CareTechTrio: Ammar Saif, Egzon Barja, Jesper Holmström

Heart rate and HRV monitor

Project Report

First-year Hardware Project

School of ICT

Metropolia University of Applied Sciences

12 May 2023

Version 2.4

Abstract

This project aimed to design and develop software for an interactive embedded system that can detect pulse and use the collected data to analyze heart rate variability parameters. The optical photoplethysmography (PPG) technique was used to detect the heart signal and extract peak-to-peak intervals. The displayed parameters were partially calculated locally with created algorithms.

The Raspberry Pi Pico W was used for implementation, along with the CrowTail Pulse Sensor v2.0 as the hardware sensor. Additionally, other components of the protoboard, such as the rotary encoder, OLED display, And LEDs, were also utilized. The programming language selected for the project was MicroPython, and Thonny was used as the IDE.

The project was successfully executed by the team's collective effort, and the product meets the set requirements with few limitations. These limitations include the inability to save the history of previous measurements and the lack of an option to save data to Kubios cloud.

For future development, the previously stated limitations could be implemented into the existing program. By improving the current peak detection algorithm, the accuracy of the program could also be increased.

Version history

Ver	Description	Date	Author(s)
1.0	Created structure for the project report. Added instructions for what should be included in the different parts of the document.	13.3.2023	Saana Vallius
1.1	Added Instructions (REMOVE WHEN READY), minor editions.	14.3.2023	Sakari Lukkarinen
1.2	Highlighted all instructions with red font.	19.3.2023	Sakari Lukkarinen
1.3	Updated Appendix 2 based on comments. Updated table of contents.	22.3.2023	Saana Vallius
2.0	Added a section for group work summary. Changed the highlighted instructions to an easier color to handle.	22.10.2023	Saana Vallius
2.1	Changed the structure and added subsections to each heading	8.5.2024	Ammar Saif, Egzon Barja, Jesper Holmström
2.2	Added content to each header and subsection to be proofread	8.5.2024	Ammar Saif, Egzon Barja, Jesper Holmström
2.3	Added references to the document	12.5.2024	Ammar Saif, Egzon Barja, Jesper Holmström
2.4	Proofreading done to the document	12.5.2024	Ammar Saif, Egzon Barja, Jesper Holmström

Contents

Definitions, acronyms, and abbreviations

ADC (Analog-to-Digital Converter)	A device that converts an analog signal into a digital signal.
Beats per minute (BPM)	The unit of measurement for heart rate.
FIFO (First In, First Out)	A data structure where the first item added is the first item removed.
GPIO (General Purpose Input Output)	Pins on a microcontroller that can be programmed for input or output.
GHz	A unit of frequency equal to one billion hertz.
Heart rate (HR)	The number of times the heart beats per minute.
Heart rate variability (HRV)	The time difference between two consecutive heartbeats.
I2C (Inter-Integrated Circuit)	A communication protocol used to connect low-speed devices.
IDE	Integrated Development Environment
LED (Light-Emitting Diode)	A light-emitting semiconductor device that operates when an electric current passes through it.
Milliseconds (ms)	The unit of measurement for heart rate variability.
MQTT (Message Queuing Telemetry Transport)	A protocol used for machine-to-machine communication that is lightweight and efficient.
OLED (Organic Light-Emitting Diode)	Organic compounds produce light in OLED display technology.
Photoplethysmography (PPG)	A technique that measures blood volume changes in the microvascular tissue of the skin.
PPI (Peak-to-Peak Interval)	The time between the highest point (peak) of one heartbeat and the highest point of the next heartbeat.
RMSSD (Root Mean Square of Successive Differences)	A measure of short-term variability in the time between heartbeats.
SDNN (Standard Deviation of Normal-to-Normal Intervals)	A measure of the overall variability in the time between heartbeats.
WLAN (Wireless Local Area Network)	A wireless network that connects devices in a limited area.

1 Introduction

The heart rate and HRV monitor project is a hardware project that is implemented during Hardware 2 at Metropolia University of Applied Sciences. The project's main goal is to create a hardware device capable of detecting heart rates and analyzing the variability. The project aims to teach first year ICT students the basic concepts of hardware engineering, resulting in a product that can be used in non-clinical environments by both trained and untrained users.

