Figure 1 ESP32-S3 Pin Layout (Top View)

Table 1 Pin Overview

Pin	Pin	Pin	Pin Providing	Pin Settings [7]		Pin Function Sets [1,2]		
No.	Name	Type [1]	Power [3-6]	At Reset	After Reset	IO MUX	RTC	Analog
1	LNA_IN	Analog						
2	VDD3P3	Power						
3	VDD3P3	Power						
4	CHIP_PU	Analog	VDD3P3_RTC					
5	GPIO0	IO	VDD3P3_RTC	IE, WPU	IE, WPU	**IO MUX**	RTC	
6	GPIO1	IO	VDD3P3_RTC	IE	IE	**IO MUX**	RTC	Analog
7	GPIO2	IO	VDD3P3_RTC	IE	IE	**IO MUX**	RTC	Analog
8	GPIO3	IO	VDD3P3_RTC	IE	IE	**IO MUX**	RTC	Analog
9	GPIO4	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
10	GPIO5	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
11	GPIO6	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
12	GPIO7	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
13	GPIO8	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
14	GPIO9	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
15	GPIO10	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
16	GPIO11	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
17	GPIO12	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
18	GPIO13	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
19	GPIO14	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
20	VDD3P3_RTC	Power						
21	XTAL_32K_P	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
22	XTAL_32K_N	IO	VDD3P3_RTC			**IO MUX**	RTC	Analog
23	GPIO17	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
24	GPIO18	IO	VDD3P3_RTC		IE	**IO MUX**	RTC	Analog
25	GPIO19	IO	VDD3P3_RTC			IO MUX	RTC	**Analog**
26	GPIO20	IO	VDD3P3_RTC			IO MUX	RTC	**Analog**
27	GPIO21	IO	VDD3P3_RTC			**IO MUX**	RTC	
28	SPICS1	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
29	VDD_SPI	Power						
30	SPIHD	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
31	SPIWP	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
32	SPICS0	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
33	SPICLK	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
34	SPIQ	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
35	SPID	IO	VDD_SPI	IE, WPU	IE, WPU	**IO MUX**		
36	SPICLK_N	IO	VDD_SPI / VDD3P3_CPU	IE	IE	**IO MUX**		
37	SPICLK_P	IO	VDD_SPI / VDD3P3_CPU	IE	IE	**IO MUX**		
38	GPIO33	IO	VDD_SPI / VDD3P3_CPU		IE	**IO MUX**		
39	GPIO34	IO	VDD_SPI / VDD3P3_CPU		IE	**IO MUX**		
40	GPIO35	IO	VDD_SPI / VDD3P3_CPU		IE	**IO MUX**		
41	GPIO36	IO	VDD_SPI / VDD3P3_CPU		IE	**IO MUX**		
42	GPIO37	IO	VDD_SPI / VDD3P3_CPU		IE	**IO MUX**		
43	GPIO38	IO	VDD3P3_CPU		IE	**IO MUX**		
44	MTCK	IO	VDD3P3_CPU		IE [8]	**IO MUX**		
45	MTDO	IO	VDD3P3_CPU		IE	**IO MUX**		
46	VDD3P3_CPU	Power						
47	MTDI	IO	VDD3P3_CPU		IE	**IO MUX**		
48	MTMS	IO	VDD3P3_CPU		IE	**IO MUX**		
49	U0TXD	IO	VDD3P3_CPU	IE, WPU	IE, WPU	**IO MUX**		
50	U0RXD	IO	VDD3P3_CPU	IE, WPU	IE, WPU	**IO MUX**		
51	GPIO45	IO	VDD3P3_CPU	IE, WPD	IE, WPD	**IO MUX**		
52	GPIO46	IO	VDD3P3_CPU	IE, WPD	IE, WPD	**IO MUX**		
53	XTAL_N	Analog						
54	XTAL_P	Analog						
55	VDDA	Power						
56	VDDA	Power						
57	GND	Power						

Final Code - https://youtu.be/WQ44kQsJJl8

ESP32_final_final_indoor.ino (Command Line)

```
/*19000505 19000541 19000437 SIDP DCN

*
* EDB4703: Systems Integration Design Project MAY 2023
* EDB3023/EEB4063: Data and Computer Network MAY 2023
* 26th July 2023
*
* GROUP 3
*
* PROJECT: Pflege Old Folks Health Monitor Wristband
* 1 Tan Yu Ting 19000505
* 2 Nur Amira binti Ahmad Rezal 19000541
* 3 Hana Najwa Binti Abdul Aziz 19000437
*
* Submitted to:
* Dr. Fawnizu Azmadi B Hussin
* Ir. Dr. Huzein Fahmi B Hawari
*
*/

#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include <MPU6050.h>
#include <stdio.h>
#include <string.h>
#include "WiFi.h"
#include "eloquent.h"
#include "eloquent/networking/wifi/WifiScanner.h"
#define THINGER_SERIAL_DEBUG

#include <ThingerESP32.h>
#define USERNAME "amirahmd"
#define DEVICE_ID "IOT_PFLEGE"
#define DEVICE_CREDENTIAL "V52QON0JAzTaserG"

const char* SSID = "DunMindYou";
const char* PASSWORD= "00009999";

//Pulse Sensor Variable
const int pulsePin = 35;           // Pulse Sensor output pin
const int numSamples = 250;        // Number of samples to average
