Imperial College London

MEng Final Year Project

Department of Electrical and Electronic Engineering

Real-Time Embedded Signal Processing For Building Medical Wearables Of The Future

Author

Khayle Torres
CID: 01753211
kt1719@ic.ac.uk

Supervisor

Prof. Pantelis Georgiou

Second Marker

Prof. Christos Papavassiliou

Co-Supervisors

Stefan Karolcik
Yuting Xu

June 21, 2023

Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

I have used ChatGPT v3.5 as an aid in the preparation of my report. I have used it to improve the quality of my English throughout, however all technical content and references comes from my original text.

Abstract

The advancement of medical technology holds immense significance for the global population, particularly in less economically developed countries facing the economic strain caused by diseases like malaria, dengue, and even the common cold. Inadequate treatment access leads to higher fatality rates, as evident in the disproportionate impact of COVID-19 on developing countries. This project aims to tackle this issue by developing a cost-effective medical wearable device known as D-SCAPE-v2, capable of performing non-invasive biosensing.

This report primarily focuses on the implementation of the embedded system and signal processing pipeline stages, which are crucial aspects of developing the D-SCAPE-v2. It provides a comprehensive account of the embedded system's integration, explores various methodologies for the signal processing pipeline, and presents optimization techniques employed to meet the wearable device's requirements.

A comparative evaluation will demonstrate the better improvements of the D-SCAPE-v2 in comparison to its v1 counterpart. Additionally, the report discusses further developments and optimizations that can be made to enhance the effectiveness, efficiency, and robustness of the wearable device.

The D-SCAPE-v2 has the potential of monitoring vital health parameters, contributing to improved patient assessments and personalized interventions. By pushing the boundaries of medical wearables, this project aims to contribute to improved healthcare outcomes and bridge the gap in healthcare disparities on a global scale.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Georgiou Pantelis, Stefan Karolcik and Yuting Xu. Their constant encouragement and insightful guidance made working on this project an amazing experience. I am truly grateful for the opportunity they have given me and to have them as my supervisors.

I am thankful to my friends both at Imperial and high school who have been a constant source of motivation and have enriched my student life.

And lastly, I would like to give my warmest appreciation to my parents who have provided nothing but love, support and continuous sacrifice in order to get to where I am today.

Contents

1 Introduction **10**
- 1.1 Motivation . . . 10
- 1.2 Objectives and Challenges . . . 11
- 1.3 Contributions . . . 11
- 1.4 Ethical/Legal Concerns . . . 12

2 Background **13**
- 2.1 Multi-site, Multi-wavelength PPG platform . . . 13
  - 2.1.1 Photoplethysmography . . . 13
  - 2.1.2 Analogue Front-Ends . . . 16
  - 2.1.3 D-SCAPE v1 . . . 16
  - 2.1.4 D-SCAPE v2 . . . 18
- 2.2 Communication Protocols . . . 19
  - 2.2.1 OSI Model . . . 19
  - 2.2.2 Communication in Embedded Systems . . . 20
  - 2.2.3 SPI . . . 20
  - 2.2.4 I2C . . . 21
- 2.3 Embedded Systems . . . 21
  - 2.3.1 Microcontrollers . . . 21
  - 2.3.2 Embedded Application . . . 22
  - 2.3.3 Interrupts . . . 22
- 2.4 Signal Processing . . . 22
  - 2.4.1 Digital Filtering . . . 23
  - 2.4.2 Feature extraction . . . 24
  - 2.4.3 Spectral Analysis . . . 24

3 Related Work **25**
- 3.1 Literature on PPG signals . . . 25
  - 3.1.1 Noninvasive measurement using medical wearables . . . 25
  - 3.1.2 Similar Products . . . 26
- 3.2 Estimating Arterial Blood Pressure using PPG . . . 26
  - 3.2.1 Single Site PPG Measurement . . . 27
  - 3.2.2 PPG + ECG . . . 28
  - 3.2.3 Multi-Site PPG . . . 28

4 AFE4900 Embedded Implementation **29**
- 4.1 Components . . . 29
  - 4.1.1 AFE4900 . . . 29
  - 4.1.2 Espressif ESP32-S3 . . . 29
  - 4.1.3 General Purpose Timers . . . 30
- 4.2 Getting Started with Embedded Applications . . . 31
  - 4.2.1 AFE4900 Library . . . 31
  - 4.2.2 AFE4900 Configuration . . . 31

4.2.3 Verifying configuration . . . 33

5 D-SCAPE-v1 Delay Problem 35

5.1 Quantifying the delay . . . 36
5.2 Implementation . . . 36
5.2.1 Iteration 1 . . . 37
5.2.2 Iteration 2 . . . 38
5.2.3 Iteration 3 - Final . . . 40
5.2.4 Verification . . . 41
5.3 Performance Issues . . . 45
5.4 Optimizations . . . 47
5.4.1 Improving Overall Speed . . . 47
5.4.2 Targetted Optimization . . . 48
5.4.3 Multiprocessing and Task Priorities . . . 48
5.4.4 Logging Optimisations . . . 49
5.4.5 Code Profiling . . . 50
5.4.6 Evaluating The New Sampling Method . . . 52
5.5 Wrist Sensor & Final Results . . . 53

6 Signal Processing 56

6.1 Peak Detection Algorithm . . . 56
6.2 Signal Processing within the Microcontroller . . . 59
6.2.1 C Code vs Python Implementations . . . 59
6.2.2 Performance Drawbacks . . . 63
6.2.3 Evaluation and Further Improvements . . . 64

7 Additional Development and Implementation 65

7.1 Temperature Sensing . . . 65
7.1.1 TMP117 Library . . . 65
7.1.2 Testing . . . 65
7.1.3 Integration with ADXL343 . . . 66
7.2 Integration with System Design . . . 66
7.2.1 Brief Integration Evaluation . . . 67

8 Conclusion 69

8.1 Further Development . . . 69
8.2 Final Remarks . . . 70

A ISCAS 2023 75

A.1 Youtube Video . . . 75
A.2 Conference Report . . . 75

B Derivations 81

B.1 Band-pass Butterworth Filter Derivation . . . 81
B.2 Band-pass Chebyshev Type I Filter Derivation . . . 83

C Sample Code Listings **86**
C.1 Project Github Page 86
C.2 AFE4900 86
C.3 TMP177 87

D Log Output Results **88**
D.1 128KHz Configuration Example 88
D.2 D-SCAPE-v1 Sample Reading 88
D.3 TMP117 Sample Log 89
D.4 TMP117 & ADXL343 Sample Log 89

List of Figures

1 Embedded Systems Diagram at high-level . . . 12
2 Transmissive PPG Sensor . . . 14
3 Reflective PPG Sensor . . . 14
4 PPG waveform shape and features. T$_S$ represents the time between two systolic peaks. T$_{SD}$ represents the time between systolic peak and diastolic peak . . . 14
5 D-SCAPE v1.0 . . . 16
6 D-SCAPE v2.0 . . . 19
7 OSI Model . . . 19
8 AFE4900 128KHz sample configuration vizualisation . . . 32
9 Operating Frequency Estimation Using Oscilloscope (ADC_RDY pin) . . . 34
10 Sine wave plots . . . 35
11 Absolute difference over time of sine waves . . . 35
12 Iteration 1 flowchart diagram for each AFE . . . 37
13 Iteration 2 flowchart diagram for each AFE . . . 39
14 Polling Signal Visualisation . . . 40
15 Iteration 3 flowchart diagram for each AFE . . . 40
16 Raw, Unfiltered, Inverted PPG signal Plot . . . 42
17 Band-pass Filtered Signal with parameters (N = 4, rs = 50, lp_cutoff = 10, hp_cutoff = 0.35, fs = 100) . . . 43
18 Plot of PPG signals of index fingers with parameters (N = 4, rs = 50, lp_cutoff = 10, hp_cutoff = 0.35, fs = 100) . . . 44
19 Zoomed in version of PPG signals . . . 44
20 Zoomed in version of PPG Signal peak . . . 45
21 Filtering of PPG signals with different sampling frequency parameters (Original fs = 100) . . . 46
22 Number of LEDs To Read and Write Time Plot . . . 50
23 Number of LEDs To Maximum Potential Operating Frequency . . . 51
24 Number of LEDs To Maximum Potential Operating Frequency . . . 52
25 Finger + Finger Configuration Output (Zoomed out) . . . 53
26 Finger + Finger Configuration Output (Zoomed in) . . . 54
27 Finger + Wrist Configuration Output (Zoomed in) . . . 55
28 Initial Peak to Peak detection algorithm results (h = 1, window = 2s, fs = 100Hz) 57
29 Peak to Peak detection algorithm results zoomed (h = 1, window = 2s, fs = 100Hz) . . . 57
30 Finalised peak detection flowchart . . . 58
31 Modified peak detection algorithm results zoomed . . . 58
32 Modified peak detection algorithm results zoomed . . . 59
33 C Code Chebyshev type I filtering (red dots - C Code version peak detection, Blue dots - Python version peak detection) . . . 60
34 Python version Chebyshev type II sample comparison . . . 60
35 Zoomed in C Code Chebyshev type I filtering (red dots - C Code version peak-to-peak detection, Blue dots - Python version peak-to-peak detection) . . . 61

36 Example visualisation of different bandpass filters with cutoff frequency 0.25-0.75Hz (Parameters: Cheby1 - [n=5, rs = 1], Cheby2 - [n=5, rs = 15], Butterworth - [n=5] 62
37 Butterworth Filtered signal with original parameters (n=5, highpass cutoff = 0.35, lowpass cutoff = 10) 62
38 Butterworth Filtered signal (n=5, highpass cutoff = 0.5, lowpass cutoff = 6) . . 63
39 Integrattion Web App (1) 67
40 Integrattion Web App (2) 68
41 Chebyshev Polynomials up to degree n = 4 83

List of Tables

1 Table Of Time Delay Between Consecutive ISR_RDY Signals 33
2 Table of the difference between consecutive reads between the two AFE chips . . 36
3 Table Of Consecutive Time Delay And Quantity (N = 10,000, Operating Hz = 100Hz) 42
4 Table Of Estimated Heart Rate Readings From Both PPG Data & Apple Watch 43
5 Table of samples missed and its corresponding effect on the operating frequency (Original fs = 1000) 47
6 Table of Read and Write Times given number of LEDs 50
7 Estimated Read & Write Gradients 51
8 Table of the difference between consecutive reads and quantity (213,304 samples) 53
9 Table of Time Taken to Perform Each Task (s) Between Original and C-Code Filtering Implementations (10000 samples) 64

1 Introduction

1.1 Motivation

One of the most prevalent seasonal infection diseases, dengue fever, affects 390-400 million people annually [1]. It is an infectious disease spread by mosquitoes in mainly tropical countries [2] and imposes a major strain on the healthcare system of countries affected [3]. Due to this, a team at the Centre for Bio-inspired Technology at Imperial has developed a low-cost wearable device that would monitor patients during outbreaks. The device named Dengue Shock Classification And Prediction wEarable (D-SCAPE) would be used to assess the probability of complications in real-time whilst also being cost-efficient to help decrease the impact of dengue in the healthcare system.

The system is based on [4] and uses a state-of-the-art, light-based vital sign monitoring technique called photoplethysmography (PPG). This involves data acquisition from multiple LED-photodiode pairs in parallel to get an accurate diagnosis for the patient. The general process is to use LEDs and photodiodes to measure different characteristics of the patient. Examples can include Haematocrit levels, Saturated Oxygen Levels and Heart Rate, to name a few.

Following the successful clinical trials in Vietnam, the D-SCAPE can now take the next steps to improve its functionality, allowing for a more robust and accurate patient assessment. One main issue with the original D-SCAPE is that the two chips used to measure the PPG signals (AFE4900) have two independent internal oscillators. Although both chips can be configured to operate at 1000Hz, there will inherently be a delay between the two measurement readings. Each chip also has a manufacturing tolerance, meaning one chip could be at 999Hz and another at 1002Hz. Although this does not appear significant initially, this linear time-varying delay poses a big issue when correlating the two PPG waveforms together, especially over a long period, such as a day or week. This has also been shown to affect the pulse wave velocity (PWV) estimation readings used to measure patient blood pressure. On top of this, the current D-SCAPE uses the GUI provided by Texas Instruments for its AFE4900 evaluation board, thus making functionality expansion a difficult task.

Therefore, this work focuses on the newest version of the D-SCAPE, the D-SCAPE-v2, which aims to match and even exceed the original performance of its v1 predecessor. It also aims to quantify the delay problem the independent internal clocks of each AFE chip posed and investigate some potential signal processing improvements. Additionally, the DSCAPE-v2 will allow portability improvements to the device, giving the wearable the functionality to operate on battery power. Lastly, this project will serve as a framework to be built on for additional functionalities or optimisations for future iterations of the D-SCAPE wearable.

Although Dengue fever is the specific platform implementation of this work, there is an emphasis that this wearable could be adapted for monitoring other diseases.

1.2 Objectives and Challenges

The project aims to implement a new embedded application to perform multi-wave PPG data acquisition for the D-SCAPE-v2 while quantifying the delays of the two AFE chips. On top of this, the project also aims to deliver a software framework that can be used as a reference or built on top of future iterations of the D-SCAPE and similar medical wearables. Specifically, the overall research objectives of this work are:

1. **Implementation**: Creating an embedded application that has the following characteristics:
   - Perform high-precision data acquisition between 2 analogue front-ends (AFE4900) and a single microcontroller (ESP32), reading up to 4 PPG signals from each AFE. This means being able to read 8 different PPG signals and perform time-critical synchronisation (Figure 1).
   - Have a method to quantify the delay between the two analogue front-ends.
   - Be able to perform low latency reads and writes to registers directly on both AFE chips.
   - Have the capabilities to be built on top of, potentially adding more sensor devices if needed.
   - Be able to integrate some form of signal processing and sanity check biomarker readings to verify that the v2 performs similarly to the v1.
2. **Evaluation**: Be able to properly evaluate the performance of the embedded application, such as:
   - Measuring the delay between PPG signals and the average delay between each signal read
   - Measuring the difference between consecutive updates between the two analogue front-ends.
   - Have a method for evaluating system performance to ensure the minimum performance requirements are met.

Performing high-precision, low-latency, multi-wavelength PPG signal acquisition poses multiple challenges. This requires a solution to be developed and proposing ways of evaluating and measuring the improvement of the proposed solution.

