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

Introduce the project briefly.

1.2 Aims and Objectives

Mention what you aim to do for project. For every aim you have you must also mention what are the objectives that you must achieve in order to fulfil the aim.

1.3 Structure of the thesis

Explain to the reader what is included in the remainder of the thesis.

1.4 Summary

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

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

2.5 Commercially available medical devices

2.6 Integration with custom IoT systems

2.7 Summary

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

Chapter 3

Analysis and Design

3.1 Introduction

Introduce what this chapter is going to present.

3.2 Next section title

3.3 Next section title

3.4 Summary

Summarise what was said in this chapter and link with 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

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);`