int pulseValue;                    // Raw pulse sensor value
```

```
unsigned long previousMillis = 0;  // Previous time in milliseconds
unsigned long interval = 1000;     // Update interval for BPM calculation

ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

//MLX90614 sensor Variable
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
const float feverThreshold = 36.5;  // Celsius

//MPU6050 Accelerometer Variable
MPU6050 mpu;
const float alpha = 1.0;              // Complementary filter parameter
(adjust as needed)
const float thresholdFall = 0.9;      // Threshold for fall detection
(adjust as needed)
const float thresholdSitting = 1.5;  // Threshold for sitting detection
(adjust as needed)
//const float thresholdWalking = 2.0;  // Threshold for walking detection
(adjust as needed)
const float thresholdRunning = 2.0;  // Threshold for running detection
(adjust as needed)
float lastAccelMagnitude = 0.0;      // Previous magnitude of acceleration
vector

//Buzzer Variable
#define BUZZER_PIN 13 // Change this to the GPIO pin connected to the
buzzer

class WiFiIndoorPositioning {
public:
  /**
         * Get current room id
         */
  template<typename Scanner>
  int predictRoomId(Scanner& scanner) {
    float x[15] = { 0 };
    uint16_t votes[3] = { 0 };

    scanner.scan();
    convertToFeatures(scanner, x);

    if (x[9] <= -80.5) {
      if (x[14] <= -89.5) {
        votes[2] += 1;
      } else {
        if (x[13] <= -73.0) {
          if (x[2] <= -41.0) {
            if (x[7] <= -60.5) {
              if (x[13] <= -75.5) {
                votes[1] += 1;
              } else {
                votes[2] += 1;
              }
            } else {
              votes[0] += 1;
            }
          } else {
            votes[2] += 1;
          }
        } else {
          votes[2] += 1;
        }
      }
    } else {
      if (x[13] <= -71.0) {
        if (x[9] <= -75.5) {
          votes[1] += 1;
        } else {
          if (x[12] <= -76.5) {
            votes[0] += 1;
          } else {
            votes[1] += 1;
          }
        }
      } else {
        votes[2] += 1;
      }
    }
    if (x[6] <= -42.5) {
      votes[1] += 1;
    } else {
      if (x[9] <= -80.5) {
        if (x[2] <= -41.0) {
          if (x[12] <= -72.5) {
            if (x[9] <= -84.0) {
              if (x[2] <= -90.0) {
                votes[1] += 1;
              } else {
                if (x[14] <= -44.0) {
                  votes[0] += 1;
                } else {
                  if (x[2] <= -88.5) {
                    votes[0] += 1;
                  } else {
                    votes[1] += 1;
                  }
                }
              }
            } else {
              votes[0] += 1;
            }
          } else {
            votes[2] += 1;
          }
        } else {
          votes[2] += 1;
        }
      } else {
        if (x[14] <= -42.0) {
          if (x[10] <= -69.5) {
            votes[2] += 1;
          } else {
            votes[1] += 1;
          }
        } else {
          if (x[2] <= -90.0) {
            votes[1] += 1;
          } else {
            votes[0] += 1;
          }
        }
      }
    }
    if (x[13] <= -73.5) {
      if (x[2] <= -41.0) {
        if (x[0] <= -45.0) {
          votes[1] += 1;
        } else {
          if (x[5] <= -43.5) {
            votes[1] += 1;
          } else {
            if (x[13] <= -78.5) {
              if (x[1] <= -46.5) {
                votes[1] += 1;
              } else {
                if (x[13] <= -86.5) {
                  votes[0] += 1;
                } else {
                  if (x[9] <= -82.5) {
                    if (x[9] <= -85.5) {
                      votes[1] += 1;
                    } else {
                      if (x[10] <= -69.0) {
                        votes[0] += 1;
                      } else {
                        votes[1] += 1;
                      }
                    }
                  } else {
                    votes[1] += 1;
                  }
                }
              }
            } else {
