European University Cyprus

School of Sciences | Department of Computer Science and Engineering

Post-Operation Vital Sign Monitoring Device

Bachelor of Science in Computer Science

Martin Aziz

Acknowledgements

Write some text for someone that you would like to say thank you for the completion of this work

Abstract

Write a page of text explaining the project, what you wanted to do, what was done, how it was done and what has been achieved.

Contents

Chapter 1

Introduction

1.1 Introduction

It is crucial to monitor patient vital signs in the post-operative period to ensure detection of any complications early on, and increase patient safety. Traditionally, nurses and clinical staff carry out this monitoring at regular intervals, which may leave gaps in observation and increase the risk of missing early warning signs. Advances in the Internet of Things (IoT) space have created new possibilities for real-time, continuous monitoring of patients using wireless sensors and communication protocols.

One of the main difficulties in implementing IoT systems in healthcare is selecting appropriate technologies for communication, that balance power consumption, range, and reliability, at a reasonable cost. Bluetooth, which is a widely used technology in consumer health devices, offers short-range, low-energy connectivity but it doesn't scale well to the scale of a hospital, for the purpose of monitoring hundreds of patients at a time. LoRaWAN on the other hand, is capable of communication at long ranges, using low power to transmit small amounts of data (perfect for sending the kinds of data sensors produce) from multiple devices to a central gateway.

This thesis presents the design and implementation of a device intended for the purpose of monitoring patient vital signs in the post-operative context, that integrates consumer medical sensing devices, and incorporates both Bluetooth as well as the LoRaWAN communication protocols. It is designed to collect data from commercially available medical devices and send that data efficiently to the cloud for analysis and alerting purposes.

1.2 Aims and Objectives

The main aim of this project is to design and develop an IoT based device to monitor post-operative vital signs. Several specific aims were defined, with clear objectives for each:

1. Investigate the clinical requirements and difficulties in the context of post-operative vital sign monitoring.
    - Conduct a literature review on the importance of post-operative monitoring.
    - Conduct a literature review on the clinical importance of each vital sign.
2. Evaluate and compare appropriate commercially available bluetooth capable medical devices for possible integration with the system.
    - Identify commercially avaialble bluetooth capable devices for measuring temperature, blood pressure, and heart rate.
    - Assess their Bluetooth connectivity and opennes, and as such determine their likely degree of integration with a custom system such as the one being developed.
3. Analyse and select suitable communication technologies for device integration and data transmission to cloud.
    - Review relevant communication protocols for device integration.
    - Review relevant communication protocols for long range data transmission to the cloud.
    - Justify the selection of technologies to be used in the system.
4. Design and implement an IoT system to monitor vital signs post-operation.
    - Develop the system architecture, consisting of the device itself as well as a simple backend cloud platform.
    - Make the device capable of connecting to and communicating with Bluetooth devices such as medical sensors.
    - Establish communication between the device itself and the backend cloud platform using LoRaWAN.
5. Test the system under realistic conditions and evaluate its performance.

- Conduct testing of the system with simulated data representing data incoming from a medical sensor, and test transmission via the LoRaWAN protocol to the cloud.
- Evaluate metrics such as data reliability, scalability, and energy efficiency.

6. Analyse the project outcomes and propose possible future improvements.
    - Discuss strengths, weaknesses, and potential points of improvement of the system.
    - Discuss any challenges or difficulties faced during design and development.
    - Suggest possible further research and enhancements.

1.3 Structure of the thesis

This thesis is organised into six main chapters, following the natural development process (starting with research and design, followed by implementation, ending with an analysis of the outcomes).

- **Chapter 1: Introduction** — Introduces the background, aims, objectives, and structure of the thesis.
- **Chapter 2: Background or Project Scope** — Reviews the clinical significance of monitoring vital signs after surgery, the role of IoT in this task, analyses current communication protocols and available medical sensing devices, and discusses some points of consideration regarding integration of said devices with the system.
- **Chapter 3: Analysis and Design** — Describes the 'user needs', requirements analysis, system architecture, detailed design of the system, and gives reasons for design desisions that were made.
- **Chapter 4: implementation and Testing** — Presents the implementation of features, implementation of communication with backend cloud via LoRaWAN, and testing procedures.
- **Chapter 5: Discussion** — Discusses the resulting system, difficulties encountered, lessons learned, and possible improvements.
- **Chapter 6: Bibliography** — Lists all the references cited throughout the thesis.

1.4 Summary

This chapter gave an introduction to the reasoning behind developing such an IoT based system for the post-operative scenario, gave an outline for the aims nd objectives of the project, and briefly covered the general structure of the thesis. The next chapter will dive deeper into the background of the project including clinical signigicance of vital sign monitoring, the role of IoT, relevant communication protocols, and compare and evaluate commercially available Bluetooth-capable medical sensing devices fo this project.