1.3 Contributions

The contributions of this project are summarized as follows:

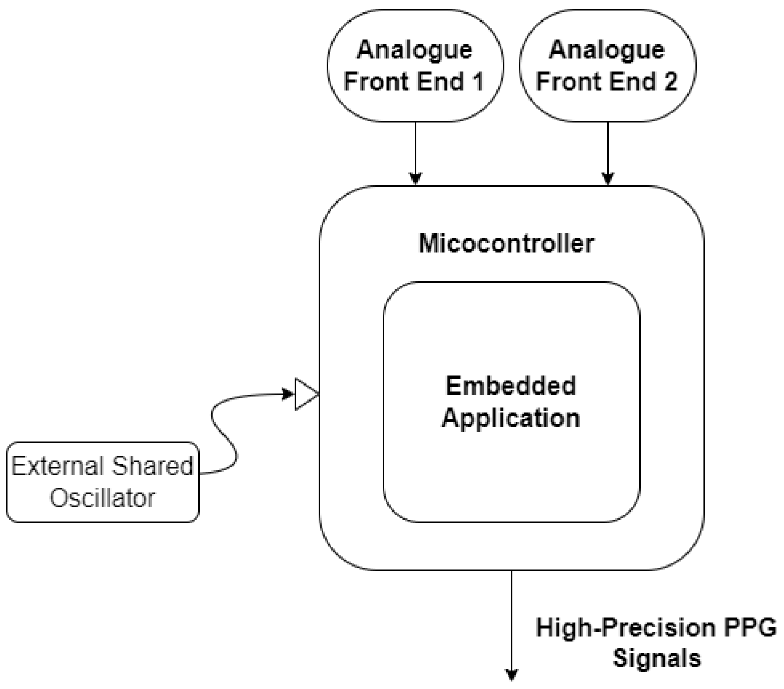


Figure 1: Embedded Systems Diagram at high-level

1. Develop an embedded application using the PlatformIO IDE Interface for data acquisition between an Espressif ESP32 microcontroller 2 AFE4900 chips. This embedded application will be written using the PlatformIO Espressif 32 platform.

2. Develop methods to minimise and quantify the delay between the PPG signals of both AFE4900 chips.

3. Provide a framework that can be used for future D-SCAPE iterations and potentially similar medical wearables.

4. Provide the mathematical models and algorithms for some signal processing techniques and code implementations (either in C code within the microcontroller or a Python script running on a machine attached to the microcontroller).

5. Integration with separate work done for the D-SCAPE-v2 system design. This is to get it closer to the production phase.

6. Additional Temperature Sensor library (TMP117) for potential future expansion.

1.4 Ethical/Legal Concerns

As the full datasheet for the AFE4900 is under selective disclosure by Texas Instruments, this will be mindfully excluded, and specific technical detail from this datasheet will not be included in the thesis. As my project excludes the involvement of any live patients when developing, there are no major legal concerns needed to be considered.

2 Background

Since this project is built on top of the D-SCAPE work, some background must be explained. This chapter will introduce the photoplethysmography technique and briefly explain how it has been developed for disease monitoring and assessment. On top of this, there will be some explanation of relevant communication protocols and their uses alongside different sensors used. This chapter aims to give more context to the requirements specified previously and why it is important to meet these.

2.1 Multi-site, Multi-wavelength PPG platform

2.1.1 Photoplethysmography

Photoplethysmography (PPG) is an uncomplicated and inexpensive optical measurement method that is increasingly used for measuring an individual's physiological state in day-to-day life [5]. Photoplethysmogram is obtained by illuminating the skin with one or more LEDs at different wavelengths. As light travels through the tissue, multiple absorption and scattering events occur [4]. A photodiode is then used to measure the intensity of the non-absorbed light reflected or transmitted from the skin [6]. The received signal from the photodiode would normally be connected to a trans-impedance amplifier, followed by a gain stage, an analogue-to-digital converter and finally an optional noise filtering stage [4].

Different Types of PPG Sensor Configurations

The operating modes of PPG sensors consist of two types, reflection and transmission. In both modes, 1 photodiode sensor and 1 led/infrared light source are still present.

- *Transmission:*

  In transmission mode, the led/infra-red light source and photodiode sensor are on opposite sides on the tissue measurement site (Figure 2). This measures the transmission of the residual light from the led/infra-red source after its absorption by the tissue [7]. Examples of common transmission mode PPG sensors are ear clip (EC), finger ring (FR), or finger clip (FC).

- *Reflection:*

  In reflection mode, the led/infra-red-light source and photodiode sensor are on the same side on the measurement site (Figure 3). This intensity of the reflected light reflected by the body surface is measured using the photosensor. This allows reflective PPG sensors to offer higher flexibility on different body measurement sites [7].

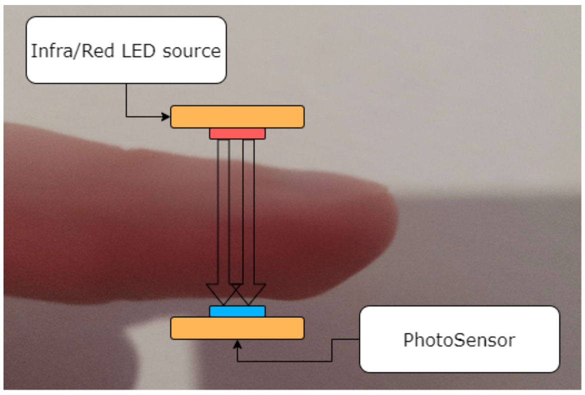


Figure 2: Transmissive PPG Sensor

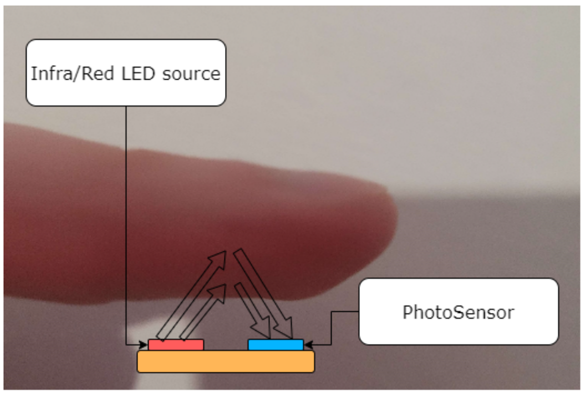


Figure 3: Reflective PPG Sensor

Interpreting PPG Signals

PPG signals take a similar shape to the one shown in Figure 4, with each pulse commonly divided into two phases. The first phase, also known as the anacrotic phase, is the region in the pulse that consists of the rising edge of the pulse. The second phase (catacrotic phase) is the region in the pulse that consists of the falling edge of the pulse. The first phase primarily concerns the systole, whereas the second involves the diastole. [8].

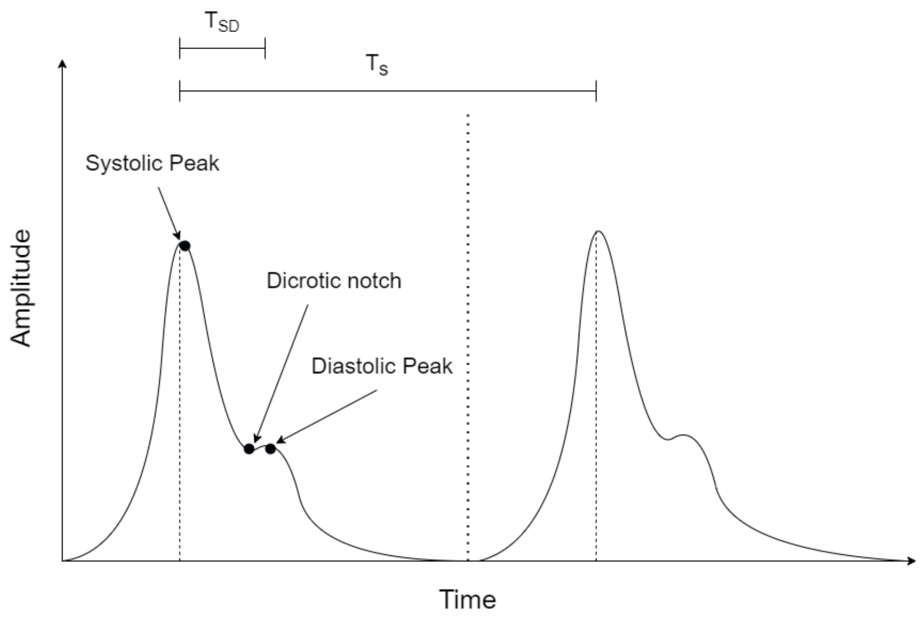


Figure 4: PPG waveform shape and features. T$_S$ represents the time between two systolic peaks. T$_{SD}$ represents the time between systolic peak and diastolic peak

PPG signals contain a number of features that can be extracted, which contain useful biomarkers. Some of these biomarkers come from the base signal, and others from its first or second derivative. Some of these include [8]:

- Systolic Amplitude
- Pulse Width
- Pulse Area

- Peak to Peak Interval
- Many more...

Additionally, PPG measurement is a non-invasive, pain-free method, making it a very useful tool in biosensing applications.

Challenges Working with PPG Sensors

One common challenge when working with PPG sensors is signal artefacts. A signal artefact is usually an error that is not related to the signal activity being measured. This is due to multiple reasons, such as external forces, equipment used, etc. In this project, we will talk about some common artefacts with regard to PPG sensing.

- Ambient Light Interference:

  Ambient Light Interference is usually an artefact most commonly caused by ambient electromagnetic signals present. This ambient light source can "leak" into the sampled signal introducing artefacts.

- Power-line interference A high-frequency artefact caused by mains power sources (Power-line interference) could also be induced onto the PPG signal recording [8]. This would present itself as a sinusoidal component in the signal recording and as a spike in the frequency power spectrum of the signal [8].

- Motion Artifact:

  Motion artefacts are errors commonly caused by the movement of patients during PPG data acquisition. It can also be caused by poor contact with the fingertip photosensor [8]. A way to potentially detect these motion artefacts could be to employ an accelerometer for measuring the body movement at the same time of data acquisition [9].

PPG Sensing Applications

PPG sensing is widely used in numerous bio-monitoring technologies, such as smartwatches and fitness sensors. It has also grown in consumer and research interest over the years [10][11].

Conventional PPG systems are typically built and optimized for either heart rate or pulse oximetry measurement. PPG, however, can also be used to extract other important biological information from an individual. One such example is a PPG signal's second derivative wave which contains important health-related information which can assist in the early detection and diagnosis of various cardiovascular illnesses that may appear later in life [6].

2.1.2 Analogue Front-Ends

An analogue front-end is a series of analogue signal-conditioning circuitry which interfaces with analogue signals and processes this as a part of a digital system.

Data acquisition from multiple PPG signals is required to track the biological markers to monitor dengue fever. This would mean building a full receiver pipeline for each PPG waveform. Instead, an analogue front-end chip developed by Texas Instruments (AFE4900) is leveraged to capture multiple PPG signals concurrently. The AFE4900 chip is an analogue front-end with the purpose of synchronized electrocardiogram (ECG) and photoplethysmogram (PPG) signal acquisition [12]. This design decision allows multiple PPG waveforms, all with different wavelengths, to be captured simultaneously. The AFE4900 chip has a PPG receiver that supports up to three time-multiplexed photodiode inputs and a PPG transmitter that supports up to 4 independent LED channels. In this project, 2 AFE4900 chips are used and connected to a single microcontroller.

2.1.3 D-SCAPE v1

This project, in particular, builds on the D-SCAPE medical wearable, which focuses primarily on dengue fever disease. The D-SCAPE is a medical wearable developed at the Centre for Bio-inspired Technology in Imperial and is composed of three components, shown in Figure 5.

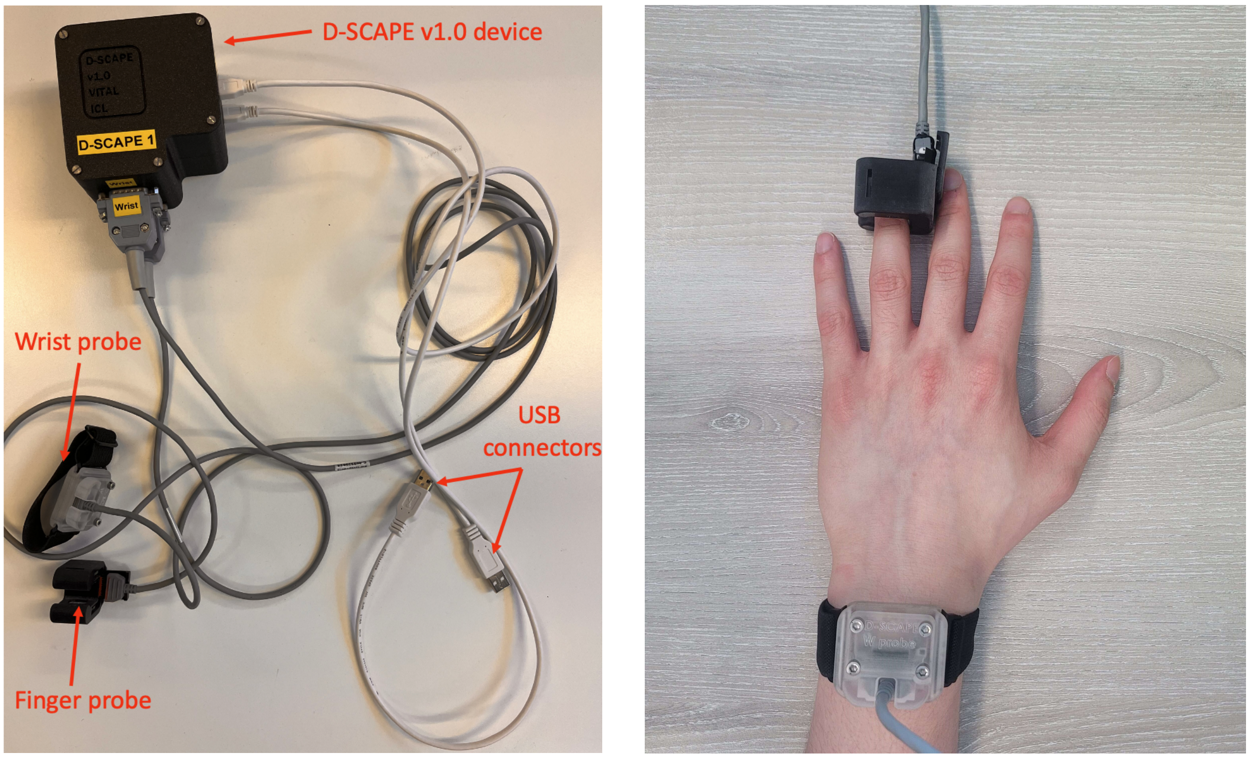


Figure 5: D-SCAPE v1.0

As dengue fever symptoms worsen, patients are more susceptible to dengue shock, which manifests as plasma leakage. This plasma leakage has a direct effect on increasing the patient's haematocrit levels as well as decreasing their blood pressure in a short period of time. Due to this, PPG signals can be used to measure the patient's haematocrit level and blood pressure over time to analyse an infected patient compared to a healthy one. This allows a way to track and monitor the progression of dengue over time [4]. This way, the requirement for accuracy of absolute vital sign estimation can be relaxed as there are more markers to be used to analyse the disease.