              if (x[12] <= -71.0) {
                votes[0] += 1;
              } else {
                votes[2] += 1;
              }
            }
          }
        }
      } else {
        votes[2] += 1;
      }
    } else {
      votes[2] += 1;
    }
    if (x[9] <= -79.5) {
      if (x[12] <= -73.5) {
        if (x[9] <= -84.5) {
          if (x[13] <= -84.5) {
            if (x[2] <= -88.0) {
              votes[1] += 1;
            } else {
              votes[0] += 1;
            }
          } else {
            votes[1] += 1;
          }
        } else {
          if (x[2] <= -86.5) {
            votes[1] += 1;
          } else {
            if (x[9] <= -80.5) {
              votes[0] += 1;
            } else {
              votes[1] += 1;
            }
          }
        }
      } else {
        if (x[10] <= -75.5) {
          votes[2] += 1;
        } else {
          if (x[12] <= -70.0) {
            votes[0] += 1;
          } else {
            votes[2] += 1;
          }
        }
      }
    } else {
      if (x[7] <= -55.0) {
        votes[1] += 1;
      } else {
        votes[0] += 1;
      }
    }
    if (x[13] <= -73.5) {
      if (x[12] <= -72.5) {
        if (x[7] <= -55.0) {
          if (x[13] <= -77.5) {
            votes[1] += 1;
          } else {
            if (x[7] <= -60.5) {
              votes[1] += 1;
            } else {
              votes[0] += 1;
            }
          }
        } else {
          votes[0] += 1;
        }
      } else {
        if (x[3] <= -43.5) {
          votes[2] += 1;
        } else {
          if (x[14] <= -85.0) {
            votes[2] += 1;
          } else {
            if (x[12] <= -68.5) {
              votes[0] += 1;
            } else {
              votes[2] += 1;
            }
          }
        }
      }
    } else {
      if (x[12] <= -76.5) {
        votes[1] += 1;
      } else {
        votes[2] += 1;
      }
    }
    if (x[7] <= -53.5) {
      if (x[13] <= -74.0) {
        if (x[13] <= -82.5) {
          votes[1] += 1;
        } else {
          if (x[14] <= -86.5) {
            if (x[12] <= -73.5) {
              votes[0] += 1;
            } else {
              votes[2] += 1;
            }
          } else {
            if (x[10] <= -67.5) {
              if (x[2] <= -89.0) {
                votes[2] += 1;
              } else {
                votes[0] += 1;
              }
            } else {
              votes[1] += 1;
            }
          }
        }
      } else {
        votes[2] += 1;
      }
    } else {
      if (x[12] <= -73.5) {
        votes[0] += 1;
      } else {
        if (x[10] <= -76.5) {
          votes[2] += 1;
        } else {
          if (x[14] <= -41.5) {
            votes[0] += 1;
          } else {
            votes[2] += 1;
          }
        }
      }
    }
    if (x[10] <= -67.0) {
      if (x[2] <= -41.5) {
        if (x[12] <= -74.5) {
          votes[0] += 1;
        } else {
          if (x[13] <= -78.5) {
            votes[0] += 1;
          } else {
            votes[2] += 1;
          }
        }
      } else {
        if (x[7] <= -53.5) {
          votes[2] += 1;
        } else {
          if (x[12] <= -69.5) {
            if (x[14] <= -86.0) {
              votes[2] += 1;
            } else {
              votes[0] += 1;
            }
          } else {
            votes[2] += 1;
          }
        }
      }
    } else {
      if (x[12] <= -69.0) {
        votes[1] += 1;
      } else {
        votes[2] += 1;
      }
    }
    if (x[14] <= -90.5) {
      if (x[7] <= -59.0) {
        votes[1] += 1;
      } else {
        votes[2] += 1;
      }
    } else {
      if (x[13] <= -73.5) {
        if (x[2] <= -87.0) {
          if (x[7] <= -55.5) {
            if (x[6] <= -44.5) {
              votes[1] += 1;
            } else {
              if (x[14] <= -89.5) {
                votes[0] += 1;
              } else {
                if (x[7] <= -65.5) {
                  if (x[10] <= -76.0) {
                    votes[2] += 1;
                  } else {
                    votes[1] += 1;