Chapter 2

Background or Project Scope

2.1 Introduction

The healing process does not end after surgery is complete, but instead it is often a long effort for weeks or months post-operation. Furthermore, it is not just the physical health of the patient which must be considered, but also and perhaps equally important their mental health and quality of life moving forward. In recent decades it has become clearer to medical professionals that extended monitoring following operations greatly impacts the results of surgeries in a positive manner, as stated in [1]. Of course, there are multiple factors affecting the extent to which complications will arise in a patient after receiving surgery, such as their age, previous health issues, or conditions which may develop in parallel to the primary diagnosis even if they are in fact unrelated. In particular, the elderly are quite vulnerable to more complications, as stated by [2], specifically referring to elderly patients who underwent hip fracture surgery. Such complications affect quality of life longterm as well as general ability to function normally, not just physical health, degree of recovery, and survivability [2].

The role of the surgeon in the postoperative context is to firstly make sure the patient is receiving the necessary support to sustain a healthy balance in their body, and do whatever they can to avoid the development of and complications that may appear as a result of the procedure. Should any complications arise despite the medical staff's best efforts, it is then the surgeon's responsibility to recognise the signs pointing to the development of said complications and take appropriate actions to quickly and effectively manage them, allowing the patient to recover to their preoperative state eventually [3]

2.2 Clinical significance of monitoring each vital sign post-surgery

Vital signs give a rich picture of patient condition post surgery, providing medical staff with the information required to adequately care for the patient. Different measurements have varying significance depending on the stage of recovery the patient finds themselves in, however they should be continuously measured to provide history and an indication of trends to clearly show the progress of the patient's condition. Immediately after surgery measurements must be taken often as that is the most critical stage of the recovery process. In the beginning, vitals such as respiratory rate and blood pressure are vastly more important due to their indication of how the patient is recovering from anesthesia, but once they sufficiently recover from it pulse rate is a better indicator of the volume of blood in the circulatory system of a patient [3]

2.3 Role of IoT in vital sign monitoring

2.4 Communication protocols in IoT-based monitoring

This section explores the communication protocols relevant to IoT-based monitoring of post-operative patients, focusing on Bluetooth and LoRaWAN. Reliable and efficient data transmission is crucial in this context, where vital signs such as heart rate, temperature, and blood pressure need to be collected with minimal power consumption and transmitted securely over short or long distances.

LoRa is a wireless modulation technique derived from chirp spread spectrum technology. It enables long-distance, low-power communication by encoding information on radio waves using chirp pulses. This makes it robust against interference and highly suitable for IoT applications that transmit small data packets at low bit rates [4]. Compared to technologies like WiFi, Bluetooth, or ZigBee, LoRa supports data transmission over significantly longer distances, particularly in sub-gigahertz bands, making it well suited for both indoor hospital and outdoor monitoring environments [5].

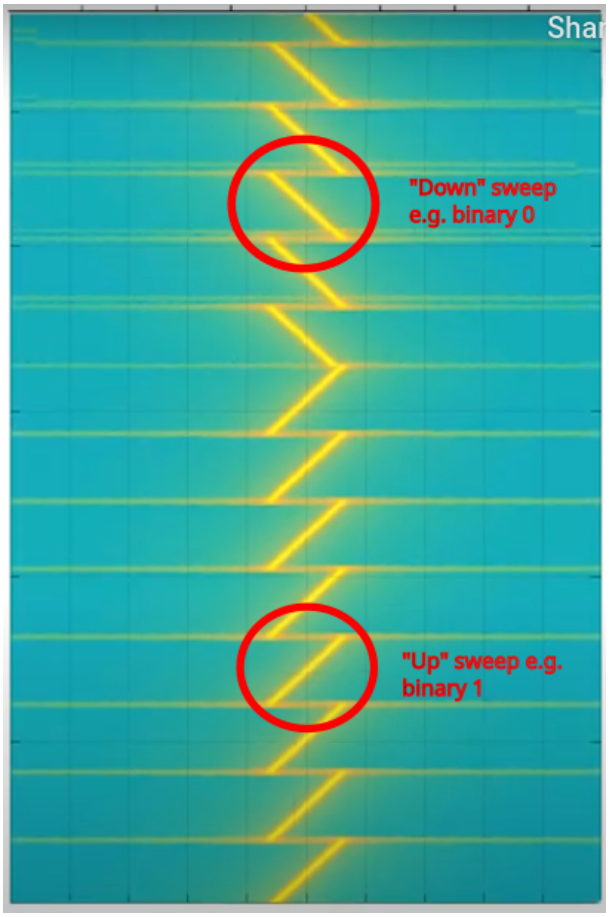

Figure 2.1: Example of a chirp spread spectrum transmission, highlighting the up and down sweeps which represent binary 1 and 0. [6]

LoRaWAN builds upon the LoRa physical layer by adding a media access control (MAC) protocol. It defines how devices connect, how data is encrypted, and how the network is managed. LoRaWAN is designed for ultra-low-power operations, allowing devices to last several years on small batteries, and offers deep indoor penetration, which is particularly valuable in hospital settings where walls and medical equipment can attenuate signals [5].

To assess the suitability of LoRaWAN, it is important to compare it to other LPWAN technologies, including SigFox, NB-IoT, and LTE-M. Below is a table summarising their main advantages and disadvantages.