The method used for the D-SCAPE for measuring blood pressure is in [13] and uses two measurement sites with 2 AFE4900 chips. The two AFE4900 chips would be set to PPG measuring mode and put on the wrist and finger. The PPG signals would then calculate the patient's pulse wave velocity (PWV). Pulse wave velocity (c) can be estimated using transit distance ($\triangle x$) and transit time ($\triangle t$) as shown in (1).

$$c = \triangle x / \triangle t \tag{1}$$

Using the Moens-Kortweg equation [14] in (2), we can write the pulse wave velocity (c) as a function of the arterial vessel elasticity (E).

$$c = \sqrt{\frac{Eh}{2R\rho}} \tag{2}$$

The vessel wall elasticity can, in turn, be written as a function of pressure [14] as shown in (3).

$$E = E_0 e^{\zeta P} \tag{3}$$

By combining the two equations, this, in turn, would be used to estimate the patient's blood pressure using the relationship between pulse wave velocity and arterial blood pressure (4).

$$P = \frac{1}{\zeta} ln(\frac{2R\rho c^2}{hE_0}) \tag{4}$$

By further simplifying constants, we finally obtain the equation in (4) where (P) is the blood pressure and ($k_{1,2}$) are constants to be obtained when calibrating.

$$P = k_1 ln(c^2) + k_2 \tag{5}$$

Assuming the distance of leading and lagging photodiodes similar to [13] is the average length of an adult male (7.44 inches or 18.90cm) hand, then by taking an average PWV range of $6m/s$ shown in [15] we can calculate the estimated number of seconds between the two peaks between the wrist and finger probes for the D-SCAPE wearable (6).

$$PWV = Distance/Time \tag{6}$$

$$0.189/6 = 0.0315s \approx 31 \text{ samples at 1000Hz} \tag{7}$$

Additionally, one of the characteristic features of severe dengue fever is plasma leakage. Plasma leakage has a direct effect on increasing haematocrit and decreasing blood pressure in a short period of time. This change in blood pressure will typically lead to small changes in PWV (e.g., 6m/s to 6.5m/s would be 2-3 samples). We can see when using this form of blood pressure monitoring, it is crucial for the PPG signals to be as close as possible.

However, the problem with the original D-SCAPE is that the two AFE4900 chips used within the original D-SCAPE-v1 have independent internal oscillators. This incurs a slight difference in operating frequency (with the same configuration) due to manufacturing tolerance. This has been measured to be a linear time-varying delay which gradually increases over time. Due to the very high precision required to measure differences in blood pressure, it is important to quantify and minimise delay as much as possible, as this can significantly decrease the reliability of the measurement.

It is important to note that this is just an estimated example, demonstrating the importance that the recorded PPG signals must have high precision and low delay. This might be slightly inaccurate as the equation uses the average length of the hand instead of arterial length, and PWV between different age groups vary.

2.1.4 D-SCAPE v2

To mitigate the main problem posed by the original D-SCAPE-v1, the new v2 version is created. This new system uses a shared external oscillator which both AFEs use to eliminate the manufacturing tolerance differences between each AFE chip. As they will both use the same shared oscillator, this should, in theory, allow each AFE chip to have a constant delay, turning the linear time-varying delay problem into a constant one. This should simplify the problem significantly since the delay of each AFE chip does not increase on every iteration.

This is done by removing two AFE4900 evaluation boards and using the AFE4900 chips directly with an ESP32-S3 microcontroller. Using a microcontroller along with the AFE chips should significantly increase the customizability of the D-SCAPE device, allowing additional features the original v1 would not have the capabilities to implement.

When mass-produced, this new v2 version is cheaper than the original v1 version, costing around $100 per board with improved performance. A diagram in Figure 6 shows the new board depicting all its components and appearing a significantly more compact version than the v1. As the E in D-SCAPE stands for wEarable, this increase in portability allows the D-SCAPE to take a closer step into becoming a finished product.

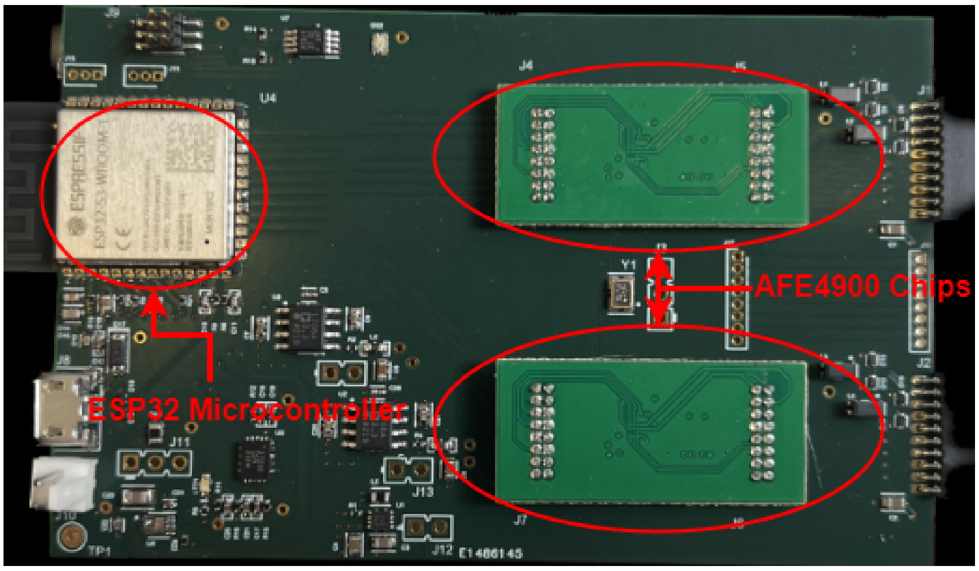


Figure 6: D-SCAPE v2.0

2.2 Communication Protocols

Communication Protocols are methods which allow the transmission of information between devices. It is widely used in almost all technology in day-to-day life, and different communication protocols can be classified into different layers [16].

2.2.1 OSI Model

The Open Systems Interconnection (OSI) Model depicts the different communication layers moving from low-level to higher-level communication, as shown in Figure 7.

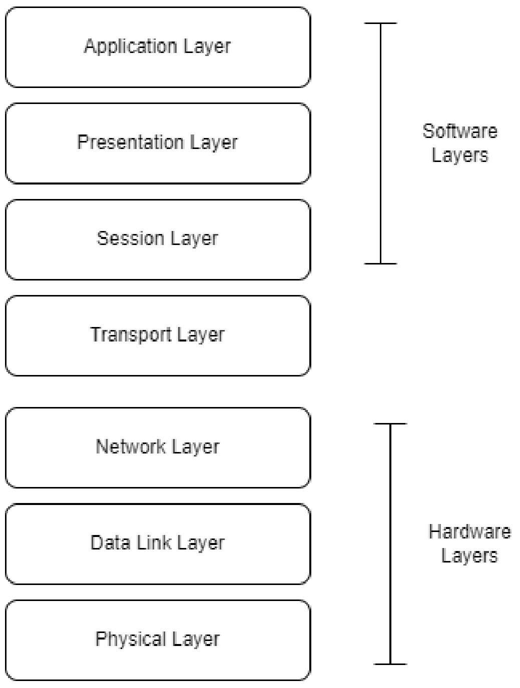


Figure 7: OSI Model

The project works primarily with PPG signals collected by the AFE49000 chips sent to a microcontroller. This means we will primarily focus on the Data Link Layer protocols (Layer 2).

2.2.2 Communication in Embedded Systems

Different serial data link layer protocols can be used when establishing a communication link in an integrated circuit. This section aims to introduce the two leading options that can be used in the project: SPI and I2C.

2.2.3 SPI

Serial Peripheral Interface (SPI) is a communication protocol that provides full-duplex (data transmission can be in both directions simultaneously) synchronous communication between two devices [17]. It uses a master-slave structure which provides a very fast, simple, low-cost interface between a microcontroller and the device connected to it. A conventional configuration that uses the SPI interface typically has 4 wire signals used for communication:

1. SCLK (Serial Clock Signal)

   SCLK, or Serial Clock Signal, is a clock output from the primary to the secondary device. Since SPI is a synchronous communication protocol, a shared clock signal between the primary and secondary devices must be shared. This signal allows the primary device to specify the frequency at which the two devices communicate and when the data bits will be transmitted.

2. MOSI (Master Output Slave Input)

   The MOSI signal is a data line that transmits information from the primary to the secondary device. This means the primary device writes to the MOSI pin, and the secondary device reads from it at every Serial Clock cycle.

3. MISO (Master Input Slave Output)

   The MISO signal, similar to the MOSI signal, is a data line that transmits information from the secondary device to the primary device. The MOSI and MISO data lines allow the full-duplex synchronous communication of the SPI interface.

4. SS/CS (Slave Select/Chip Select)

   The Slave Select/Chip Select line specifies which secondary device the primary device is communicating with. This allows a single communication bus to support multiple secondary devices.

2.2.4 I2C

I2C, similar to SPI, is a serial communication protocol that provides a half-duplex (data transmission is only one direction at a given time) synchronous communication between two devices [18]. I2C and SPI share a similar feature: they can connect multiple secondary devices to a single primary device. I2C, on top of that, also can have multiple primary devices controlling single or multiple secondary devices (multi-master protocol) [18]. A conventional configuration that uses the I2C interface typically has 2 wire signals used for communication:

1. SDA (Serial Data)

   The SDA signal is a data line for the primary and secondary devices to send and receive data. It is also the line used for addressing the case of multiple secondary devices or specifying whether the primary device is performing a read/write operation.

2. SCL (Serial Clock)

   SCL, or Serial Clock Signal, is a clock output from the primary to the secondary device. Similar to the serial clock of SPI, the SCL signal allows the primary device to specify the frequency at which the two devices communicate and when the data bits will be transmitted.

2.3 Embedded Systems

One or more hardware devices must be used to analyse a patient's biomarkers. Some of these hardware devices measure data in the form of analogue signals. This signal is processed, potentially filtered and then sent to a microcontroller unit (MCU). This is what typically consists of an embedded system.

An embedded system is a system that consists of multiple components to perform a specific function. A typical conventional system consists of multiple hardware devices connected to one or more microcontroller units. Examples of such hardware devices could include a 7-segment display, a camera, an accelerometer, etc. Building an embedded application requires knowledge of the structure of the embedded system. It is also important to understand the intended purpose of each device for that specific embedded system to meet project requirements.

2.3.1 Microcontrollers

A microcontroller is a small microcomputer designed to perform specific tasks in an embedded system. Microcontrollers typically consist of a CPU, RAM, ROM and I/O ports [19]. Instead of managing each hardware device individually, a microcontroller allows this process to be done simultaneously.

Typically, a serial bus protocol (e.g., USB, UART) is used with a computer to program some software into the microcontroller unit (MCU). This software will determine how the MCU will interact with all the devices within the embedded system. It is the central communication device between all devices, making it an integral part of the data processing pipeline.

Each microcontroller will have its own peripherals allowing different functionalities as it is generally tailored to specific applications. For this project, the microcontroller used is the Espressif ESP32-S3. The features and peripherals of interest with regards to the ESP32 will be discussed in more detail in Section 4

2.3.2 Embedded Application

An embedded application is software programmed on a device for a specific function; in embedded systems, this typically means the Microcontroller Unit. Each embedded application is programmed to meet the objectives of each project, utilising the different devices in the embedded system. Meeting project specifications will require developing an embedded application utilising multiple different peripherals and features of the ESP32 and numerous software libraries.

2.3.3 Interrupts

Embedded system interrupts are used in microcontrollers and other embedded systems to handle asynchronous events. This interrupt is a signal generated by either a hardware device pin or software condition that halts the normal execution flow of a program to respond accordingly.

When an interrupt occurs, the processor halts the current execution of the program and performs the callback function for that specific interrupt handler or interrupt service routine (ISR). After the callback function has been completed, the program resumes where it had left off. Interrupts provide a very efficient way to handle time-critical events without continuous polling. In the context of medical wearables, some interrupts can be used to either detect if the patient temperature is too low, signal when the data is ready to sample, perform an action every x number of seconds, etc...

2.4 Signal Processing

Signal processing involves analyzing, modifying and extracting information from signals. The main aim of signal processing is to be able to manipulate the signal to extract or enhance meaningful/useful information from it.

Signals are either analogue or digital and can come from multiple sources, such as a temperature sensor, accelerometer, photodiode detector, etc. Signal processing is a crucial step in the signal processing pipeline since most of the signals would be too noisy and unusable without it.

Signal processing techniques can be categorized primarily into two distinct domains:

1. Analog Signal Processing: Primarily involves manipulating continuous signals.
2. Digital Signal Processing (DSP): Primarily involves manipulating signals in discrete time. Some slight modifications to the maths of the original analogue version of the signal would have to be done to apply to the digital version.

As this project primarily involves digital embedded systems, the work on signal processing will mainly be done in the digital domain (DSP). Additionally, some signal processing tasks used in this work can be further classified based on their specific tasks.

2.4.1 Digital Filtering

A digital filter takes a series of inputs $x_0, x_1, ...$ and produces a series of outputs $y_0, y_1, ...$. This involves manipulating the digital signals to enhance certain characteristics (e.g., filter only to see a certain range of frequencies), remove unwanted components or selectively attenuate/pass a range of frequencies. In the specific context of the D-SCAPE-v2, the main use of the digital filter is to remove noise and DC bias to enhance the signal features. This can then be further processed down the signal-processing pipeline.

The two main types of digital filters include:

1. Finite Impulse Response (FIR) Digital Filters: Also known as a non-recursive digital filter, the current output is only affected by its inputs (not previous outputs).

   An N^{th} order digital FIR filter can be implemented as follows:

   $$y[n] = c_0 * x[n] + c_1 * x[n-1] + \cdots + c_N * x[n-N] \tag{8}$$

   where $y[n]$ represents the output at time index n, $x[n]$ represents the input signal at time index n, $c_0, c_1, ...c_N$ are the filter coefficients and N is the filter order. Typical FIR filters include the lowpass, bandpass, bandstop and highpass filters.

   As FIR filters do not utilize the feedback IIR filters do, this simplifies the implementation and avoids instability, thus allowing for stable and predictable behaviour. However, some drawbacks of FIR filters are that they often require a much higher filter order than IIR to achieve similar performance. This additional storage and/or computational work is not very desirable in an embedded system such as the D-SCAPE-v2.

2. Infinite Impulse Response (IIR) Filters: Also known as recursive filters, since its output is a function of its previous outputs and the input.

An N^{th} order digital FIR filter can be implemented as follows:

$$y[n] = c_0{*}x[n]+c_1{*}x[n-1]+\cdots+c_N{*}x[n-N]-(d1{*}y[n-1]+d2{*}y[n-2]+...+dM{*}n[n-M]) \tag{9}$$

This shows that the current n^{th} output is a linear combination of N inputs and the previous M inputs. Different filters have different filter coefficients (c and d coefficients), each giving its specific frequency response. Some common IIR filters include the Butterworth filters, Chebyshev I & II filters and many more.

IIR filters are more efficient computationally than FIR filters, as they require a lower filter order to yield. It typically also has narrow transition bands for the same filter order, which will be useful for PPG signal processing. Since the signal processing required for this project requires tight transition bands and lower computational complexity, this favours using an IIR filter over an FIR filter.

2.4.2 Feature extraction

Feature extraction is the process of taking a digital signal and extracting characteristics or components from it. This is typically used in conjunction after filtering, as mentioned above. These characteristics can encompass a wide range of values such as frequency bands, time-domain characteristics, statistical properties and many more, some of which are mentioned in the PPG section. The two main features that this work will primarily be looking at are the beats per minute estimated from the measured signal and a peak detection algorithm. The beats per minute will be used to estimate heart rate. In contrast, the peak detection algorithm will primarily be used in pulse wave velocity to estimate the patient's heart rate.