                  }
                } else {
                  votes[1] += 1;
                }
              }
            }
          } else {
            votes[0] += 1;
          }
        } else {
          if (x[6] <= -42.5) {
            votes[1] += 1;
          } else {
            if (x[5] <= -43.5) {
              votes[1] += 1;
            } else {
              if (x[9] <= -85.5) {
                if (x[12] <= -75.5) {
                  votes[1] += 1;
                } else {
                  votes[2] += 1;
                }
              } else {
                votes[0] += 1;
              }
            }
          }
        }
      } else {
        if (x[9] <= -79.0) {
          votes[2] += 1;
        } else {
          votes[1] += 1;
        }
      }
    }
    if (x[10] <= -71.5) {
      if (x[2] <= -41.0) {
        if (x[12] <= -74.5) {
          votes[0] += 1;
        } else {
          votes[2] += 1;
        }
      } else {
        votes[2] += 1;
      }
    } else {
      if (x[10] <= -68.5) {
        if (x[2] <= -87.0) {
          if (x[2] <= -89.5) {
            votes[0] += 1;
          } else {
            votes[1] += 1;
          }
        } else {
          votes[0] += 1;
        }
      } else {
        if (x[12] <= -70.0) {
          votes[1] += 1;
        } else {
          votes[2] += 1;
        }
      }
    }
    if (x[2] <= -41.0) {
      if (x[10] <= -71.0) {
        if (x[13] <= -73.0) {
          votes[0] += 1;
        } else {
          votes[2] += 1;
        }
      } else {
        votes[1] += 1;
      }
    } else {
      votes[2] += 1;
    }


    // return argmax of votes
    uint8_t classIdx = 0;
    float maxVotes = votes[0];

    for (uint8_t i = 1; i < 3; i++) {
      if (votes[i] > maxVotes) {
        classIdx = i;
        maxVotes = votes[i];
      }
    }
```

```
    return (lastRoomId = classIdx);
  }

  /**
         * Get current room name
         */
  template<typename Scanner>
  String predictRoomName(Scanner& scanner) {
    uint8_t roomId = predictRoomId(scanner);

    switch (roomId) {

      case 0: return (lastRoomName = "kitchen");

      case 1: return (lastRoomName = "room");

      case 2: return (lastRoomName = "toilet");

      default: return (lastRoomName = "???");
    }
  }

  /**
         * Test if current location is the given one by id
         */
  bool isIn(uint8_t roomId) {
    return roomId == lastRoomId;
  }

  /**
         * Test if current location is the given one by name
         */
  bool isIn(String roomName) {
    return roomName == lastRoomName;
  }


protected:
  uint8_t lastRoomId = 255;
  String lastRoomName;

  template<typename Scanner>
  void convertToFeatures(Scanner& scanner, float* x) {
    while (scanner.hasNext()) {
      String ssid = scanner.ssid();
      float rssi = scanner.rssi();

      scanner.next();
      if (ssid == "DISPLAY-03") {
        x[0] = rssi;
        continue;
      }

      if (ssid == "DISPLAY-01B") {
        x[1] = rssi;
        continue;
      }

      if (ssid == "LEARNING HUB 05") {
        x[2] = rssi;
        continue;
      }

      if (ssid == "GeniusPOS") {
        x[3] = rssi;
        continue;
      }

      if (ssid == "Galaxy A023651") {
        x[4] = rssi;
        continue;
      }

      if (ssid == "LEARNING HUB 46") {
        x[5] = rssi;
        continue;
      }

      if (ssid == "DIRECT-b6-HP M280 LaserJet") {
        x[6] = rssi;
        continue;
      }

      if (ssid == "TP-LINK_9118") {
        x[7] = rssi;
        continue;
      }

      if (ssid == "DIRECT-ef-HP M281 LaserJet") {
        x[8] = rssi;
        continue;
      }

      if (ssid == "EnergisingFutures") {
        x[9] = rssi;
        continue;
      }
      if (ssid == "NasiKandaq-2.4Ghz") {
        x[10] = rssi;
        continue;
      }

      if (ssid == "CERDAS") {
        x[11] = rssi;
        continue;
      }

      if (ssid == "AnchorAP1") {
        x[12] = rssi;
        continue;
      }

      if (ssid == "AnchorAP3") {
        x[13] = rssi;
        continue;
      }

      if (ssid == "TP-Link_CD36") {