Table 2.1: LPWAN technologies - Advantages & Disadvantages [7–10]

Technology	**Advantages**	**Disadvantages**
LoRaWAN	• Long range • Low power consumption • Adaptive data rates • Deep indoor penetration • Low cost	• Limited by duty cycle • Unsuitable for real-time or low latency applications
SigFox	• Lightweight protocol • Minimal overhead • Long battery life • Wide coverage • Underground support	• One-way communication • Limited data rates • One operator per country
NB-IoT	• High scalability • Robust QoS • Extended range • Structure penetration • Backed by operators	• Licensed spectrum costs • Lower data rates than LTE-M • No roaming support
LTE-M	• High data rates • Excellent coverage • LTE network integration • Efficient power use	• Higher costs • Firmware updates consume power • Not for high-volume data

Bluetooth is also widely used in medical IoT devices. While it offers short-range connectivity and is well-suited for personal health devices like heart rate monitors and themometers, its power consumption and range limit its usefulness for hospital-wide or remote patient monitoring. Bluetooth Low Energy (BLE) improves power efficiency and is increasingly integrated into wearable medical devices.

sectionCommercially available medical devices

Testing

While the focus of this project was not to benchmark the LoRaWAN protocol itself, preliminary simulations were conducted to validate its suitability for post-operative monitoring applications. A single base station scenario was simulatedusing open-source software provided by Lancaster University [11]. The simulation was adapted to run on Python 3, and tests were conducted with 100 and 200 node configurations.

The key objective was to confirm that LoRaWAN could handle multiple nodes transmitting small amounts of data at low power over long distances. Results showed that as the number of nodes doubled, the number of collisions also increased, which is consistent with LoRaWAN's access behaviour. Despite the increase in collisions, transmission efficiency per node remained stable, and energy consumption scaled predictably with the node count. These outcomes support the choice of LoRaWAN as a communication backbone in the system, balancing scalability and energy efficiency in a medical monitoring context.

2.5 Integration with custom IoT systems

2.6 Summary

Summarise what was said in this chapter and link with the next chapter.

Chapter 3

Analysis and Design

3.1 Introduction

This chapter will present the analysis and design porition of the project. Identified are the system users and their needs, which are then translated into technical requirements, and finally the system's functional and architectural design is described. Several diagrams are included to aid with visualising the behaviour, workflow, and structure of the system.

3.2 User Needs and Requirements

When developing any technology it is imperative to have a solid foundation upon which to build on and refer to throughout the development process. It should help provide a clear understanding of the problem, the needs expectations of the people who will be using it, and give some insight into how the original problem will be solved in the implementation stage. In the case of this project, the users include nurses and doctors, and the patients who will be using the device. There are several different users each with different required levels of control and functionalities, therefore it is crucial to translate their needs into measurable and clearly defined requirements that will steer the design and eventually implementation of the system.

The following pages present two key tables accomplishing that task: the **User Needs table** which outlines the what is required of the system by the users to solve their problem, and the **User Requirements table** which turns those needs into specific, quantifiable, and actionable requirements for the system to fulfil.

Table 3.1: User Needs Table

ID	**Description**	**Source**
UN-01	Accurate capture of data from various medical devices in order to monitor patient status	Arduino
UN-02	Simple pairing of the device to other medical devices through Bluetooth in order to easily setup and guarantee reliable data transfer	Arduino
UN-03	Updating/setting of limits/conditions for different measurement types in order to customise each device to a particular patient's needs	Arduino/ Cloud/ Medical staff
UN-04	Process data according to preset conditions in order to send only relevant data to the cloud	Arduino
UN-05	Storing of data in the short-term in the case of high network congestion to prevent loss. (Only store after processing if values are abnormal/relevant to be sent to the cloud)	Arduino
UN-06	Alert medical staff of patient status if critical or abnormal reading gathered in order to allow them to check on said patient in person	Cloud
UN-07	Alert users about device status in order to allow them to keep it in a state where it is running normally	Arduino
UN-08	Provide a general picture about the patient's status daily in order to track it over time (daily averages viewed in context of weeks, months, years)	Cloud

Table 3.2: User Needs Table

Req. ID	**User Requirement Name**	**Description**	**Justification/Comment**	**Reference**
UR-01	Temperature Reading	5 times/12 hours Accuracy: to nearest 0.1°C	5 readings in a single 12 hour period should be enough to detect any changes in temperature early enough after get out of bounds	UN-01
UR-02	Blood Pressure Reading	1 time/12 hours Accuracy: ±3mmHg for both systolic and diastolic	The recommended number of times to measure BP is once per day	UN-01

Req. ID	User Requirement Name	Description	Justification/Comment	Reference
UR-03	Heart Rate Reading	3 readings spaced 2 minutes apart / hour, and keeping the average of the 3 readings as final BPM value Accuracy: ± 5 BPM	In a clinical/hospital environment there would be constant EKG measurement so there is no reason medically not to measure heart rate as often as possible. However to conserve battery, avoid network congestion, and keep the processor constantly working, 3 readings averaged on the hour is sufficient for this system	UN-01
UR-04	Bluetooth Pairing with Medical Devices	Visual interface to display BT connection status, and allow force disconnecting a device which has connected if it is not a medical device	Pairing with a device should be incredibly easy seeing as a likely user group will be elderly individuals	UN-02