The documentation that accompanies the project explains the purpose and structure of the project, provides details about the necessary technical features needed for successful implementation, and gives insight into the challenges faced and the outcome of the project. The report is mainly intended for lecturers and students, but it also considers readers with little technical background knowledge.

The report is divided into six parts. Part 1 offers a brief introduction to the heart rate monitor. Part 2 describes the theoretical and scientific background of the measurements and indexes. Part 3 outlines the methods and materials used for this project. Part 4 presents the implementation of the project. Part 5 summarizes the group work done on the project, and Part 6 is the conclusion.

2 Theoretical Background

This section explains the scientific background of the project in two subsections. The first part introduces medical terms such as heart rate and heart rate variability. The second part focuses on the technical aspect of measuring heart rate.

2.1 Heart Rate and Heart Rate Variability

Heart rate is a measure of the number of times the heart beats per minute. It is an important indicator of cardiovascular health and can be used to monitor changes in the body's stress levels or physical activity. Heart rate variability (HRV) refers to the variation in time between successive heartbeats. HRV is an important measure of cardiac health and can be used to assess the body's response to stress, exercise, and other factors.

Heart rate and HRV are measured in beats per minute (BPM) and milliseconds (ms) respectively. Normal resting heart rate for adults is typically between 60 and 100 BPM, and HRV can range from 20 to 200 ms.

2.2 Heart Rate Detection

In this project, the Crowtail Pulse Sensor v2 is used to measure heart rate and HRV. The sensor is connected to a microcontroller that processes the sensor signal and calculates heart rate and HRV using various algorithms, including peak detection and time-domain analysis.

Peak detection algorithms work by identifying the peaks in the PPG signal that correspond to each heartbeat, while time-domain analysis algorithms work by analysing the time between successive heartbeats to calculate HRV.

PPG works by shining light into the skin using LEDs. Changes in blood flow within the microvascular tissue affect the amount of light reflected back to a photodiode sensor. This allows for indirect measurement of heart rate and variations in heartbeats - the basis for HRV calculations.

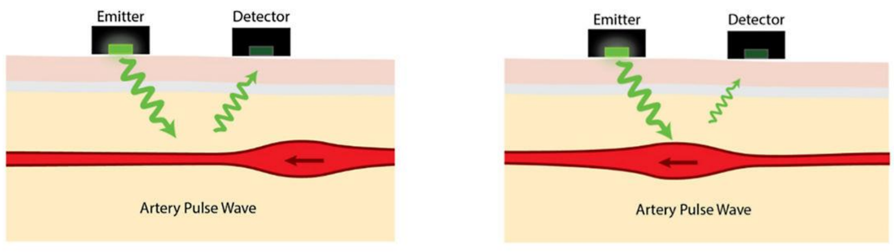


Figure 1: Example of a photoplethysmography device

3 Methods and Material

The next section details the materials used in the project. The hardware and software used are discussed in separate sections. The final section provides a summary of how the project functions.

3.1 Hardware Requirements

The Raspberry Pi Pico W microcontroller board was selected for this project due to its ability to support a wide range of peripherals. Additionally, it comes with a 2.4 GHz wireless LAN connection that is used to transmit HRV data to Kubios for analysis purposes.

Figure 2: Raspberry Pi Pico W

The SSD1306 OLED display is a type of display that can output information to the user. It is commonly used in various electronic devices and is known for its crisp and clear display. The display is designed to be user-friendly, allowing for easy reading and interpretation of the information being displayed. Whether you are using it for personal or professional purposes, the SSD1306 OLED display is a reliable and effective tool for displaying vital information.

Figure 3: SSD1306 OLED Display

The metal encoder of the rotary encoder serves three purposes: Rot Switch, ROT A, and ROT B. The Rot Switch behaves like a typical push button switch, while the ROT A and ROT B functions allow the user to rotate the encoder in clockwise and anticlockwise directions. The Rot Switch button function is used to choose the desired activity in the project.