2.4.3 Spectral Analysis

Spectral analysis, in the context of signal processing, refers to a method which analysis the frequency components of a signal. This involves examining the distribution of power across the range of frequencies of the signal to get information about its spectral properties. Some common digital signal processing techniques used for spectral analysis can include:

1. Fourier Transform: Decomposes a complex signal into its constituent frequencies; this allows analysis of a time-domain signal in the frequency domain.
2. Power Spectrum: Represents the distribution of power or energy of a signal across different frequencies
3. Spectogram: A visual representation that displays the time-varying frequency content of a signal, showing how the signal's frequencies change over time.
4. Many more...

3 Related Work

The use of photoplethysmography (PPG) is found in numerous applications, and a lot of research has already been done to utilize its effectiveness. Measuring biological characteristics using PPG is also nothing new, with multiple commercial and medical applications used daily, such as pulse oximeters, smartwatches, fitness trackers, sleep trackers, and multiple others. This section will highlight the previous work done on PPG signals and explain the differences compared to the work in this report. As PPG is quite well established, this section aims to contextualize the current challenges faced when using PPG and identify the potential improvement opportunities that the D-SCAPE addresses.

3.1 Literature on PPG signals

3.1.1 Noninvasive measurement using medical wearables

PPG is a very uncomplicated and inexpensive optical measurement method with the capability of monitoring patients in a non-invasive manner. As stated previously, PPG signals numerous features which contain useful patient biomarkers. Due to this, multiple studies have been conducted to utilize these desirable characteristics to develop noninvasive measurement wearable devices.

A paper in 2018 that reviews wearable PPG sensors and their potential future applications [6] explains the significant benefits of using PPG technology. The paper showed that PPG sensing technology had been shown to provide medical practitioners with a tool that will allow early detection and diagnosis of cardiovascular diseases. PPG signals can also provide numerous advantages over traditional heart rate monitoring techniques such as ECG-based systems. One such example is that in order to measure heart rate using PPG technology, only 1 sensor will need to be placed on the body. In contrast, a traditional ECG-based system will require at least three bioelectrodes to be placed [6]. On top of this, ECG requires the placement of electrodes directly on the skin, which can be invasive and may cause discomfort, especially for long periods of time.

Wearable PPG sensors can only be placed at certain body locations [6]. The accuracy, however, of each different measurement site is not equivalent, with each measurement site being more effective than others [20]. These include:

- Forehead
- Earlobe
- Wrist/Forearm
- Fingertip

- Ankle

Although PPG signals provide numerous benefits, it is clear from the literature that obtaining high-quality PPG signals can be difficult and challenging. As it is susceptible to artefacts, most of the work to mitigate this either use signal processing approaches [21][22][23], PPG signal in conjunction with accelerometer data [24][25][26] or even both.

3.1.2 Similar Products

It is summarised in the article in [6] that although PPG sensing provides a very promising technology due to its numerous benefits, further investigations using low power consumption to determine more health-related information must be conducted. In 2019 one such example was a paper on "A Wearable Wrist-Band with Compressive Sensing based Ultra-Low Power Photoplethysmography Readout Circuit" [27] which presents a PPG wrist-band wearable that uses compressive sampling to achieve ultra-low power consumption. This study uses a single LED channel and single site measurement method to infer useful biomarkers without compromising performance.

Other commercial wearable devices have also integrated PPG sensing technology for daily life. Some examples include the Apple Watch, Samsung Galaxy Watch, fitness trackers and even the Oura ring. These products typically use a single-site approach to measure common biomarkers inferred by PPG sensing. Some products also implement multi-wavelength sensing, using multiple light wavelengths to either measure more biomarkers [6], reduce noise [27], and potentially improve user experience (e.g., using infrared light at night instead of green light).

In summary, PPG-based technology has improved quite heavily over the years, with most of the work primarily done using a single-site and single or multi-wavelength approach. The D-SCAPE is currently unique in that, to the best of my knowledge, it is the only product that utilises a multi-site, multi-wavelength approach. Although it is clear why multi-wavelength PPG sensing has advantages, the multi-site approach might still be unclear. To explain this, it is important to discuss the current research on continuously estimating blood pressure using PPG.

3.2 Estimating Arterial Blood Pressure using PPG

Up to this point, multiple studies attempting to utilize PPG sensing to measure blood pressure (BP) have been done. Although a correlation between certain PPG signal features and blood pressure can be inferred, some work still needs to be done to achieve consistent and dependable arterial BP measurements using PPG [28]. In this section, we will discuss a couple of different ways blood pressure has been attempted to be measured using PPG signals and its differences towards the approach used in the D-SCAPE.

3.2.1 Single Site PPG Measurement

A study performed in 2013 attempted to develop an innovative continuous non-invasive method to measure blood pressure using a single-site PPG sensor [29]. This is done by measuring PPG signals and a ground truth blood pressure (BP) and then fitting this data using a regression model. This model will then be used to infer blood pressure using only the PPG waveform. Although the initial results are promising, the techniques have some drawbacks, including variability and wide limits of agreement with actual blood pressure measurements. As a result, further development and refinement are needed before the technique can be applied in clinical practice.

Another study performed in 2019 [30] utilises a spectro temporal deep neural network (ResNet) instead of a regression model, showing lower overall errors. In this study, the raw PPG signals and their first and second derivatives are used alongside personalization. On top of this, a reference dummy Regression model and a random forest model are also used for comparison. Results show that the two best-performing models are the ResNet on raw PPG with no personalization (MAE of 10.52mmHg for SBP and 7.67mmHg for DBP) and ResNet on the PPG and its derivatives with personalization (MAE of 9.43mmHg for SBP and 6.88mmHg for DBP). The standard for the validation of blood pressure that is proposed by the Association for the Advancement of Medical Instrumentation, European Society of Hypertension and International Organization for Standardization requires that the estimated probability of tolerable error ≤ 10 mmHg is at least 85%. This means that the results without personalization do not meet this requirement, however, results with personalization come close.

Lastly, a study in 2022 [31] uses a UNet with Deep supervision to measure an approximated arterial blood pressure; a MultiResUNet is then used to refine the estimated Arterial Blood Pressure (ABP). By attempting to compute the Diastolic Blood Pressure (DBP), Mean Arterial Pressure (MAP), and Systolic Blood Pressure (SBP) from the estimated ABP waveform, we get the lowest error so far with (MAE = 3.449 ± 6.147mmHg, 2.310 ± 4.437mmHg, 5.727 ± 9.162mmHg respectively). Additionally, 95% of the samples predicted have an error that lies between [-11.825: 15.0637], [-9.095: 10.357], [-22.531: 19.367]. Although it is unclear whether the results have met the standard for the validation of blood pressure, the results are very close and show some good promise.

In conclusion, much work has been done in measuring blood pressure using single-site PPG signals, with the most promising results originating from deep neural networks. It can be seen that measuring blood pressure using a single-site PPG sensor can be quite challenging to achieve the necessary level of accuracy required for clinical practice. As a result, another approach using ECG signals has been utilized to improve estimation accuracy.

3.2.2 PPG + ECG

On top of using a single PPG sensor, a multi-sensor fusion approach alongside ECG sensors was used to improve the accuracy of the BP estimation. Initially, approaches involving both PPG and ECG to measure BP were mainly unreliable [32] [33]; however, several studies have since appeared, showing initial promise in estimating BP to a reasonably good degree of accuracy. One such example is in [34], in which both linear and non-linear models are considered. It was demonstrated that a baseline efficacy of using ECG signals to estimate Pulse Arrival Time (PAT) alongside PPG, fitted to a linear model, can be achieved. On top of this, using a non-linear model produced smaller errors; however, the estimates were also coarser.

Additionally, a recent study in 2020 [35] attempted to use the same ECG and PPG sensing configuration with machine learning and extracted more informative features to improve blood pressure estimation. It was shown that using signal features of both PPG and ECG signals alongside the machine learning algorithm proposed in the article leads to reliable estimation of BP [35].

These examples illustrate the point that using a multi-sensor approach, such as an ECG + PPG combination, typically allows more accurate blood pressure estimation in comparison to a PPG signal by itself.

3.2.3 Multi-Site PPG

Although using ECG alongside PPG leads to good blood pressure estimates, ECG sensing requires contact with electrical probes on the body. Contact with the skin over long periods of time could potentially cause discomfort, which is not an ideal characteristic of a medical wearable. This leads us to the work done in [13], in which two PPG sensors are used to estimate blood pressure. This consists of leading and lagging PPG signals to calculate the Pulse Wave Velocity (PWV) from the differences between the two signals. With the introduction of a second reference PPG signal, provided high enough precision, this should theoretically allow the potential for similar blood pressure monitoring performances to that of the ECG and PPG combination. On top of this, the D-SCAPE wearable uses a multi-wavelength approach on both sites to get more features than a single-wavelength approach would have otherwise given. The mathematical equations used to estimate blood pressure are described in [13] and summarised in Section 2.1.3. It is important to note that although a mathematical approach to measure BP is taken, a machine-learning one could also be viable and potentially be looked into.

4 AFE4900 Embedded Implementation

4.1 Components

4.1.1 AFE4900

The AFE4900 chip, designed by Texas Instruments, is currently used in the original D-SCAPE wearable in the form of the evaluation module (AFE4900 EVM). This evaluation module is an embedded system which utilizes multiple hardware peripherals alongside the AFE4900 chip for biosensing applications. The evaluation module comes alongside the AFE4900 EVM software, which supports numerous functionalities, some of which include:

- ADC Capture & Analysis
- Device Configuration
- Data saving
- Filtering
- Configuration Load/Save

Although the AFE4900 EVM evaluation module provides a very convenient, easy-to-use tool for configuring the D-SCAPE wearable, in the 2nd iteration of the D-SCAPE, only the AFE4900 chip will be used alongside a microcontroller. Although using two separate AFE4900 EVM boards allows PPG acquisition between two measurement sites, these two boards have 2 independent internal clocks, which incurs a linear time-varying delay due. As stated previously, the idea is to provide a shared internal oscillator for the two AFE chips and an embedded systems application that utilises this to quantify this delay for PPG acquisition.

Additionally, there is limited expandability in using the Software GUI tool compared to an embedded application. One such example is using an accelerometer to detect whether a patient is moving and, if so, temporarily switch off the AFE reading to save power or use the accelerometer data to remove motion artefacts [36]. This means a new custom-embedded application for the D-SCAPE must be developed to use the AFE4900 chip with the particular application in mind.

4.1.2 Espressif ESP32-S3

The microcontroller used in the project will be the ESP32-S3 chip by Espressif. The ESP32 comes with the ESP-IDF (Espressif IoT Development Framework), a software development environment for Espressif chips. The ESP32 supports numerous peripherals and has a dual-core microprocessor, potentially useful for multi-processing. When considering synchronized data acquisition, the important peripherals to consider are:

- 4 x SPI interfaces
- 2 x I2C interfaces
- 4 $\times$ 54-bit general-purpose timers

Both SPI and I2C communication protocols are both valid for the project. However, some advantages warrant elucidation to select the most suitable protocol. Both interfaces allow 1 primary device to multiple secondary devices. Although this is a useful feature, it is not too important in this project as both protocols have multiple interfaces. This means that each AFE device could be communicated with its own independent interface. The multi-master feature that the I2C provides is also not too valuable for this project in particular, as there will only be 1 microcontroller used.

When considering the purpose of these communication protocols, it is more advantageous to use SPI over I2C. This is due to its superior speed over the I2C protocol, which is important for low latency in real-time applications. When configuring the ESP32 as a primary device, SPI speeds can reach up to 80MHz, translating to 80Mbit/s, whereas I2C can go up to 100Kbit/s in standard mode and 400Kbit/s in fast mode.

On top of this, SPI also allows for full-duplex communication, which is useful as writing register configurations would not cause a loss in data acquisition. The dual-core utilisation inside the ESP32 chips could also be allocated for each communication interface (1 core allocated for 1 protocol interface). This, in theory, would allow parallel communication with each AFE chip, reducing the delay between acquisitions compared to the typical serial execution of tasks by a single core.

4.1.3 General Purpose Timers

The general-purpose timer peripherals inside the microcontroller serve as a way to facilitate the integration of real-time data into the software processing pipeline. The timers can have the highest precision up to 40MHz (minimum divider of 2), which can serve as a very high precision time measurement method. Additionally, it incurs a small overhead as performing 1 timer read adds a delay of 2×10^{-6}**s (averaged over 10000 samples)** given a divider value of 80. At the targetted 1000Hz operating frequency, this adds a $\pm 0.1\%$ error which is almost negligible.

This means the general purpose timers are a good method for measuring timestamps and provide good precision at low overhead. This is important to elaborate on as the general purpose timers will be used at multiple points in implementation, such as:

1. Verifying that the AFE configuration is working.
2. AFE delay quantification.
3. Concurrent AFE measurement with timestamp association.

4. Profiling the performance of the embedded system.

4.2 Getting Started with Embedded Applications

PlatformIO is the IDE of choice due to its numerous features, different software development kits (SDKs) and frameworks. It supports the Espressif 32 platform and the ESP-IDF framework, which will include integral tools. PlatformIO can also be installed via a simple extension inside VSCode, making the IDE more convenient to use.

4.2.1 AFE4900 Library

As stated before, the AFE4900 chip does not come by itself but with an entire evaluation module and the AFE4900 EVM software application. This means in order to program on the AFE4900 chip; it will need to be separated from its evaluation module and customised to be programmable. The AFE4900 datasheet, under selective disclosure, will also need to be acquired to adhere to its specifications and protocols when performing data acquisition. Due to this, a customized board consisting of 2 AFE4900 chips was to be made for this project, in particular, providing a shared oscillator for the two chips to use. To use the board, an embedded systems application consisting of an AFE4900 library is to be implemented.

With the invaluable support and guidance provided by my PhD mentor Yuting Xu, an AFE4900 library is created and implemented consisting of multiple functions and register address mappings that use the functionality provided by the AFE4900. Additionally, an extra data structure containing the AFE configuration has also been created, which provides very simple simplification upon changing pin structures if needed. An example of how this can be instantiated is shown in Appendix C.1 Listing 1.

As the AFE4900 library implementation is similar to other embedded systems libraries (with the exception of tiny specifics in the specified datasheet), specifics of certain functions implemented are not explained and can be found in the code.

4.2.2 AFE4900 Configuration

Additional to the AFE library, a "afe4900_configs" c file was created in order to store useful AFE register configurations so that it can be easily loaded with a simple function. Each configuration, however, is only compatible with a certain operating frequency, with the original D-SCAPE-v1 configuration compatible with an operating frequency of 1.024MHz. The default internal operating frequencies of each independent AFE chip are 128KHz when using the internal oscillator and 32KHz using the shared external one.

When creating a new configuration, the mappings should be structured to ensure each appropriate LED and photodiode is used and processed within each cycle of the current operating frequency. The variable "PRPCount" stands for "Pulse Repetition Period Count" and defines the count value for the counter that sets the Pulse Repetition Frequency (PRF). In other words, it's the number of sub-cycles there are between each cycle in the operating frequency. When configuring the LEDs and Photodiodes to be turned on and off at this time, the values specified must be rounded to the nearest integer within the range of the PRPCount. An example 128KHz configuration can be seen in Appendix D Listing 4 with its visualisation shown in Figure 8. Although the entire range of values is not used, this is not necessary as finishing all LED sampling and conversions early can allow the wearable to become more power efficient.