Req. ID	User Requirement Name	Description	Justification/Comment	Reference
UR-05	Per Vital Sign Configuration	Visual interface to be able to select a vital sign that will be measured, and set the upper and lower acceptable limits for it. This must persist across power loss and general restarting of the device. In addition, the limits must make sense in the context of the vital sign they relate to. For example it should be impossible to set an upper limit of 60°C for temperature	Each patient has different medical needs and as such will need customised ranges for the acceptable readings. Furthermore, the patients should not be required to have to set these limits in the event of a power loss of the device or device restart	UN-03

Req. ID	User Requirement Name	Description	Justification/Comment	Reference
UR-06	Data Processing and sending to the cloud	Process each data reading in the context of the device type's preset limits (lower/upper). Readings which are below/above the limits will be added to a queue. The front data will be sent to the cloud using the LoRaWAN protocol. Once confirmation is received that the data is successfully received, it will be dequeued and the next data (if it exists) will be transmitted. Allow up to 3 attempts before considering that there is a problem with the device/connection to the network and alerting the user.	Data that is within the acceptable limits does not need to be sent. The queue's purpose is to ensure any critical readings get to the cloud platform where the medical professionals can respond accordingly.	UN-04 UN-05

Req. ID	User Requirement Name	Description	Justification/Comment	Reference
UR-07	Notification of medical staff for abnormal reading	After sending data through the LoRaWAN protocol to the cloud, create some kind of notification for the medical staff in the form of of an email or "ticketing" system	A popup notification is not sufficient because it might be missed	UN-05 UN-06
UR-08	Device status information	Constantly monitor device status (BT connection, state) and display this information to the user	Due to the fact that status information must always be visible, and we do not want to occupy the LCD screen with it all the time, the information will be available through coloured LED lights	UN-07
UR-09	Patient status tracking over time	Regardless of whether an abnormal reading is gathered, at fixed times of the day and for a set number of times, take readings and send them to the cloud even if they are not abnormal.	Patient history could be vital for providing medical staff with insight into the patient's condition, however there is no need to take more than one non-abnormal reading per day	UN-08

3.3 System Behaviour and Workflow

Note: Diagrams need some tweaking before I can include them in here.

State diagram: Describing system states (DISCONNECTED, CONNECTED, SETUP, READING, PROCESSING, TRANSMITTING)

Use case diagram: Showing interactions between users (nurses, doctors, patients) and the system

Activity flowchart: Visualise workflow

3.4 System Architecture and Design

Note: Diagrams need some tweaking before I can include them in here.

High-Level Architecture diagram: Overview of the system from the medical devices to the cloud with the BT module and arduino in the middle