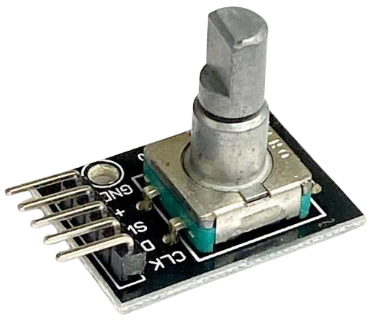


Figure 4: Rotary Encoder

The Crowtail Pulse Sensor 2.0 is a compact and easy-to-use device designed for measuring heart rate. It utilizes a photoplethysmography (PPG) sensing technique, which detects changes in blood volume by shining a light through the skin and measuring the amount of light that is absorbed or reflected.

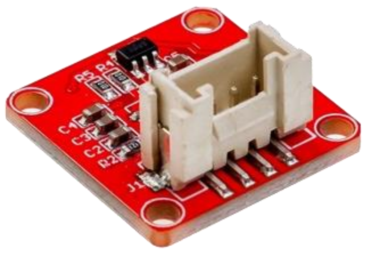


Figure 5: Crowtail Pulse Sensor 2.0

3.2 Software Requirements

Thonny IDE and MicroPython are the tools used to program the Raspberry Pi Pico W. Thonny IDE is a Python IDE that is specifically designed for use in embedded systems, while MicroPython is the programming language that is used for coding.

Figure 6: Thonny IDE Logo

Figure 7: MicroPython Logo

The Raspberry Pi Pico W can be programmed directly by downloading the MicroPython firmware via Thonny IDE. This firmware enables direct programming of the device.

MicroPython has fewer libraries than traditional desktop Python, but it has more library options that are specifically developed for embedded systems. One example of this is the ssd1306 library, which is used to control the OLED display and is dependent on the MicroPython-specific library FrameBuffer. Other libraries that are used in the project include machine, array, network, socket, urequests, and ujson.

The lecturers of Metropolia University of Applied Sciences have provided two additional files to assist with the project. The first of these files, fifo.py, provides a logical data storage solution that is interrupt-safe. The second file, piotimer.py, contains a class for hardware timing.

3.3 Functionality

The Crowtail sensor measures heart rate data and transmits it to a Raspberry Pi for processing. The microcontroller utilizes sophisticated algorithms to pinpoint peak-to-peak intervals within the signal. To maximize accuracy, these algorithms are meticulously calibrated using test data and a filtering process.

Key HRV (Heart Rate Variability) metrics are calculated from the collected peak-to-peak interval (PPI) data. These include:

- Mean PPI
- Mean heart rate
- SDNN (Standard Deviation of Successive Interval Differences)
- RMSSD (Root Mean Square of Successive Differences)
- HRV plot shape

The development board handles these calculations, presenting the results on the OLED screen.

Additionally, the peak-to-peak interval data is wirelessly sent to Kubios Cloud. Kubios performs advanced analysis to determine recovery and stress indexes. These results are then returned to the device and displayed on the OLED screen alongside the locally calculated HRV parameters.

The system incorporates a rotary encoder for user control. This allows users to select activities and initiate measurements based on information presented on the OLED display.

4 Implementation

Chapter 4 outlines the program and algorithms implemented in the project. The initial subsection reviews and summarizes the implementation requirements. The second subsection details the software implementation, with separate sub-subsections describing the functions and the logic of the code.

4.1 Implementation Requirements

The implementation of this project resolved around collecting the analog signal from the Crowtail Pulse Sensor v2.0. (For more information on the sensor, refer to Chapter 2.2. For optimal and accurate readings, direct contact between the human skin and the optical heart rate sensor is essential. Due to the variability of the strength of the pulse from person to person, it is recommended to identify the strongest pulse source beforehand for the most precise results.

The software for measuring the heart rate and the HRV analysis is developed on a single-board computer, Raspberry Pi Pico W. The coding aspect of the software is implemented through Thonny IDE, using MicroPython as the programming language. After calculating and measuring is done, results are displayed on the OLED screen. For data transmission, the software has MQTT integration, in this case a secondary Raspberry Pi Pico is used as a MQTT broker to publish HRV results via WLAN (Wireless Local Area Networks).
The peak detection algorithm carries out the main functionality of the device along with the calculations of all heart rate variability parameters.