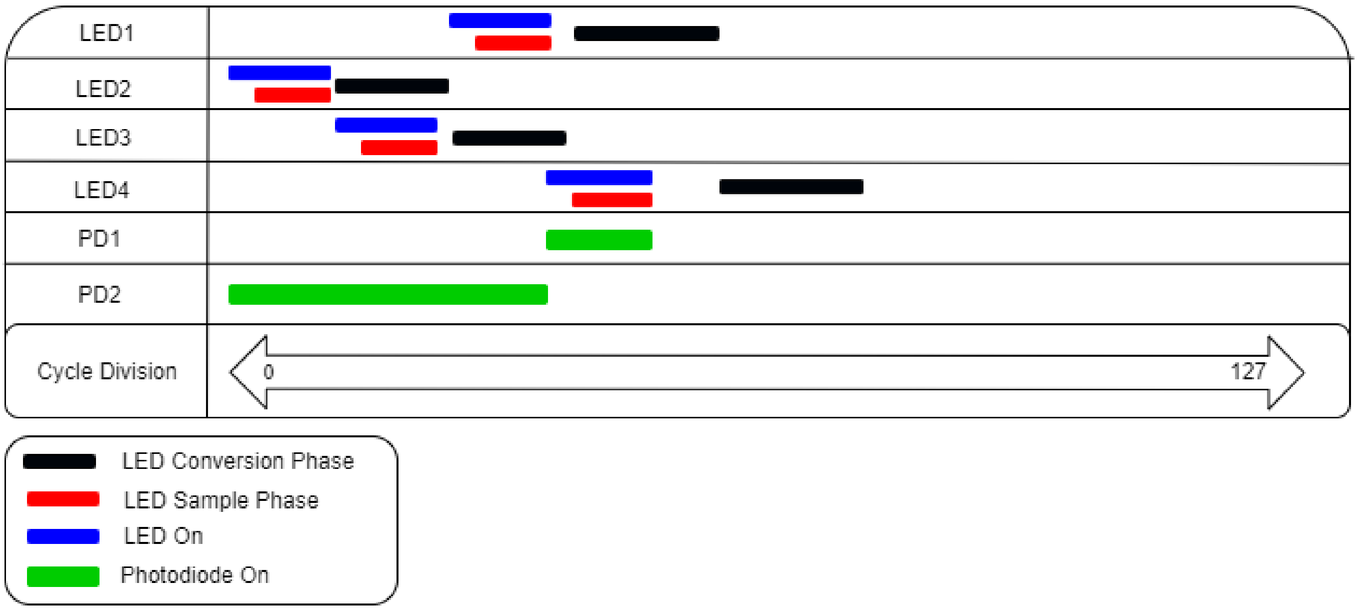


Figure 8: AFE4900 128KHz sample configuration vizualisation

The register symbol meanings are listed below:

- Names of the style $LEDxLEDSTC and $LEDxLEDENDC are registers that determine the start and end of LED x.
- Names of the style $LEDxSTC and $LEDxENDC determines the start and end of the sample LED x.
- Names with CONVST and CONVEND determine the conversion of LED x.
- TG_PDxSTC and TG_PDxENDC determine the start and end counts of the photodiode x.
- PRPCOUNT is pulse repetition count, as stated previously.

For a configuration to work properly, carefulness is required when assigning register values and ensuring each component does not overlap. These are listed below:

1. Each conversion stage of each LED does not overlap each other.

2. The start of each LED is slightly before its sample start; after both are done, this can be followed by the LED conversion stage.

3. To capture LED readings, the photodiode must also be switched on at the points the LEDs are switched on.

Additionally, it is worth noting that the change in operating frequency also affected the LED intensity outputted by the finger and wrist probes. Due to the slower operating frequency compared to the original D-SCAPE-v1, the light outputted is visibly less bright. To get similar LED intensity values as previously, the register AFE4900_LEDCNTRL1 is modified, which controls the intensity values of LEDs 1, 2 and 3.

An example of how this configuration C-File is used is in the final line on Appendix C.1 Listing 1.

4.2.3 Verifying configuration

As this new AFE setup has a different operating frequency, verifying the new configuration works as intended is very important. To do this reliably, two methods are proposed.

The first method is a software-based method, utilising one of the ESP32s general-purpose timers and measuring the delay between each consecutive measurement reading from the singular AFE. This is done by binding an interrupt service routine to a pin that indicates whenever a new measurement from the AFE chip is ready (ADY_RDY pin). When the ADC_RDY pin goes high, the callback function performs a timer read to measure the delay between the previous and current timestamp values. Using a sample number of **10000** samples (Table 1), a near consistent (99.95% of the samples recorded) $9.8 \times 10^{-4} \pm 0.1 \times 10^{-4}$s (rounded to 4d.p) is estimated which is equivalent to around **1020.4**Hz. This confirms to a reasonable degree of accuracy that the configuration is working at the expected 1000Hz operating frequency.

Time (s)	Quantity (N)	Estimated Frequency (2.d.p) (Hz)	Percentage of Sampled Data (%)
9.8×10^{-4}	9943	1020.41	99.43
9.7×10^{-4}	52	1030.93	0.52
9.6×10^{-4}	3	1041.67	0.03
0.0	2	NA	0.02

Table 1: Table Of Time Delay Between Consecutive ISR_RDY Signals

The second method is a hardware-based method, utilising an oscilloscope and cross-comparing with the software-based results. By utilizing an oscilloscope, an estimated frequency between each ADC_RDY can be read. This is done to verify the validity of the configuration, as well as the software-based method, allowing more confidence in the low overhead and high accuracy it provides in timing estimation. As seen in Figure 9 the oscilloscope estimates a frequency of 1023.40Hz between consecutive ADC_RDY's. This is a difference of around 3Hz compared to the software method above. This difference in frequency is typically due to several factors, such as the floating point precision error upon converting the timer values, the divider value not being at a low enough value which decreases precision (divider of 80 = 1MHz precision), and quantization error. The error, however, is low enough to verify that the configuration works as intended and also increases the confidence in the low overhead incurred by the general-purpose timers.

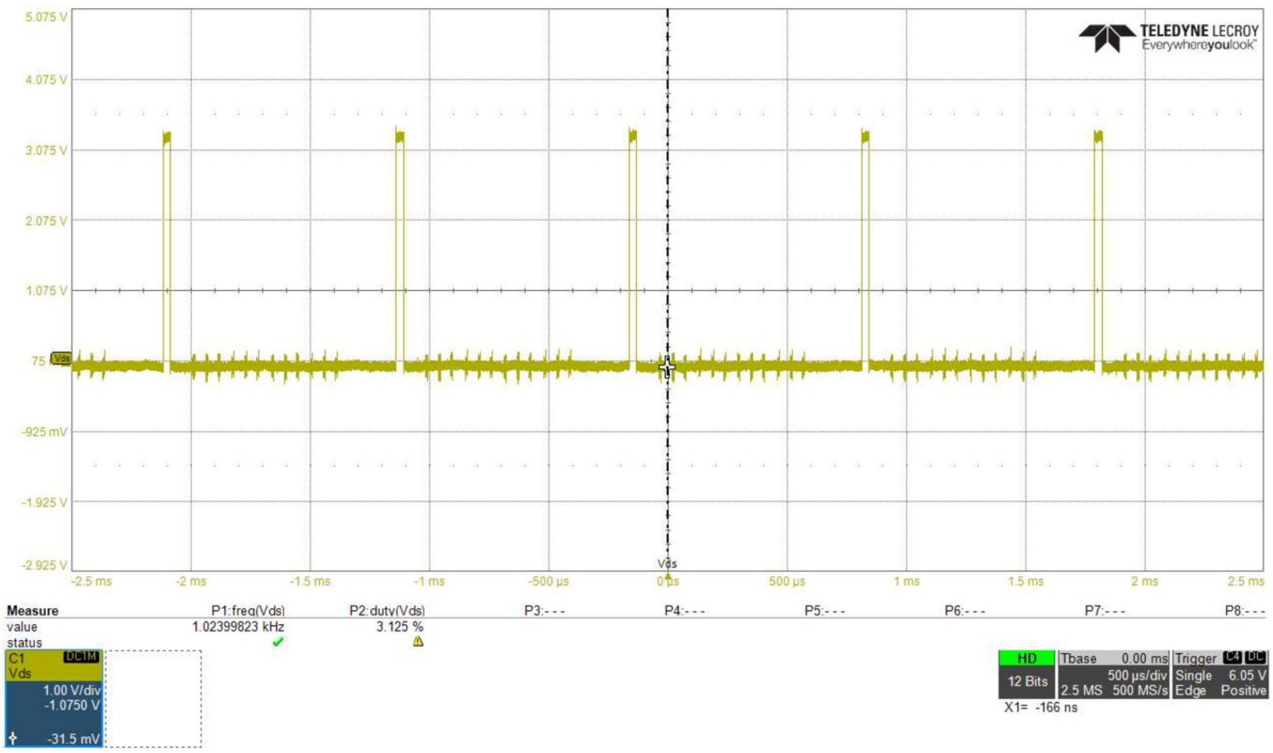


Figure 9: Operating Frequency Estimation Using Oscilloscope (ADC_RDY pin)

5 D-SCAPE-v1 Delay Problem

Now that the foundational AFE4900 library has been implemented and the AFE configuration verified, the primary objective of quantifying the delays between each chip can be progressed. As stated previously, the main issue the D-SCAPE-v1 originally faced when monitoring blood pressure was due to the delay between the two independent internal oscillators of the AFE chips. The D-SCAPE uses a PPG monitoring technique that tracks the PPG readings of a patient over a prolonged period of time and estimates an averaged biomarker. As the delay was linearly increasing, this is a big problem when looked at over an extended period.

The D-SCAPE-v1 output format for a single chip included the timestamp in the first column and the 4 PPG signal waveform values in the remaining columns. This was recorded in batches of 20, with the first row in the batch of 20 containing the timestamp and the rest with 0. An example reading of the original D-SCAPE can be seen in Appendix D Listing 5

Due to the linearly increasing delay imposed by the two independent operating frequencies, each consecutive sample read makes the AFE chips more and more out of sync. Although not initially significant, this variable delay can make estimating blood pressure difficult. This can be demonstrated with the sine plot in Figures 10 11, which shows the slight difference in frequency leads to the sine waves becoming more and more out of sync over time. Additionally, as each timestamp is associated with a batch of 20, this can lead to further inaccuracies within the sample readings not containing timestamps (Since the time difference between each batch sample is to be estimated).

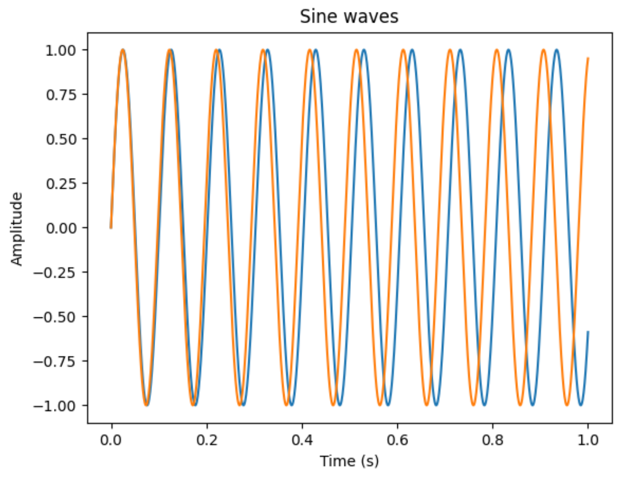


Figure 10: Sine wave plots

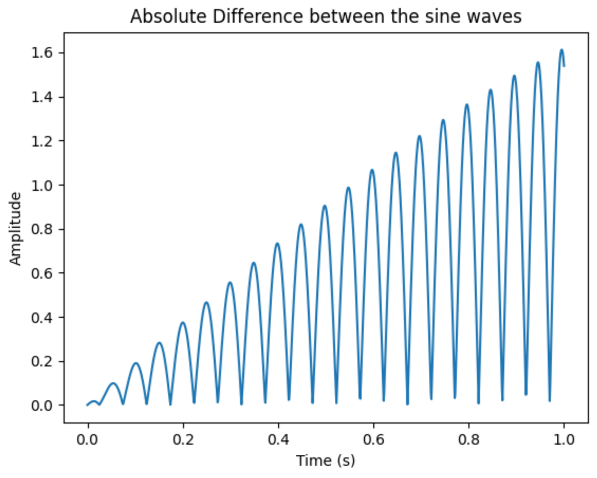


Figure 11: Absolute difference over time of sine waves

Furthermore, since an increase in blood pressure can be as small as a 2-sample change at 1000Hz, this variable delay poses a major issue in estimating Pulse Wave Velocity. This will, in turn, lead to poor blood pressure difference estimation.

As the implementation of the new modified board includes a shared external oscillator, this means that the two AFE chips should, in theory, be working at the same operating frequency. Provided the same configuration, this should lead to each AFE chip performing the same series of operations at a constant time. This will theoretically change the original linear time-varying delay to a constant one, significantly simplifying the problem.

5.1 Quantifying the delay

Although this is good theoretically, this still requires verification. This is done by utilising a similar ISR setup with the ADC_RDY pin of each AFE described in Section 4.2.3. An average of **113978** samples is used to measure the timestamp delay between consecutive samples between the two PPG readings. This resulted in a mean of -8.987×10^{-5}**s** with **99.94%** of the sample population equal to a constant -9×10^{-5}**s** (rounded). The corresponding table is shown below in Table 2.

Time Difference Between Consecutive Reads Of The Two Chips (s)	Quantity (N)	Percentage of Sampled Data (%)
-9×10^{-5}	113914	99.94
-6×10^{-5}	1	8.77×10^{-4}
-1×10^{-4}	44	3.8×10^{-2}
-8.9×10^{-4}	9	7.90×10^{-3}
-8×10^{-5}	4	3.51×10^{-3}
-8.8×10^{-4}	6	5.26×10^{-3}

Table 2: Table of the difference between consecutive reads between the two AFE chips

As seen from the results, the use of the external oscillator confirms the theoretical assumptions proposed, showing a fairly consistent delay between PPG waveform readings between the two chips. Additionally, the variance within the table might be due to other factors, such as floating point precision error. This, however, is not significant since this number is very small compared to the consistent delayed measured.

5.2 Implementation

In theory, the optimal embedded system setup would be to have both AFE4900 chips working completely synchronized while using a shared oscillator. However, this is very difficult to implement as this requires each AFE chip to start at the same clock cycle. For the current project, this is an infeasible solution since this would mean creating a customised microcontroller that starts the SPI transactions on both AFE chips simultaneously. Therefore for the remainder of the project, this constant delay will be a variable considered for this specific implementation. However, the implementation that uses a customised microcontroller could be a potential implementation worth looking at in the future.

Due to the limitation imposed by this constant delay, a new method is implemented, utilising the general-purpose timers used when measuring each PPG signal reading. The implementation plan consisted of performing an SPI read and taking its associated timestamp whenever the ADC_RDY pin is set to high. This is because the ADC_RDY pin indicates the availability of new data that needs to be read immediately. Failing to execute the read operation within the operating frequency time frame could result in data loss. This means for 1000Hz, the window to read the data provided by the AFE is 1ms.