Block diagram: Overview of circuit

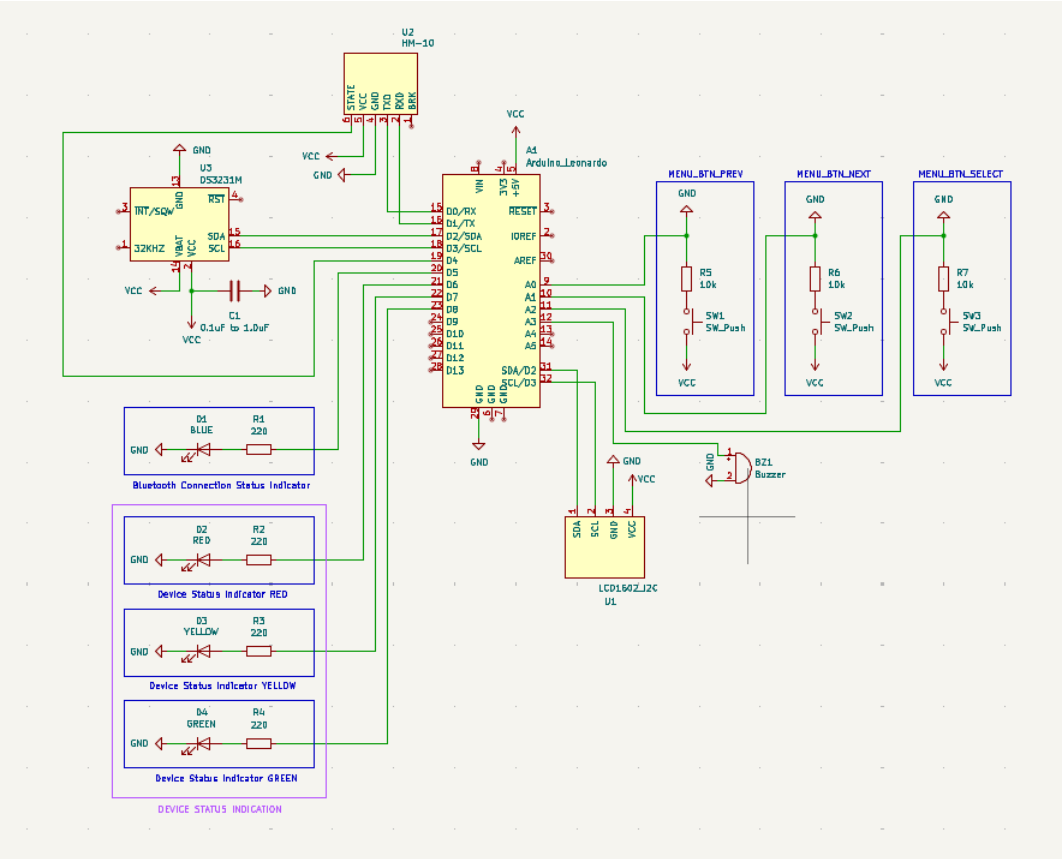

Figure 3.1: Detailed circuit shcemtatic of all all components in the device

3.5 User Interface Design

Note: Diagrams need some tweaking before I can include them in here.

Navigation Flow diagram: Show how a user might navigate around the device menus

Will discuss about usability and simplicity

3.6 Project Planning

Note: Diagrams need some tweaking before I can include them in here.

Gantt chart here

Will discuss how project timeline led to certain design decisions being made

3.7 Summary

This chapter expanded on the design of the system, starting from the user needs analysis, behaviour of the system, system architecture, and design of the interface. The provided diagrams show the organisation of the system, laying the foundation for the implementation which will be discussed in detail in the next chapter.

Chapter 4

Implementation and Testing

4.1 Introduction

Introduce what this chapter is going to present.

4.2 Implementation

4.2.1 Next subsection

4.2.2 Next subsection

4.3 Testing

4.3.1 Next subsection

4.3.2 Next subsection

4.4 Summary

Summarise what was said in this chapter and link with the next chapter.

Chapter 5

Discussion

5.1 Introduction

Introduce what this chapter is going to present.

5.2 Discussion

5.3 Difficulties phased

5.4 Knowledge acquired

5.5 Future work

5.6 Conclusions

Bibliography

Bibliography

[1] T. A. D'Amico, "Defining and improving postoperative care," *The Journal of Thoracic and Cardiovascular Surgery*, vol. 148, no. 5, pp. 1792–1793, 2014.

[2] S. K. Kare, K. R. Kumar, L. P. Kumar, and N. Naidu, "Post-operative complications in elderly patients undergoing hip fracture surgery: An observational study," *Asian Journal of Medical Sciences*, vol. 15, no. 3, pp. 191–195, 2024.

[3] E. A. Surwit and T. Y. Tam, "Postoperative care," Jan 2008. [Online]. Available: https://www.glowm.com/section-view/heading/Postoperative%20Care/item/36

[4] "What are LoRa and LoRaWAN?" https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/, accessed: 2025-5-8.

[5] "LoRa — LoRa documentation," https://lora.readthedocs.io/en/latest/, accessed: 2025-5-8.

[6] R. Wenner, "LoRa CHIRP," 17 2017.

[7] "Sigfox: Advantages and disadvantages," https://www.rfwireless-world.com/Tutorials/advantages-and-disadvantages-of-Sigfox-wireless-technology.html, accessed: 2025-5-8.

[8] "NB-IoT: Advantages and disadvantages," https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-NB-IoT.html, accessed: 2025-5-8.

[9] "LTE-M: Advantages and disadvantages," https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-LTE-M.html, accessed: 2025-5-8.

[10] "LoRaWAN: Advantages and disadvantages," https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Lora-or-LoRaWAN.html, accessed: 2025-5-8.

[11] "No title," https://www.lancaster.ac.uk/scc/sites/lora/lorasim.html, accessed: 2025-5-8.

Appendices

Appendix A

Write a few words about what is included in this appendix

Appendix B

Write a few words about the contents of the appendix

Manuals

User Manual

This device is designed to collect key health metrics in the post-operative phase via Bluetooth, and help you and your medical professionals monitor said vital signs. The three measurements it is capable of collecting and monitoring currently are blood pressure, temperature, and heart rate. It allows setting ranges for each metric, so that if a reading is outside these ranges it can automatically send it to the cloud and alert your medical professionals in order to allow them to more efficiently take care of you.

5.6.1 Getting Started

Power On

- Plug in device or add 6xAA standard batteries
- The LCD display will briefly display a message informing you it is booting, followed by "No connection" if there is no device currently set to auto-connect within Bluetooth range, or "Read" if there is a device within range and it automatically connects.