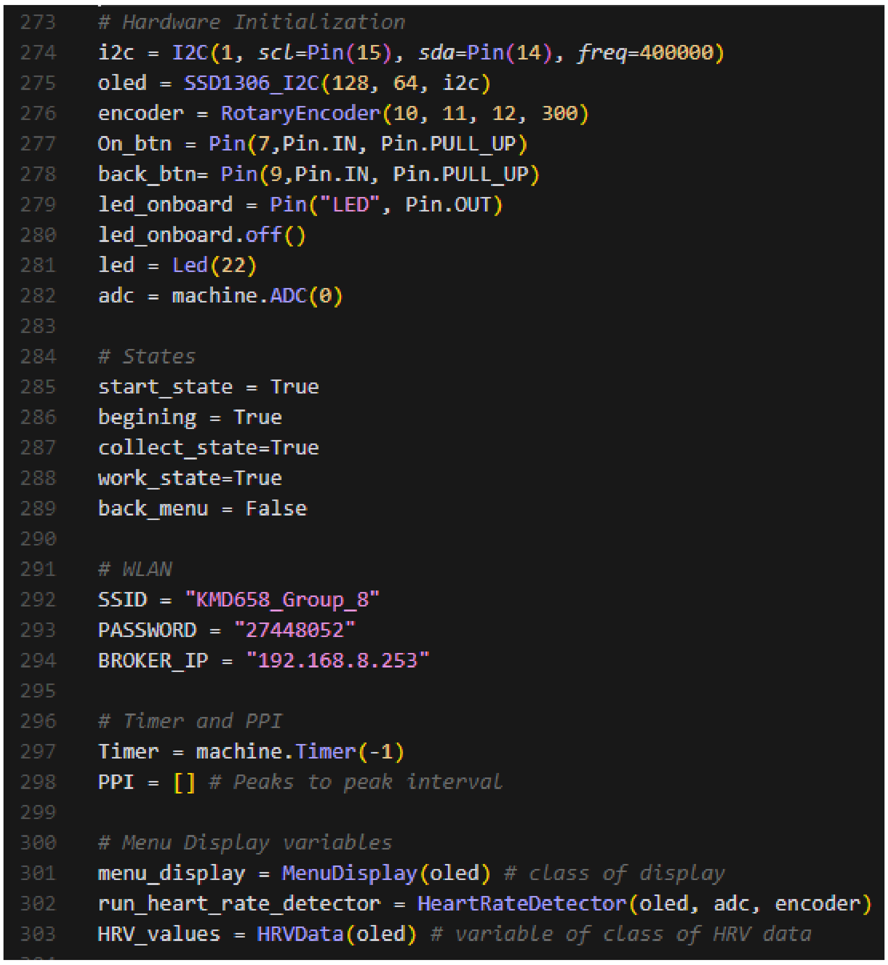
For a seamless workflow inside the team, the project was implemented throughout multiple steps. A Gitlab repository was utilized to store and access different versions of the project along with the final version (main.py).

4.2 Software Algorithms

To ensure successful execution of the program, several libraries are imported which provide the necessary functionalities for interfacing the hardware and handling data.
The ‘**machine**’ library is used to control the GPIOs (General Purpose Input Output pins), I2C (Inter-Integrated Circuit) ADC (Analog-Digital Converter), and Timer functionalities crucial for the operation of sensors and actuators.

The ‘**ssd1306**’ library is utilized to control the OLED display, allowing the program to display visual feedback and interfaces directly with the user.
The ‘**Fifo**’ libary is used to implement FIFO (first In, First Out) queues, which are essential for storing rotating events and managing data flow efficiently.
The ‘**utime**’ and ‘**piotimer**’ libraries are imported to provide the program precise timing operations, which prove to be essential for tasks such as measuring the time intervals between heartbeats or controlling the duration of data collection.

In addition to the libraries imported, the pins related to ADC, OLED, LEDs and rotary encoder are defined in the program. Then, a group of four global variables are separated and declared according to their purpose as shown in Figure 8.



```
# Hardware Initialization
i2c = I2C(1, scl=Pin(15), sda=Pin(14), freq=400000)
oled = SSD1306_I2C(128, 64, i2c)
encoder = RotaryEncoder(10, 11, 12, 300)
On_btn = Pin(7,Pin.IN, Pin.PULL_UP)
back_btn= Pin(9,Pin.IN, Pin.PULL_UP)
led_onboard = Pin("LED", Pin.OUT)
led_onboard.off()
led = Led(22)
adc = machine.ADC(0)

# States
start_state = True
begining = True
collect_state=True
work_state=True
back_menu = False

# WLAN
SSID = "KMD658_Group_8"
PASSWORD = "27448052"
BROKER_IP = "192.168.8.253"

# Timer and PPI
Timer = machine.Timer(-1)
PPI = [] # Peaks to peak interval

# Menu Display variables
menu_display = MenuDisplay(oled) # class of display
run_heart_rate_detector = HeartRateDetector(oled, adc, encoder)
HRV_values = HRVData(oled) # variable of class of HRV data
```