5.2.1 Iteration 1

The first iteration of the method implemented is shown in Figure 12, which attempts to perform an SPI and timer read inside the ISR function, which is then sent to an ISR safe queue as a message packet. Since the ISR function is highly prioritised and the ISR safe queue is a non-blocking data write, this should be executed quickly. This aims to minimise the latency between an ADC_RDY and an SPI read to eliminate potential data loss. Although good in theory, this method does not work primarily due to the attempt to perform an SPI read inside an ISR.

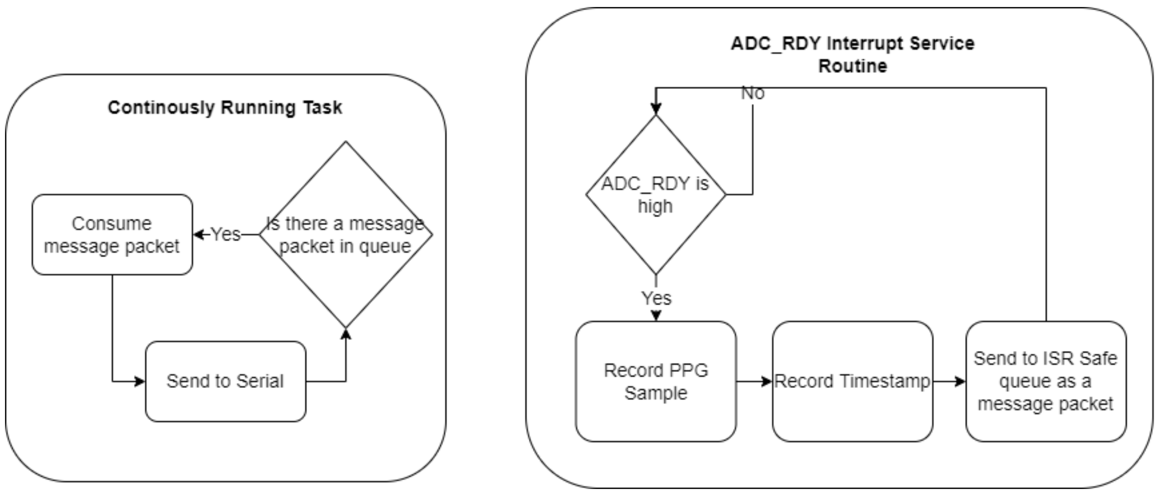


Figure 12: Iteration 1 flowchart diagram for each AFE

Performing SPI transactions using the ESP-IDF Master Driver falls under 2 categories:

1. Interrupt Transactions
2. Polling Transactions

Interrupt transactions block the transaction routine until the transaction completes, allowing the CPU to perform other tasks [37]. In the context of the D-SCAPE-v2, this could be writing data received previously to serial, making interrupt transactions very favourable. Multiple transactions will be queued, and the driver will automatically handle them in a queue-like manner. Polling transactions, on the other hand, don't use interrupts or queues. The routine will keep polling the SPI Host's status bit until the transaction is finished, which results in a smaller transaction duration. Although favourable, a disadvantage is that the CPU is busy while these transactions are in progress [37].

When performing an interrupt transaction SPI read, this calls the ”spi_device_transmit” function, which ”sends an SPI transaction, waits for it to complete, and returns the result” [37]. This consists of queuing a transaction, enabling the SPI interrupt, and performing multiple FreeRTOS function calls which are not ISR safe (e.g., spi_device_get_trans_result uses the function xQueueReceive instead of xQueueReceiveFromISR). Lastly, it will then wait and return the result. As this is an interrupt transaction, calling this from an ISR induces concurrent interrupt handling. This will, in turn, introduce additional complexities and risks in the embedded system. Furthermore, the documentation states that this function is not thread-safe and, thus, not recommended to be called within an ISR. Additionally, waiting on a queue from within an ISR is generally discouraged due to several reasons:

1. Blocking Behaviour: The blocking behaviour from queuing a request can lead to delays in servicing other interrupts and even the main loop itself since an ISR is an asynchronous event that halts the execution of the main loop. Typically ISRs are meant to be performed quickly and efficiently with minimal processing.

2. Priority Inversion: Priority inversion occurs when a lower-priority task, such as waiting on a queue, holds up a higher-priority task. Although, in this case, waiting on the queue is desirable, this is still typically considered bad practice and can result in unexpected behaviour.

3. Limited Scope: Since ISRs typically operate using a limited scope with restricted access to memory and system resources, this limits the functionality of an ISR. Calling such functions can be dangerous as the program within the ISR might not have the available resources within the called scope to finish the task leading to bad behaviour.

Due to these reasons and the numerous errors that appear in the embedded application, this makes this method unsuitable to use.

5.2.2 Iteration 2

Since SPI reads can't be performed from within an ISR, the second iteration of the method is implemented, shown in Figure 13. This second method sends a polling flag into an ISR safe queue for a continuous polling function to check (xQueueSendFromISR and xQueueReceiveFromISR). Upon receiving the flag, this prompts the function to perform an SPI and timestamp sample reading. This is then sent into another task with high priority, which is used to serially print the recorded data. The intuitive reasoning for this method is that the polling flag will indicate that an ADC_RDY is set to high. Provided a low enough delay between when the polling flag is sent and when it is received, this should act identically similar to the

ADC_RDY pin. The difference, however, is that the SPI communication will be executed in a high-priority task instead of an ISR function. Since high-priority tasks do not operate under the limited scope and resources of an ISR function, this should allow the regular functionality of the SPI library.

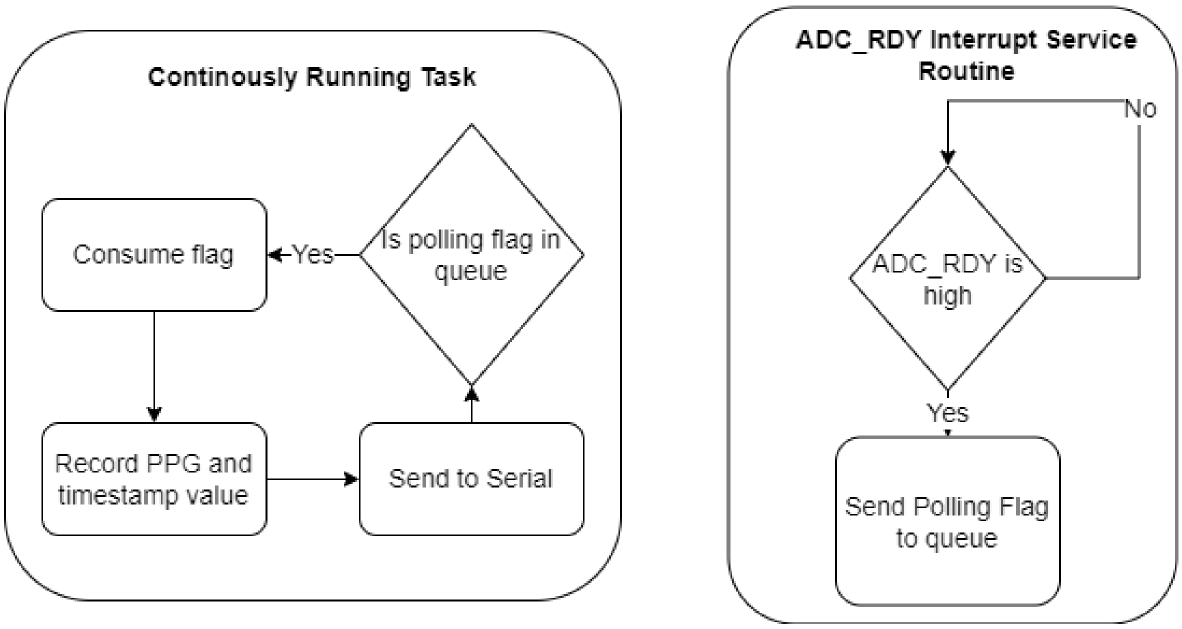


Figure 13: Iteration 2 flowchart diagram for each AFE

Although this method, at first glance, appeared to work as intended providing valid PPG sample values upon prompt, it suffered a major drawback primarily due to the queue data structure used (QueueHandle_t). Upon profiling, it can be seen that the time between sending a polling flag from the ISR and receiving the polling within the continuously running function provides a delay of approximately 0.2s. As the targetted operating frequency of the D-SCAPE is ideally 1000Hz, this gives a timestamp error of 20000% and therefore skipping 199 samples before actually reading the PPG value from the associated ADC_RDY interrupt flag. Additionally, the delay incurred from sending and receiving the polling signal using the queues are quite significant to the point that it downsamples the signal readings. As the variable delay incurred by the sending and receiving of the polling flag is of a wide range and quite large, this makes the ADC_RDY pin pointless. This is because this would result in delayed PPG readings, missing multiple cycles from the one meant to be recorded. A visualisation example of this can be seen in Figure 14.

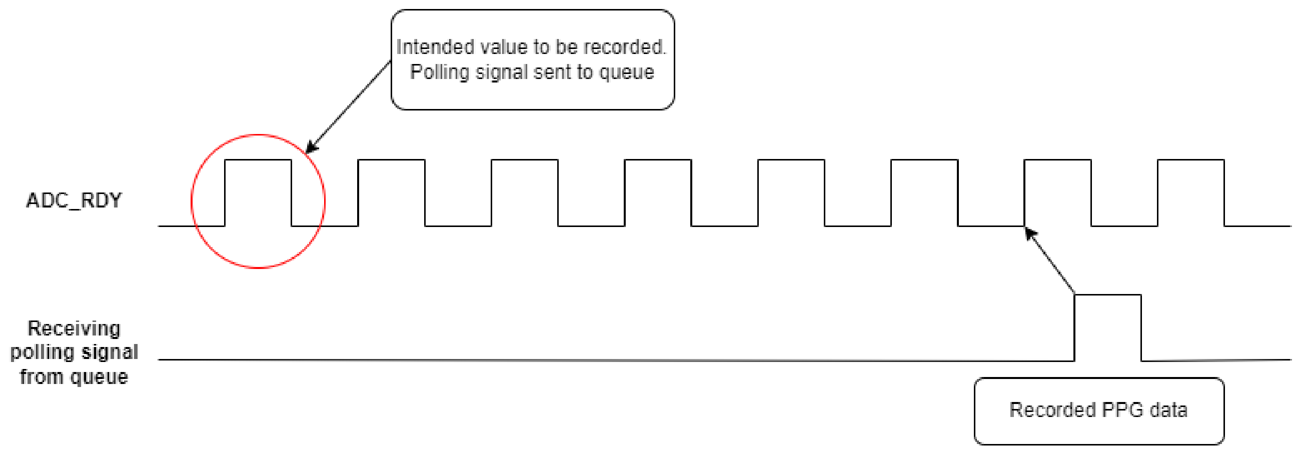

Intended value to be recorded. Polling signal sent to queue
ADC_RDY
Receiving polling signal from queue
Recorded PPG data

Figure 14: Polling Signal Visualisation

5.2.3 Iteration 3 - Final

A third and final iteration is implemented to circumvent the two issues mentioned above. Given that SPI readings cannot be done within the context of ISR, it is clear that this must be done inside the main program loop. Therefore the main program loop can be set to record the PPG samples at a similar interval in which the AFE updates the data. Initially, this was done by setting the general-purpose timers to monitor the timestamp at which it recorded data continuously. Whenever the recorded timestamp has changed (i.e., reached 1 cycle's worth of the current AFE operating frequency), a new PPG sample is recorded, and the timestamp is updated. This can be visualised by the flowchart in Figure 15. This is to provide a loop fast enough to record PPG samples at the same operating frequency specified in the AFE chips.

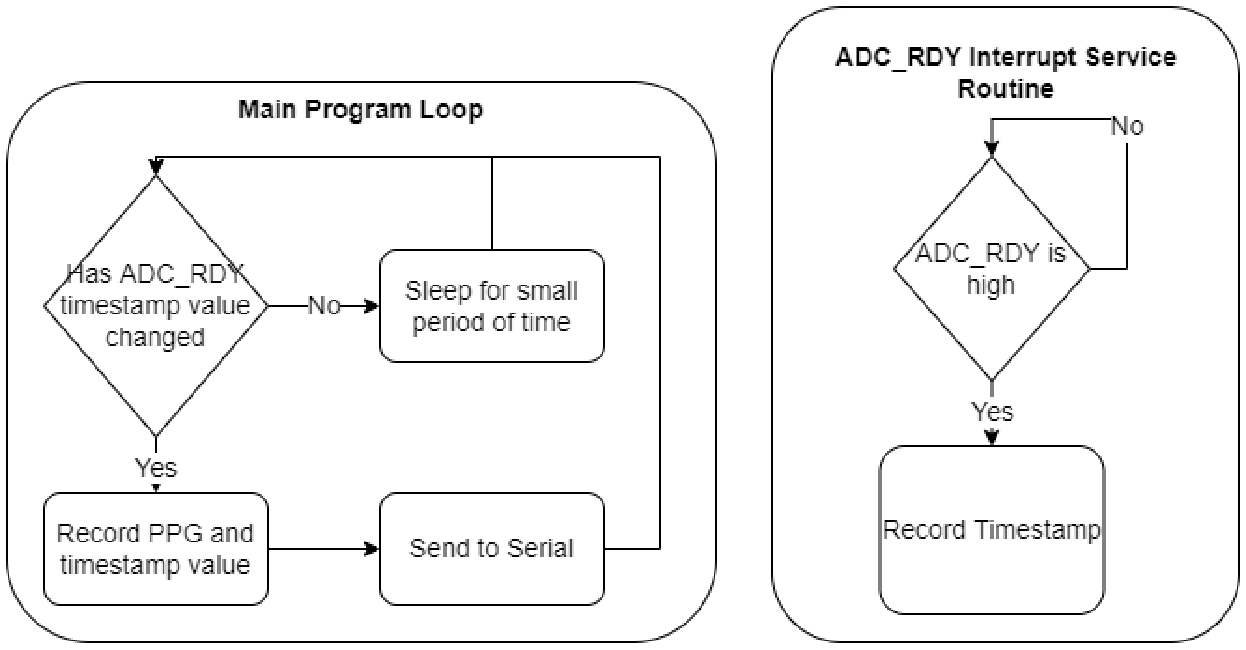

Main Program Loop
Has ADC_RDY timestamp value changed
No
Sleep for small period of time
Yes
Record PPG and timestamp value
Send to Serial
ADC_RDY Interrupt Service Routine
No
ADC_RDY is high
Yes
Record Timestamp

Figure 15: Iteration 3 flowchart diagram for each AFE

Although this is a good start, it must be recalled that the two AFE chips do not update the data simultaneously due to the constant delay between them. This means the current reading provides good PPG signal data but associates inaccurate timestamps. Additionally, if the AFE operating frequency does not match the frequency assumed inside the main loop, there is a possibility that some data points will be duplicated or missed, contingent upon whether the AFE operating frequency is higher or lower.