5.6.2 Menu Navigation

Buttons

- **Prev** (left)
- **Select** (middle)
- **Next** (right)

Basic Actions

- **Prev/Next:** Move the highlighted option or adjust values
- **Select:** Confirm choice or save a value

5.6.3 Main Menus

Device has no active Bluetooth connections

- **No connection**
  - Displays when there are no active Bluetooth connections
  - Pressing select on this option has no effect
- **Setup**
  - Allows setting up the acceptable range for each vital sign, where values read from Bluetooth which are within this range will be considered healthy readings
  - Pressing select on this option switches to the Setup menu

Device has at least one active Bluetooth connection

- **Read**
  - Enter mode where waiting for incoming Bluetooth data from a paired sensor
  - Buttons have no effect here. When data comes in, device will automatically check the data or remain in this mode if invalid data received.
- **Setup**
  - Allows setting up the acceptable range for each vital sign, where values read from Bluetooth which are within this range will be considered healthy readings
  - Pressing select on this option switches to the Setup menu
- **Disconnect**
  - Disconnect all connected sensors **Note: Not implemented yet**

Menu to set acceptable ranges for each vital sign

- **Setup BP**
  - Set minimum and maximum for systolic and diastolic blood pressure
  - Prev button decreases the value, Next increases it. Select confirms the value.
- **Setup Temp**

  - Set minimum and maximum for temperature.
  - Prev button decreases the value, Next increases it. Select confirms the value.
- **Setup HR**
  - Set minimum and maximum for heart rate.
  - Prev button decreases the value, Next increases it. Select confirms the value.
- **Back**
  - Returns to the previous menu
- The maximum can not be less than the minimum for any given vital sign. For example, concerning the systolic blood pressure maximum, it must be equal to or more than the minimum for systolic blood pressure.

5.6.4 Configuring Acceptable Ranges

Blood Pressure (BP)

1. Select **Setup BP**
2. Adjust **SYS min** (systolic mimimum) until reached desired value
3. Press the Select button to commit that value
4. Adjust **SYS max** (systolic maximum) until reached desired value
5. Press the Select button to commit that value
6. Adjust **DIA min** (diastolic mimimum) until reached desired value
7. Press the Select button to commit that value
8. Adjust **DIA max** (diastolic maximum) until reached desired value
9. Press the Select button to commit that value
10. After the fourth value, return to Setup

Note: If you accidentally set a "max" below its corresponding "min", the device will display a short error message (Err: Max < Min. Re-enter. It will then prompt you to re-enter both values.

Temperature (TEMP)

1. Select **Setup Temp**
2. Adjust **TEMP min** until reached desired value
3. Press the Select button to commit that value
4. Adjust **TEMP max** until reached desired value
5. Press the Select button to commit that value
6. Returns to Setup

Heart Rate (HR)

1. Select **Setup HR**
2. Adjust **HR min** until reached desired value
3. Press the Select button to commit that value
4. Adjust **HR max** until reached desired value
5. Press the Select button to commit that value
6. Returns to Setup

5.6.5 Taking Readings

1. **Ensure you have at least one paired external sensore of type BP/TEMP/HR**
2. In Connected menu, select **Read**
3. The device listens coninuously until it gets a valid reading. When a new reading arrives, it is automatically processed and transmitted to the cloud if necessary.

5.6.6 Readings In Range or Out of Range

- Within range:
  - No action - device remains in Connected state
- Out of range:
  - The device transitions to the Transmitting state and sends the reading to you cloud central console via LoRaWAN
  - You or your medical professionals can then review the reading remotely.

5.6.7 Maintenance & Care

- Keep the LCD and buttons free from dust and moisture.
- Store in a dry, room-temperature environment.
- Clear with a soft, dry cloth - do not use solvents or other liquids.

Technical Manual

This manual is intended to be read by people wishing to gain a better understanding of the structure of the codebase for this project, so that they can effectively extend, modify, or otherwise change it. This firmware implements a finite-state machine to manage all functions of the device. These include:

- Connection handling
- Configuration of the acceptable ranges for Blood Pressure, Temperature, and Heart Rate
- Data acquisition, validation, threshold checking, and transmission to cloud via LoRaWAN

All user interaction with the device takes place on a 16x2 I^2C LCD, navigated by 3 physical buttons.

5.6.8 Directory & File Structure

```
/src
 +-- globals.h           // Pin, global variable, & EEPROM
   ↪ address definitions, extern globals
 +-- menu.h/.cpp         // Menu definitions & handlers
 +-- states.h/.cpp       // State functions & FSM table
 +-- utils.h/.cpp        // Button debounce, logging, value
   ↪ adjustment, validation
 +-- Waveshare_LCD1602.* // Customized LCD driver
 +-- vital_monitor.ino   // setup() / loop() wiring FSM +
   ↪ peripherals
```

5.6.9 Global Definitions & Naming Conventions

Macros:

- `BTN_PREV`, `BTN_SELECT`, `BTN_NEXT`: GPIO pins connected to the Previous, Select, and Next buttons respectively

- `LED_BLUE`, `LED_GREEN`, `LED_YELLOW`, `LED_RED`: GPIO pins driving the four status LEDs
- `BT_STATE`: GPIO pin reading the Bluetooth module's connection status (HIGH = connected)
- `G_*_ADDR`: EEPROM addresses for storing each threshold, where * will have information about the vital sign, and whether it is a min or a max. All uppercase, and are prefixed with **G_**

    e.g. `G_TEMP_MIN_ADDR` (the address storing the minimum threshold for temperature) or `G_BP_SYS_MAX_ADDR` (the address storing the maximum threshold for blood pressure, specifically the systolic value)
- `G_*_THRESHOLD_MIN, G_*_THRESHOLD_MAX`: Default (factory) minimum and maximum allowable values for each vital sign. All uppercase, and are prefixed with **G_**