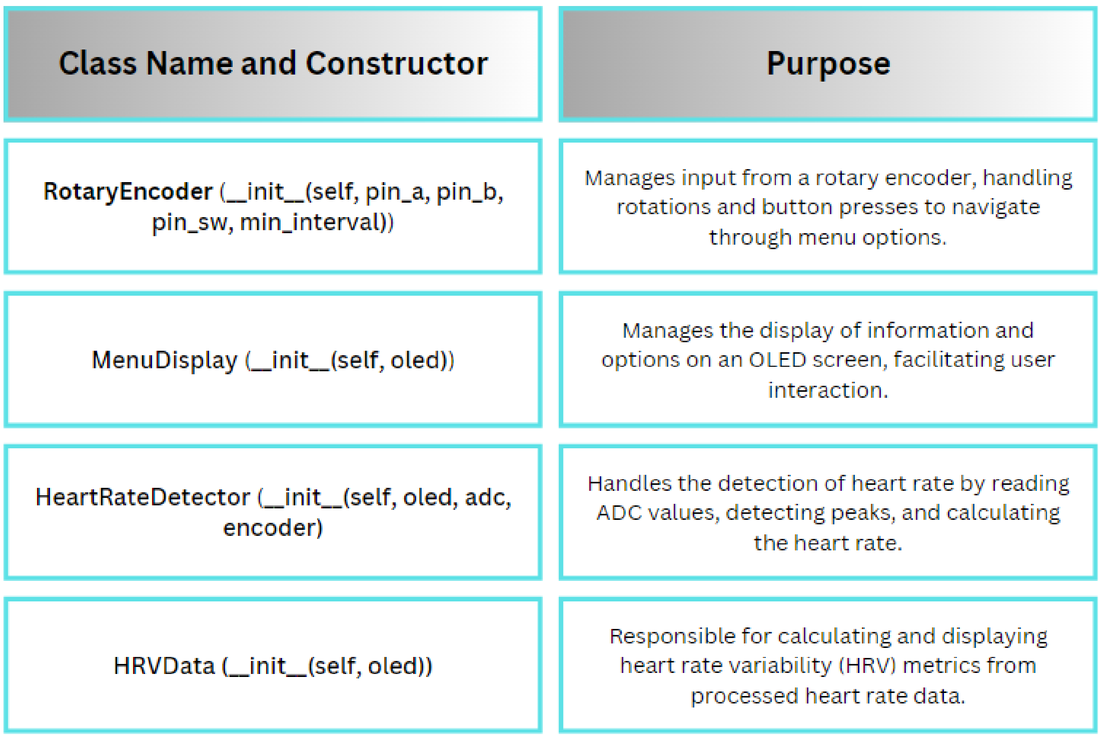
Figure 8: List of hardware setup alongside global variables

From Figure 8, the first group of global variables, the statement management variables control the initial state of the application. Typically, set to True to indicate the start of the program. Similarly, the rest of the statement variables; the beginning state, collecting state, work state and back menu state, control the flow of the application.

The variables in the second group, WLAN variables, are made of all the necessary credentials needed to connect to the WLAN. The third group of variables serve to schedule tasks at specific intervals such as stopping the ADC data collection after a certain set time. The last batch of global variables handles the interactions with the OLED display, the detecting of heart rate and computing various statistical measurements.

4.3 Classes

This program's functionality relies on using four classes listed in Table 1 below. The table defines the Classes' name and constructor in the column on the left and breaks down the purpose of the class in the column on the right.