        x[14] = rssi;
        continue;
      }
    }
  }
};

WiFiIndoorPositioning indoor;
String curr_loc="Current location: ";
String curr_pos="Patient is currently in ";

void setup() {
  Wire.begin();
  Serial.begin(115200);

  pinMode(pulsePin, INPUT);
  mlx.begin();
  mpu.initialize();
  pinMode(BUZZER_PIN, OUTPUT);

  WiFi.begin(SSID, PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
    //delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  thing["Temperature"] >> [](pson& out){
      out = temperature_display();
  };

  thing["Temperature Status"] >> [](pson& out){
      out =   temperature_status_display();
  };

  thing["Pulse"] >> [](pson& out){
      out = pulse_value();
  };

  thing["Positioning"] >> [](pson& out){
      out = patient_pos();
  };


  thing["Indoor Positioning"] >> [](pson& out){
      out = patient_loc();
  };

}

int pulse_value(){
  int bpm;
  pulseValue = analogRead(pulsePin);  // Read raw pulse sensor value
  Serial.print("Pulse Value : ");
  Serial.println(pulseValue);
  unsigned long currentMillis = millis();  // Current time in milliseconds
  if (pulseValue < 2000) { pulseValue = 0; }
  // Calculate BPM every interval
  if (currentMillis - previousMillis >= interval) {
    bpm = map(pulseValue, 2000, 4032, 70, 150);  // Map pulseValue to BPM
range (60-100)
    // Print the BPM value
    if (bpm < 0) { bpm = 0; }
    Serial.println(bpm);

    previousMillis = currentMillis;  // Update previousMillis
  }

  return bpm;

}
```

```
void loop() {
    thing.handle();
}

String temperature_status_display(){
  // Read the object temperature (the temperature of the body)
  float objectTemp = mlx.readObjectTempC();
  String fever= "Fever detected!";
  String normal= "Normal body temperature.";

  // Check if the body temperature exceeds the fever threshold
  if (objectTemp >= feverThreshold) {
    Serial.println(fever);
    return fever;
    // Add further actions here, e.g., alerting the user, activating an
alarm, etc.
  }
  else{
    Serial.println(normal);
    return normal;
  }
}

float temperature_display(){
  // Read the object temperature (the temperature of the body)
  float objectTemp = mlx.readObjectTempC();
  Serial.print("Body Temperature: ");
  Serial.print(objectTemp);
  Serial.println(" °C");

  return objectTemp;
}
// String Temp_status(){
//   Serial.println("Fever detected!");
// }
String patient_pos(){
  int16_t accelX, accelY, accelZ;
  int16_t gyroX, gyroY, gyroZ;
  mpu.getMotion6(&accelX, &accelY, &accelZ, &gyroX, &gyroY, &gyroZ);

  String fall = "Fall detected!";
  String sit = "Sitting/Walking detected.";
  String run = "Running detected.";

  // Convert raw accelerometer values to float
  float floatAccelX = accelX / 16384.0;
  float floatAccelY = accelY / 16384.0;
  float floatAccelZ = accelZ / 16384.0;
  // Apply a complementary filter to get the gravity component
  float accelMagnitude = sqrt(pow(floatAccelX, 2) + pow(floatAccelY, 2) +
pow(floatAccelZ, 2));
  accelMagnitude = alpha * accelMagnitude + (1 - alpha) *
lastAccelMagnitude;
  Serial.print("Value accel ");
  Serial.println(accelMagnitude);
  lastAccelMagnitude = accelMagnitude;

  // Detect different activities based on the magnitude of the acceleration
vector
  if (accelMagnitude < thresholdFall) {
    Serial.println(fall);
    digitalWrite(BUZZER_PIN, HIGH);
    return fall;
    // Take appropriate actions when a fall is detected
    // For example, send an alert or trigger an action
  } else if (accelMagnitude < thresholdSitting) {
    Serial.println(sit);
    return sit;
    // Take appropriate actions for sitting
  }else if (accelMagnitude < thresholdRunning) {
    Serial.println(run);
    return run;
    // Take appropriate actions for running
  }
  //delay(1000);  // Adjust the delay according to your requirements
}

String patient_loc(){

  Serial.print(curr_loc);
  Serial.println(indoor.predictRoomName(wifiScanner));

  String position= curr_loc + indoor.predictRoomName(wifiScanner);

    // customize as per your needs
  return position;
  //delay(1000);
}
```