Global variables:

All lowercase, and are prefixed with **g_**

- **Button states:**
  - `g_prev_button_state`, `g_select_button_state`, `g_next_button_state`: Debounced current readings of the Prev/Select/Next buttons.
- **Menu navigation:**
  - `g_current_option_index`: Index of the currently highlighted menu entry.
  - `g_last_option_index_displayed`: Last index sent to the LCD (to avoid unnecessary redraws every loop).
  - `g_selection_pending`: Flag indicating a Select press is awaiting handling.
- **Threshold values:**
  - **Blood pressure**
    - `g_bp_systolic_threshold_min`, `g_bp_systolic_threshold_max`: User-configurable BP bounds (uint8_t)
  - **Temperature**
    - `g_temp_threshold_min`, `g_temp_threshold_max`: User-configurable TEMP bounds (uint16_t). Stored as the temperature×10 (e.g. 36.5°C stored as 365 to save 2 bytes in memory compared to storing it as a float or double

which would require 4 bytes).

  - **Heart rate**
    * `g_hr_threshold_min`, `g_hr_threshold_max`: User-configurable HR bounds (uint8_t).

- **Finite-State Machine Control:**
  - `g_current_state`, `g_previous_state`: Current and last state in the finite-state machine.
  - `g_setup_caller_state`: Remembers which state launched the Setup menu (for "Back" logic, after entering a setup sub-state such as "Setup BP", which would then set `g_previous_state` to the "setup" state instead of "Disconnected" or "Connected").
  - `g_multi_reset`: Signals entry into a multi-step threshold setup to initialise its values without needing to create a separate sub-state for each individual value (for example no need to create a sub-state for each of the following: systolic minimum, systolic maximum, diastolic minimum, diastolic maximum).
- **Data buffer:**
  - `g_received_data_buffer[G_RECEIVED_DATA_BUFFER_SIZE]`: Holds the latest NUL-terminated Bluetooth message (e.g. "BP:120/80").
- **Debug:**
  - `debug_enabled`: Enables or silences `log_msg("DEBUG", ...)` output.

State Enumeration (states):

The `states` enum defines each major step of the device's operation:

- `DISCONNECTED`

  No active Bluetooth connections; waiting to connect. User may enter `SETUP`.
- `CONNECTED`

  Bluetooth is up; user may begin reading data, enter `SETUP`, or disconnect the connected Bluetooth devices.
- `SETUP`

  Main configuration menu, where user can choose to edit BP, TEMP, or HR thresholds.
- `SETUP_BP`, `SETUP_TEMP`, `SETUP_HR`

Sub-menus for adjusting blood pressure, temperature, or heart-rate limits.

- `READING`

  Actively listening to receive a measurement string from the Bluetooth module.

- `PROCESSING`

  Parsing the incoming data and checking it against the configured thresholds.

- `TRANSMITTING`

  Sending an out-of-range measurement onward via LoRaWAN, then returning to `CONNECTED`.

5.6.10 State Machine Architecture

`stateTable[]`

An array mapping each `states` value to its handler `StateFunc`

```
struct StateTable stateTable[] = {
        {DISCONNECTED, state_disconnected},
        {SETUP, state_setup},
        {SETUP_BP, state_setup_bp},
        {SETUP_TEMP, state_setup_temp},
        {SETUP_HR, state_setup_hr},
        {CONNECTED, state_connected},
        {READING, state_reading},
        {PROCESSING, state_processing},
        {TRANSMITTING, state_transmitting}
};
```

`loop()` & **Transition Logic**

```
if (g_current_state in {DISCONNECTED, CONNECTED, SETUP})
        next_state = handle_menu(g_current_state);
else if (g_current_state == READING)
        next_state = state_reading();
... etc ...
next_state = check_bt_connection(next_state);
```

```
if (next_state != g_current_state)
        change_state(next_state);
```

`check_bt_connection()`

- Monitors the `BT_STATE` input pin to detect stable Bluetooth connection or disconnection events, and drives FSM transition accordingly.
- Suppresses connection checks during certain states where it is irrelevant (`PROCESSING`, `TRANSMITTING`, `SETUP`, `SETUP_BP`, `SETUP_TEMP`, `SETUP_HR`,).
- Returns a `states` enum value, either:
  - Unchanged (`current_state`)

    If still within the initial stabilisation period, or if in a state that should ignore Bluetooth changes.
  - `CONNECTED`

    Upon detecting a sustained HIGH on `BT_STATE`.
  - `DISCONNECTED`

    Upon detecting a sustained LOW on `BT_STATE` after a previous HIGH.