A solution implemented is by utilising the ISR function alongside registers maps whenever the ADC_RDY is set to high. This method is very similar to the one proposed in the second iteration, with the change of swapping the ISR safe queue for a register map. Register maps are much more efficient than ISR-safe queues and are executed sequentially. To do this, two ISR functions are set up (1 for each AFE chip), which record the 64-bit timestamps at which each ADC_RDY pin is set to high. Additionally, inside the program loop, the program checks that both timestamps inside the ISR have been updated. Whenever an ADC_RDY is set to high, the corresponding time variable keeping track of it also gets updated, indicating a new PPG sample reading is to be read.

As we have shown previously, there is a constant delay between the two chips. This means recording both PPG readings simultaneously should be viable when each of their ADC_RDY pins has been set to high. This is because, unlike the linear time-varying delay, the constant delay means that the ADC_RDYs between chips will **always** alternate, and there will never be a moment where 2 consecutive ADC_RDYs from the same chip occur provided both chips have the same configuration.

It is worth noting a second method which uses a simple boolean mapping instead of a 64-bit timestamp alongside an initial calibration period, is also considered due to its more simple and power-efficient implementation. This is done by initially calibrating the wearable to estimate the operating frequency of the AFE chips, including manufacturing tolerance, and estimating the constant delay between the two chips. This works since the timestamp can only be measured once in the beginning, provided the delay and operating frequency do not change; this provides a good estimation of the signal without the overhead of measuring the additional timestamps. This was, however, rejected due to some of the reasons discussed in Section 5.3.

5.2.4 Verification

After implementation, it is important to verify that everything works as intended. This is done using numerous different techniques, which include:

- Checking the differences between recorded timestamps on the serial monitor and ensuring this matches the expected frequency.
- Sanity checking by plotting and filtering the sampled PPG signal and estimating a biomarker such as heart rate.
- Ensuring concurrent PPG readings can be done on both AFE chips.

For this section, a lower operating frequency of 100Hz was used for each AFE chip due to some performance bottlenecks mentioned in Section 5.3. Upon recording 10,000 PPG samples including their corresponding timestamps (Table 3), an averaged delay of 9.766×10^{-3} can be estimated which results in **102.40Hz**.

Time (s)	Quantity (N)	Estimated Frequency (2.d.p) (Hz)	Percentage of Sampled Data (%)
9.766×10^{-3}	9997	102.40	99.98
10.742×10^{-3}	2	93.09	0.02

Table 3: Table Of Consecutive Time Delay And Quantity (N = 10,000, Operating Hz = 100Hz)

The difference between the aimed 100Hz operating frequency and the estimated 102.40Hz operating frequency could be due to several reasons. The first issue could be a floating point precision error when converting the unsigned 64-bit timer value to a decimal value. Secondly, some manufacturing tolerance within the operating frequency could also have affected the output readings as the measured 102Hz operating frequency is fairly within reason of 100Hz. Lastly, although there is a delay between the time an ADC_RDY pin is set to high and the timestamp value output, this is almost negligible and fairly consistent, therefore most likely not affecting the output frequency. Typically, a difference in targetted and actual operating frequency can be an issue for a high-precision medical wearable. However, this problem is not significant due to the implementation method. Since the application can quantify the difference between the targetted vs actual operating frequency, provided the actual operating frequency is within reason, this poses very little effect on the actual efficiency and effectiveness of the medical wearable.

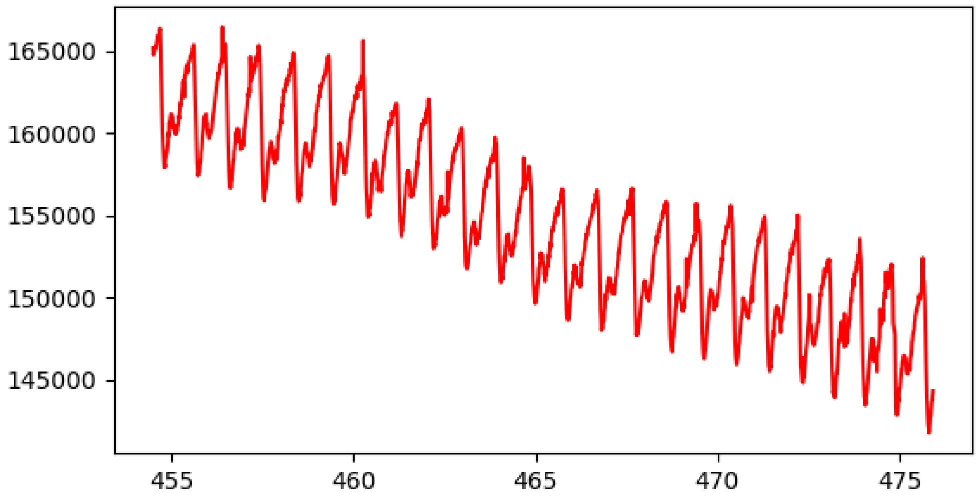


Figure 16: Raw, Unfiltered, Inverted PPG signal Plot

By recording a sample PPG reading consisting of around 2,100 samples (around 21 seconds of data), the raw inverted PPG signal can be seen in Figure 16. As seen from the inverted PPG plot, observable systolic and diastolic peaks and dicrotic notches are seen. However, a large amount of noise also causes the signal to contain erratic and jagged features.

Upon using a Bandpass Chebyshev type 2 filter to remove the DC and noise components, we get the following plot in Figure 17. This plot shows a much cleaner version of the original plot in Figure 16, showing the similar expected features of a common PPG waveform.

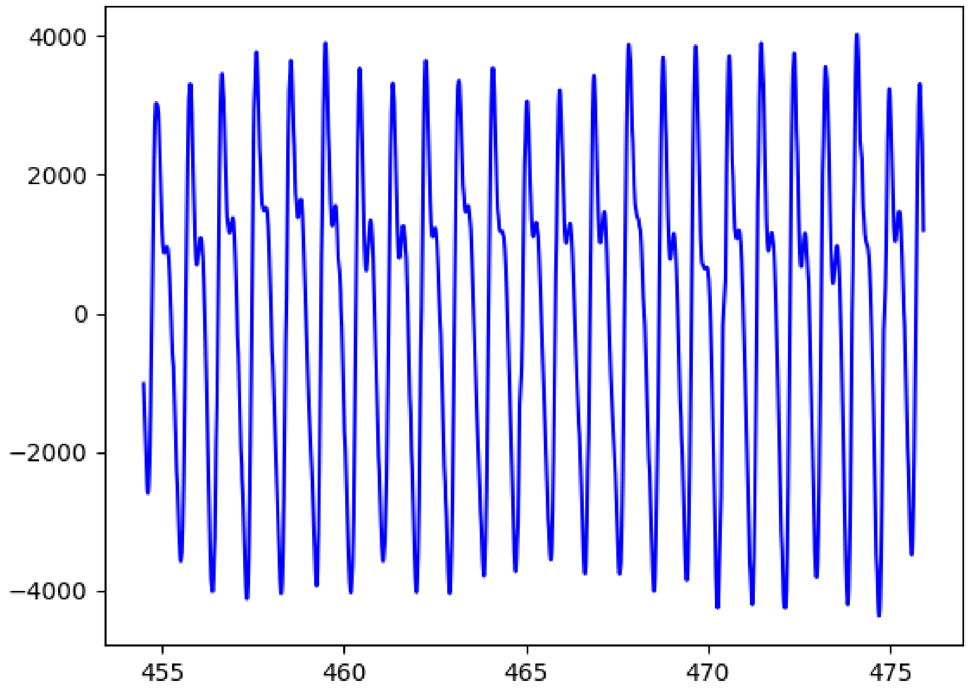


Figure 17: Band-pass Filtered Signal with parameters (N = 4, rs = 50, lp_cutoff = 10, hp_cutoff = 0.35, fs = 100)

Upon performing the Fast Fourier Transform algorithm and taking the largest frequency component, an estimated **65.75** beats per minute (bpm). This is verified alongside an Apple Watch with an estimate of **64.9**bpm showing promising results in the biomarker sanity check test (Table 4).

PPG Inferred Readings (bpm)	Apple Watch Readings (bpm)	Absolute Error (%)
65.79	64.9	1.3

Table 4: Table Of Estimated Heart Rate Readings From Both PPG Data & Apple Watch

Lastly, it is important to test whether the application can perform concurrent PPG readings, as this is one of the main features required in the D-SCAPE wearable. To do this, two finger probes with the same configuration are used to show that both chips provide similar plots. The same finger (index) will be used for both finger probes as well as how the arm is extended since this will provide identical arterial lengths for blood pulse propagation.

A plot of two filtered PPG waveforms of around 2000 samples is shown in Figure 18. The big spike around the 16-second timestamp indicates the point at which the two index fingers

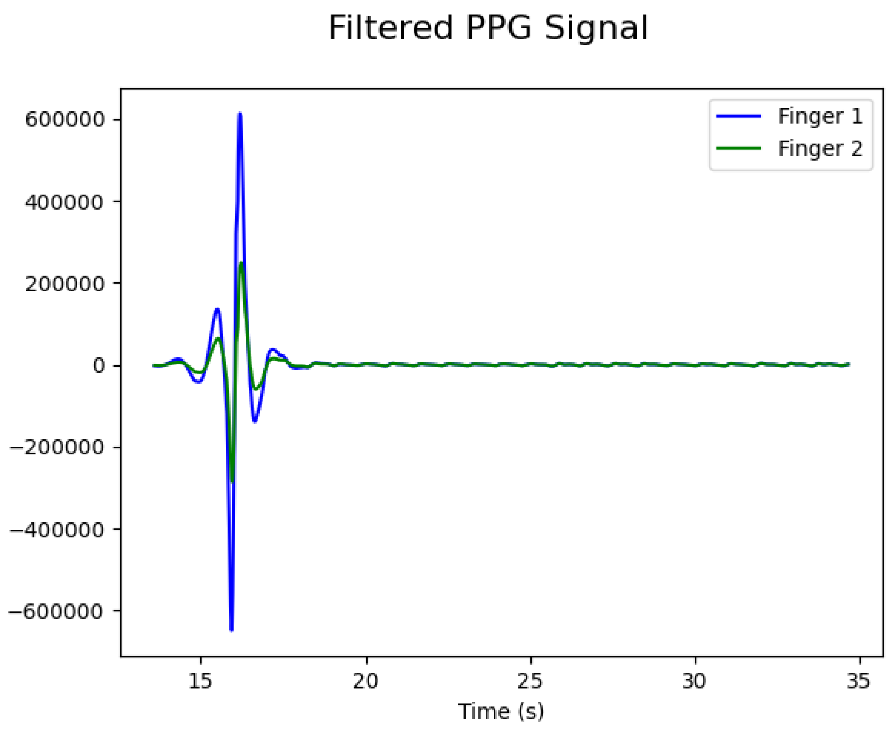


Figure 18: Plot of PPG signals of index fingers with parameters (N = 4, rs = 50, lp_cutoff = 10, hp_cutoff = 0.35, fs = 100)

are placed inside the finger probes. Additionally, it also explains the unexpected behaviour around that time period. As expected, its zoomed-in version in Figure 19 shows a very strong similarity between the two plots. Since 2 index fingers are used with roughly equal arterial lengths, the peaks of the heartbeats recorded are expected to arrive at the exact same time. From Figure 19, it can also be seen that there is a slight difference between the peaks of each heartbeat. By manually inspecting the samples at which each waveform reaches its peak, it can be seen that the difference between the two waveforms is fairly constant, but with some minor differences. This indicates that the waveforms could have been slightly corrupted by noise (e.g., noise artefacts) which causes a very small inaccuracy between the two signals.

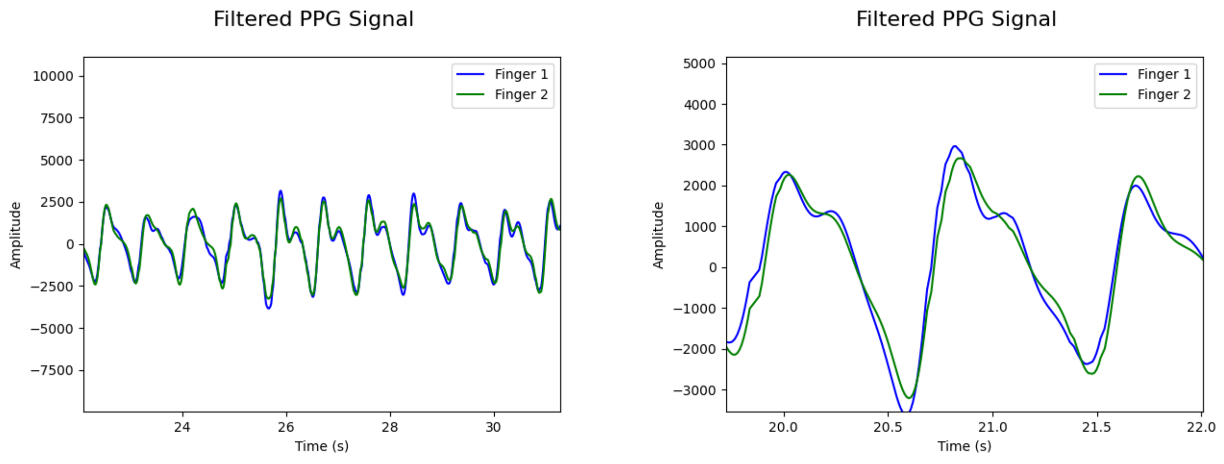


Figure 19: Zoomed in version of PPG signals

Regardless of this slight inaccuracy, however, these are still very promising results which clearly illustrate the expected behaviour that the heartbeats have travelled similar arterial lengths to be recorded at the two index fingers. Upon closer inspection in Figure 20, it can be seen that the true absolute peak of the heartbeats is still not estimated properly, in which the true peak typically lies between two sample recordings. This indicates that a higher precision is needed to infer better heart rate peaks properly and, in turn, calculate a better Pulse Wave Velocity (PWV) estimate.