Class Name and Constructor	Purpose
RotaryEncoder (__init__(self, pin_a, pin_b, pin_sw, min_interval))	Manages input from a rotary encoder, handling rotations and button presses to navigate through menu options.
MenuDisplay (__init__(self, oled))	Manages the display of information and options on an OLED screen, facilitating user interaction.
HeartRateDetector (__init__(self, oled, adc, encoder)	Handles the detection of heart rate by reading ADC values, detecting peaks, and calculating the heart rate.
HRVData (__init__(self, oled))	Responsible for calculating and displaying heart rate variability (HRV) metrics from processed heart rate data.

Table 1: The classes of the program

The rotary encoder class along with its constructor is designed to handle the user input from the encoder button as shown earlier in Figure 4. It serves to manage

to type of user inputs, rotational movements (for navigating through menu options) and button presses (for selecting preferred menu options).
The menu display class serves to manage and refresh the OLED display interface based on the user interaction and system status. It displays menu options, and all of the other relevant data such as the BPM readings and HRV metrics. In addition, it uses three specialized methods to display specific textual messages at various stages of the application.
Heart rate detector class is tasked with acquiring and processing the raw heart rate data collected from the analog signal via the ADC. It detects heart rate peaks, and calculates the BPM based on these detected peaks.
Additionally, this class takes several key arguments such as the OLED, ADC and ENCODER used for managing user inputs, reading, and displaying real-time information and results.
The HRV data class utilizes several methods for calculating the HRV analysis metrics. Each method calculates and computes different metrics such as meanPPI , meanHR, SDNN and RMSSD.

All these classes are designed to encapsulate specific aspects of handling, processing, displaying, and managing heart rate and HRV data within the application, offering an organized and modular approach to developing complex functions.

4.4 Main Program

The flow charts below are intended to provide a clear and concise structure of the software. It breaks down the concept of the software into smaller more manageable pieces, outlines the key steps and illustrates the critical elements that determine the direction of the program and its results.

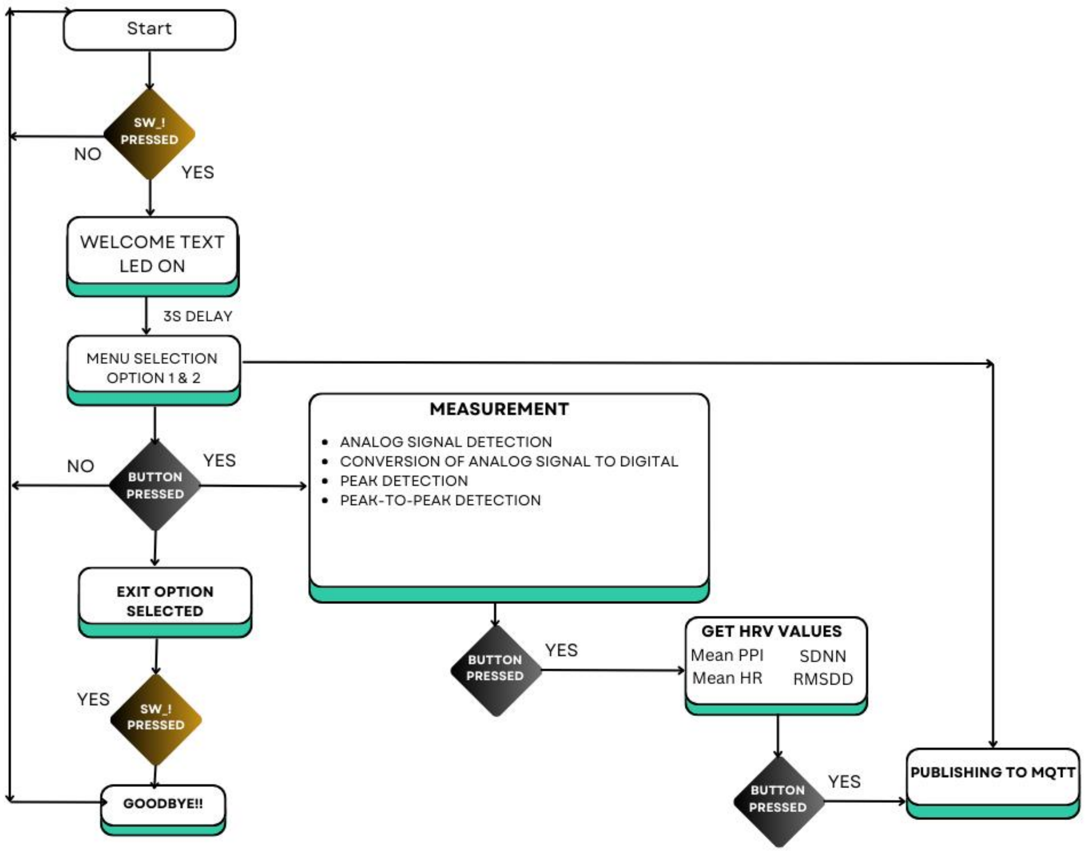


Figure 9: Flow chart of the program.