`change_state()`

- Updates `g_previous_state` and `g_setup_caller_state` (for Setup menu "Back").
- Clears LCD and resets menu indices when entering new state.
- Triggers LED update.

5.6.11 Menu System

Menu Tables (`menu_table[]`)

- Disconnected: `{"No connection", "Setup"}`
- Setup: `{"Setup BP", "Setup Temp","Setup HR", "Back"}`
- Connected: `{"Read", "Setup", "Disconnect"}`
- Reading/Processing/Transmitting: single-item static menus (no prev/select/next button functionality)

`handle_menu()`

1. Calls `handle_menu_options_buttons()`, which:
    - Edge-detects Prev/Select/Next buttons.
    - Wraps navigation index so that pressing Next on the last entry wraps around to the first entry and vice-versa.
    - Returns 0-N where N is the number of options for that state's menu. Return value is the index of the selected option.
    - Returns 255 if no selection was made.
2. Once Select is detected, dispatches to a new state:
    - In `DISCONNECTED`: "Setup" → `SETUP`.
    - In `SETUP`, enters corresponding `SETUP_*` sub-state or returns via `g_setup_caller_state`
    - In `CONNECTED`:
      "Read" → `READING`,
      "Setup" → `SETUP`,
      "Disconnect" → `DISCONNECTED`

5.6.12 Button & LCD Utility Functions

`debounceReadButton()`

Standard 20ms software debounce; tracks `stable_state`.

`log_msg()`

Takes two required parameters: the level of the log message, and the message itself. There are two overloaded versions which take a third optional argument (`const bool` and `unsigned`). Respects `debug_enabled`.

`handle_value_adjust_u8, handle_value_adjust_u16`

- Support tap for single increment/decrement, as well as hold for auto-repeat.
- Parameterised by step size, hold delay, and repeat interval. Step size is the amount to increment/decrement, hold delay is how long to wait before considering a button

press as a hold, and repeat interval is used to control speed of press and hold increment/decrement.

`validate_message()`

- Ensures `g_received_data_buffer` matches one of:
  - **BP:2-3 digits/2-3 digits**
  - **TEMP: dd.d**
  - **HR:2-3 digits**
- Rejects malformed strings before processing.

5.6.13 Threshold-Setup Framework

Two template functions:

- `multi_threshold_setup_u8()` for 8-bit thresholds.
- `multi_threshold_setup_u16()` for 16-bit temperature thresholds.

Behaviour on entry (`g_multi_reset` flag):

- Reset `step`, `last_drawn`, `last_select`
- Clamp existing `*values[i]` to `[lo[i]..hi[i]]` blocking the user from incrementing/decrementing past the default minimum and maximum allowable values for each vital sign (`G_*_THRESHOLD_MIN` and `G_*_THRESHOLD_MAX`).

Per-step UI loop:

1. Clear & redraw prompt + current `*values[step]`.
2. Call `handle_value_adjust`.
3. On **Select** button rising edge:
   - If max<min (for paired steps), display error and repeat this step.
   - Otherwise write changed values to EEPROM.
   - Advance `step`; if `step == count`, return to `previous_state`.

5.6.14 Bluetooth Data Flow

1. `state_reading()`
   - Monitors `BT_STATE` pin for disconnect.

- Reads bytes until `\n`.
- Strips optional `\r`.
- Validates via `validate_message()`.
- Sends `ACK` or `RETRY` over Bluetooth module's UART.
- On valid data, copies it into `g_received_data_buffer` and transitions to the `PROCESSING` state.

2. `state_processing()`
   - Parses buffer depending on prefix ("BP:", "TEMP:", "HR:").
   - Checks parsed values against `g_*_threshold_{min,max}`.
   - If out-of-range, returns `TRANSMITTING` state, otherwise returns `CONNECTED` state.

3. `state_transmitting()`
   - Placeholder to send `g_received_data_buffer` to cloud via LoRaWAN (Not implemented yet).
   - Always returns `CONNECTED` state.

5.6.15 EEPROM Storage

- Addresses:
  - **BP:** bytes 0-3 (one byte each, min/max for systolic and diastolic)
  - **TEMP:** 4-7 (two bytes each, min/max) (uint16_t little-endian)
  - **HR:** 8-9 (one byte each, min/max)
- Initialisation (in `setup()`):
  - Read each address; if uninitialised (0xFF/0xFFFF) as they will be on the very first boot of the device, load default macros (`G_*_THRESHOLD_MIN`, `G_*_THRESHOLD_MAX`)

    For temperature which is a uint16_t, need to read in a special way:

    ```
    g_temp_threshold_min = EEPROM.read(G_TEMP_MIN_ADDR) |
            (EEPROM.read(G_TEMP_MIN_ADDR + 1) << 8);
    ```

    Read the low byte first and logical OR it with the high byte, shifted 8-bits left.

- Writing:
  - Only write when new value $\neq$ stored value to minimise wear on the EEPROM cells (limited writes).

5.6.16 LCD Driver Adjustments

- Based on manufacturer's Waveshare_LCD1602 library
- Had to comment out the line: `Wire.setPins(4, 5);`