As displayed in Figure 9, the program begins by pressing the SW_1 button. As a sign to let the user know that the device is now on, LED9 is turned ON and remains ON until the device is turned OFF again. Subsequently, a welcome message is displayed. The welcome message is displayed for 3 seconds after which the application enters the main program and a While True loop. Here, the program waits for inputs from the rotary encoder either rotational movements or button pressed.

At this point, the program is in the menu selection state. User scrolls the menu with the rotary encoder and initiates the preferred option with the rotary encoder button. Selecting the menu options 1 and 2 moves through to the next state of the program; while selecting the last option (exit), the program breaks out of the loop and prompts the user with a goodbye message indicating the shutdown of the device.

Options 1 and 2, when selected, prompt the user to a waiting state serving as a preparation for the user to position the sensor in their strongest pulse location. It asks the user to press the rotary encoder button when ready to get a measurement. As the button is pressed, the program begins to sample heart rate signal. The program then goes on to capture data for 60 seconds and displays a reading every 5 seconds, each sample is stored as a buffer to calculate and compute the HRV values for later.

The algorithm detects a new peak by comparing the ADC value against a dynamically calculated threshold, the indexes of each peak are stored.
Later peaks are identified similarly, and the peak-to-peak interval (PPI) between consecutive peaks is computed as the difference between their indices, which is then converted to milliseconds. These intervals are then added to a list known as the PPI array, which stores all detected intervals. The **meanHR_calculator** function is triggered once there are enough entries in the PPI array to perform calculations, typically after collecting several intervals.

The mean heart rate (HR) is then continuously displayed on the top part of the OLED screen during the measurement process, along with each detected PPI. At this state, after completing the set measurement duration, if the PPI array has sufficient intervals (indicating a successful measurement), the system then offers the user the chance to proceed to the next state providing HRV analysis.

The HRV values then are displayed continuously in the middle section of the screen, and with another press of the rotary encoder the system attempts to connect to the WLAN and sends the PPI data to a server. At this state after the user opts to send the values over to the server, the values then are published to the MQTT broker. A message is displayed at this point indicating the state of the data sent to the MQTT (data sent successfully, or unable to connect).

The screen display persists until the rotary encoder is pressed, which resets the variables and clears any stored data, returning the system to its initial state ready to start a new measurement upon the next button press.

Below is an illustration of the interface the user is prompted at different states of the program.

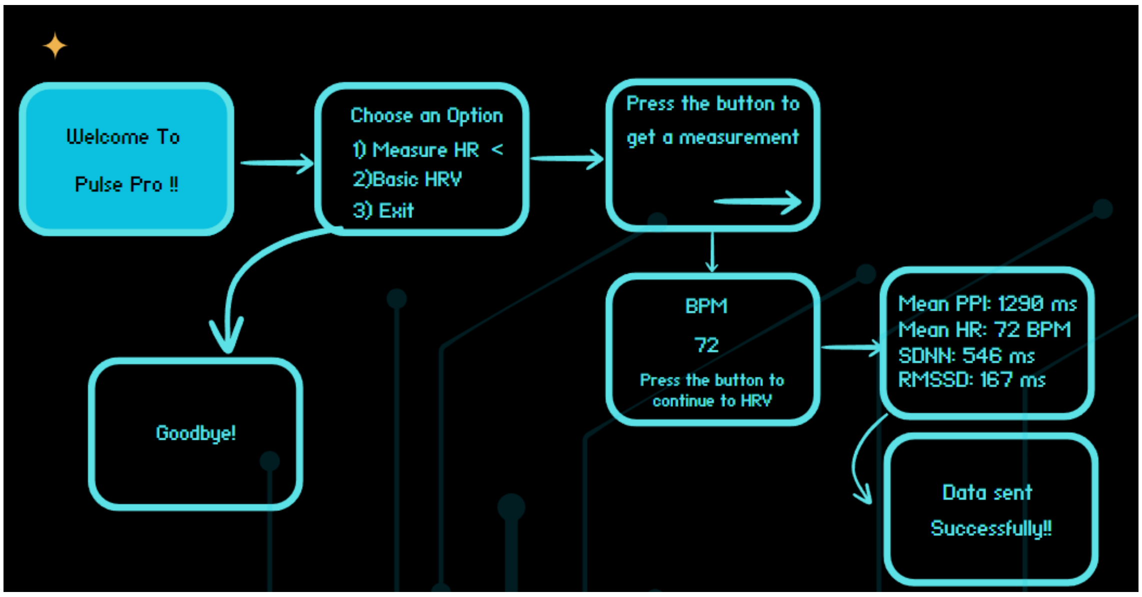


Figure 10: Illustration of how the application moves through each stage.

5 Group Work Summary

For this section, the group members of this project will contribute directly by writing summaries outlining their role in the project. Two summaries will be written:

- Mid-Project Summary: Focuses on tasks we've completed thus far, challenges we've faced, and our planned contributions for the next phase.
- Final Summary: A comprehensive overview of our work, including achievements, lessons learned, and how our contributions impacted the overall project success.

5.1 Midway Summary

The team divided the assignments among its members, with all of the members completing equal number of assignments required, which were then evaluated by their lecturer, Joseph Hotchkiss. Ammar and Egzon focused more on project coding while Jesper mostly worked on the assignments and the project report. The team has had difficulties implementing “History” and “Kubios” parts to the project, but the team is still working on implementing them. The team plans to complete the HR Monitor Device by the end of the deadline, and to present it to their class.

5.2 Final Summary

The team has successfully finished the project with a few limitations, such as the “History” and “Kubios” parts, however, the device still managed to provide consistent and accurate bpm readings and HRV analysis. In the end of the presentation the team received positive feedback from the lecturers Joseph Hotchkiss and Ulla Paatola.

The team had a great collaboration, maintained a good workflow with great task coordinating throughout the course. Meetings in the campus were arranged and responsibilities were allocated to each group member. The team will continue to finish the project report before the end of the deadline.

6 Conclusions

In the end, we have developed a device that can provide consistent and accurate BPM readings, as well as calculating and displaying analysis which then can be sent through MQTT. As stated previously, there were a few limitations such as “History” and “Kubios”, however, the device still managed to provide consistent measurements. In the future, time management could be improved and future implementations could be implemented, such as improving the UI, different themes and other visuals.

Overall, the team is happy with the results, despite all the problems the team has encountered, the team has successfully finished the project.

References

1. Raspberry Pi Pico W [Internet]. Components101. Available from: https://components101.com/development-boards/raspberry-pi-pico-w
Accessed: 08/05/2024

2. Rufus C. Using SSD1306 OLED display on ESP32 (+ Bonus Project). Medium [Internet]. 2023 Mar 14; Available from: https://medium.com/@ceavinrufus/using-ssd1306-oled-display-on-esp32-bonus-project-b9157dc0d06d
Accessed: 08/05/2024

3. Raspberry Pi Pico Rotary Encoder [Internet]. How to Electronics. 2022. Available from: https://how2electronics.com/electronic-hourglass-with-raspberry-pi-pico-rotary-encoder/
Accessed: 08/05/2024

4. Crowtail- Pulse Sensor 2.0 [Internet]. Elecrow.Openhardware. 2024. Available from:https://www.elecrow.com/crowtail-pulse-sensor-p-1673.html
Accessed: 08/05/2024

5. Wikipedia contributors. Thonny [Internet]. Wikipedia. 2024. Available from: https://en.wikipedia.org/wiki/Thonny
Accessed: 08/05/2024

6. Wikipedia contributors. File:MicroPython-logo.svg - Wikipedia [Internet]. 2014. Available from: https://en.wikipedia.org/wiki/File:Micropython-logo.svg
Accessed: 08/05/2024

7. Using photoplethysmography (PPG) for designing optical heart rate sensors [Internet]. Ansys. 2023. Available from: https://www.ansys.com/blog/modeling-human-skin-and-optical-heart-rate-sensors
Accessed: 08/05/2024