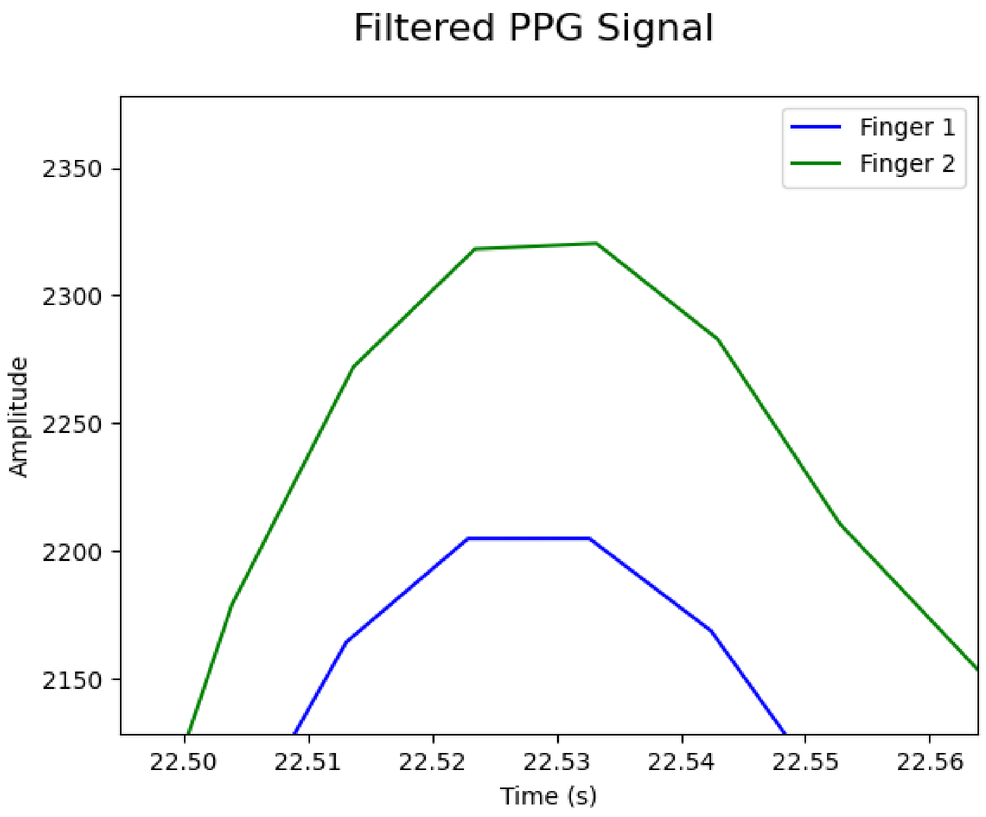


Figure 20: Zoomed in version of PPG Signal peak

5.3 Performance Issues

A major issue was encountered upon attempting to increase the operating frequency of the current application from 100Hz to 1000Hz. Although it has been verified that both AFE chips are working at 1000Hz, occasionally, there are points at which the PPG signal recording would be limited to the speed of the ESP32 microcontroller. This was initially discovered through unexpected outputs from filtering the PPG waveforms and verifying through timestamp difference measurement.

It is also worth mentioning why the more efficient Boolean mapping method in Section 5.2.3 was rejected, and the approach of monitoring the timestamp of each sample was taken. Variance in performance due to bugs or bottlenecks can happen at points in the application. It is important to monitor this variance when the microcontroller is not reaching minimum performance, as this wearable is intended to operate on real-time data. As this is a medical wearable, it is crucial to guarantee that the minimum operating frequency is met, as failing to do so can either lead to loss of information or bad data fed into the signal processing pipeline. The boolean mapping calibration method measures each AFE chip's operating frequency and assumes this is constant throughout the program. If this method is implemented, there would be no way to monitor this performance bug, and it would appear like the program is working as intended. On the other hand, by measuring the timestamp at which each sample is taken, the program can quantify the current operating frequency the microcontroller is at, which increases the robustness of the wearable.

By setting a targetted operating frequency of 1000Hz (same as the original D-SCAPE), taking 8 consecutive LED readings, and taking the average difference between each recorded sample, the average delay and operating frequency are estimated to be 9.766×10^{-3}s and 102.40Hz respectively (**10000** samples), around 10 times lower than the targetted operating frequency. Upon calculation, this means approximately 9 samples are missed before a sample is measured, significantly decreasing the waveform's precision.

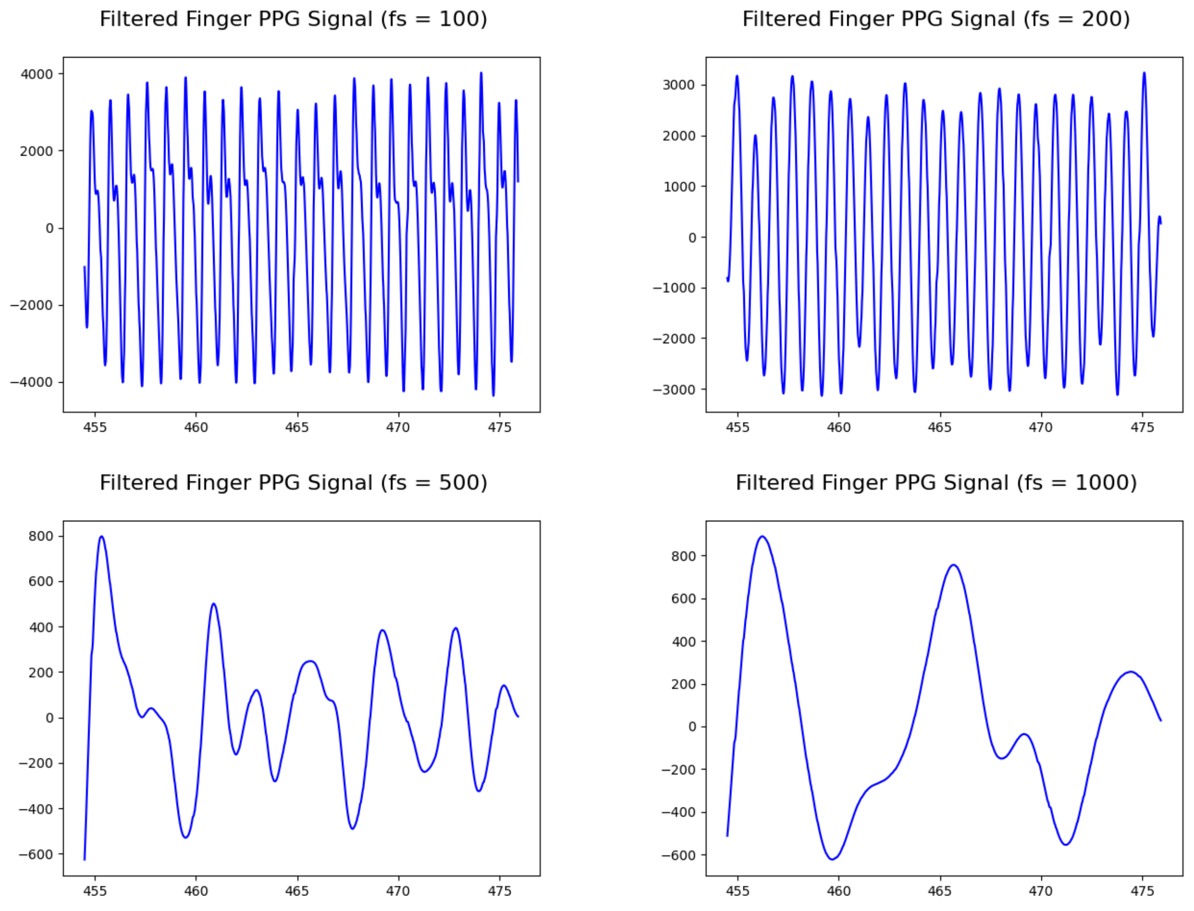


Figure 21: Filtering of PPG signals with different sampling frequency parameters (Original fs = 100)

If the provided sampling frequency does not match the actual sampling frequency recorded by the medical device, this can result in bad filtering due to incorrect parameter assumptions and lead to inaccurate PPG signal feature extraction. An example of this can be shown in Figure 21 in which any other sampling frequency parameters provided than the original 100Hz resulted in major information loss in the PPG waveform. Furthermore, as each AFE chip is targetted to operate at 1000Hz, this means frequently missing a small amount of samples results in a major change of sampling frequency as shown in Table 5.

Samples Missed (N)	Operating Frequency (Hz)
1	1000
2	500
3	333.33
4	250
5	200
6	166.7
7	142.9

Table 5: Table of samples missed and its corresponding effect on the operating frequency (Original fs = 1000)

As the original D-SCAPE-v1 uses an operating frequency of 1000Hz, it is crucial for the v2 to have this as a minimum requirement in addition to the high precision required to estimate blood pressure. As previously specified, a change in Pulse Wave Velocity (PWV) can change the waveform in as small as 2 samples. Therefore if the D-SCAPE-v2 cannot reach a minimum operating frequency of 1000Hz, this will heavily impact its biomarker monitoring capabilities.

5.4 Optimizations

So far, it has been verified that consecutive PPG waveforms can be read from the two AFE chips. Additionally, the constant delay between the two chip frequencies can be quantified, as well as monitoring the microcontroller's signal acquisition performance. However, as summarised in the previous section, a severe performance bottleneck causes variable performance and sampling rates and makes the PPG readings inconsistent. As the D-SCAPE-v1 operates at 1000Hz, this is a minimum performance requirement that needs to be achieved.

The Espressif guide in maximising execution speed [38] was followed to attempt to maximise the execution speed of the code. The aim was to optimise the execution times to speed up the microcontroller enough to operate at a consistent 1000Hz.

5.4.1 Improving Overall Speed

There are simple optimisations which can be done that help to improve overall execution speed. Some such examples are:

- Changing the Flash SPI Speed from 40MHz to 80MHz. This should double the speed at which code is loaded or executed from the flash. This, however, can come at the cost of more power.
- Removed all instances of floating point arithmetic (floats) and double precision floating point arithmetic (double). This is because floating point calculations are slower than integer/fixed point calculations. Additionally, the ESP32 has one singular floating point hardware unit, which can slow down the program if numerous floating point calculations are needed. As there is no double precision hardware unit for calculations involving doubles, this is emulated in software and is, therefore, very slow. All instances where both data types are used are replaced with either 32 or 64-bit integers, and any conversions to floats are moved later down in the pipeline.
- Reduced logging overhead. This simply consisted of minimising the characters to be printed to serial to the minimum required to transfer the information.

5.4.2 Targetted Optimization

Another optimization done was to attempt to increase loop execution code by reducing cache misses. By default, all code in the app is executed from the cache [38]. Therefore the application can slow down if there is a ”cache miss”, in which the CPU will stall until the instructions are loaded from flash memory.

Utilizing the IRAM (Instruction RAM) keyword allows frequently used instructions to be moved into the IRAM. IRAM, being a limited memory resource, allows for the loading of functions copied into it during boot time, ensuring that they consistently execute at maximum speed thereafter [38]. This should allow the continuously executing program to have no cache misses, thus speeding up performance and potentially decreasing performance variability.

5.4.3 Multiprocessing and Task Priorities

By default, the main task (app_main) is pinned to 1 core (Core 0) and is a function that executes with minimum priority. As the ESP32 contains 2 cores, this clearly does not exploit its capabilities optimally. Additionally, since the main task loop has minimum priority, other background tasks are potentially being executed before the data loop. An example of a task with higher priority is the general-purpose timer system tasks for the ESP32, used to log timestamps. This task is pinned to 1 core (Core 0) and can execute callbacks with high priority (22) taking precedence over the main program loop.

For this optimisation, the main program loop (recording PPG data and sending to serial) is turned into a task which utilises both the CPU cores, thus allowing multiprocessing capabilities. Additionally, this task and the ISR callback functions are set to be a maximum priority, taking precedence over any other tasks the ESP32 has to execute. Increasing the task priority of the data loop should force the ESP32 to execute the primary task first before moving on to any other background processes. Additionally, the utilisation of both CPUs to perform the task instead of 1 should allow faster execution of such tasks.

5.4.4 Logging Optimisations

Currently, the ESP32 uses the default serial output channel as UART0, which has an unchangeable baud rate of 115200 bits per second. Each transmission packet outputs 32x8 LED values as well as 64x2 timestamp values. Given 1000Hz, this results in 384000 bits per second from the calculations shown in (10). This induces a major bottleneck upon writing data into serial as the baud rate is insufficient to achieve the specified requirements. This will continuously increase the data inside the transmission buffer and cause unexpected behaviour or bottlenecks. Therefore changing the serial output channel to Custom UART and changing the baud rate to a sufficiently high number (e.g., 921600 bits per second) should ensure that the serial output does not serve as a bottleneck in achieving a 1000Hz operating frequency.

$$(32 \times 8) + (64 \times 2) = 384 \text{ bits/transaction} \tag{10}$$

$$384 \times 1000 = 384000 \text{ bits/second} \tag{11}$$

Additionally, different ways to log, such as fwrite, write, printf, and ESP_LOGI, have all been measured to optimise the output function used. ESP_LOGI as a serial output logging function is the worst by a large margin due to the numerous additional overhead it introduces as it is the most complex function out of the four.

Until this point, the printf function has been the primary method of logging. This function converts integers to characters before sending instead of using the raw bit data. This would be slow compared to the other two output functions, which directly output the raw integer values into serial. As such, the printf function will not be used as the main logging function in the embedded application.

Therefore, the optimal logging function is between the fwrite or write function. Upon performing a sample of 4 LED writes averaged with 100 000 samples, the measured speeds of fwrite are $1.88 \times 10^{-}4$s whereas write is measured at $2.2 \times 10^{-}4$s. From the results, it can be seen that the fwrite function is slightly faster than the write function. However, since the fwrite function is designed to be portable, it might not be optimised for speed and therefore have a certain overhead and is buffered. The write function, however, is a platform-specific unbuffered syscall, potentially making it a more optimal choice for this particular application [38].

5.4.5 Code Profiling

The next step in attempting to optimise the program is to profile the code to get a better understanding of where the bottlenecks occur. The current loop of the main application can be broken down into 2 simple stages, 1. Reading Stage, 2. Writing Stage. By profiling both sections of the code with respect to the number of LEDs between the ranges of 1-8, the corresponding read and write times (**10000** samples) can be seen in Figure 22 as well as its corresponding Table in Table 6

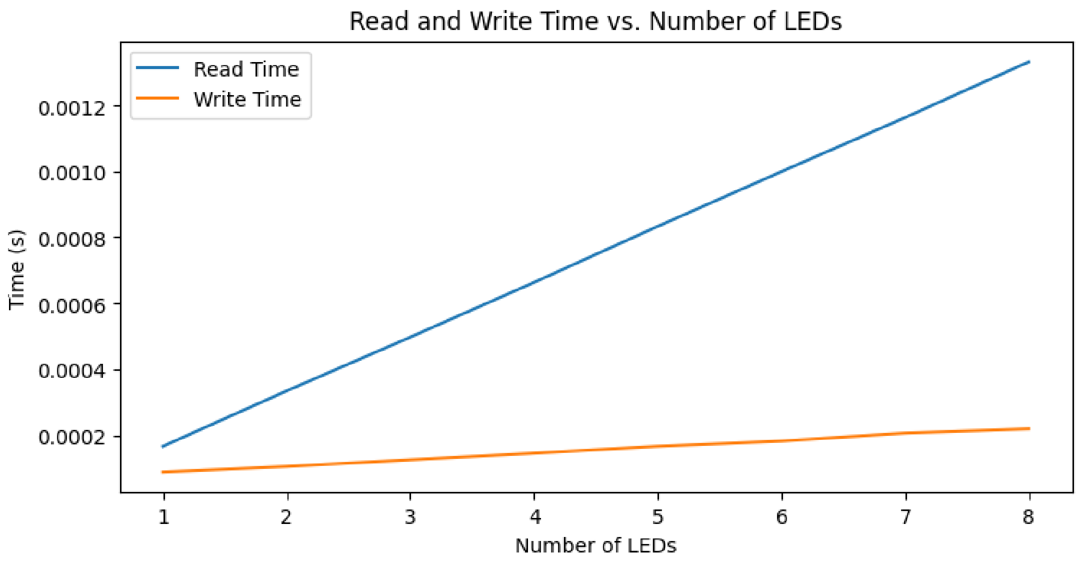


Figure 22: Number of LEDs To Read and Write Time Plot

Number of LEDs (N)	Read Time (s)	Write Time (s)	Maximum Operating Frequency (Hz)
1	1.66×10^{-4}	8.8×10^{-5}	3937.0
2	3.34×10^{-4}	1.05×10^{-4}	2277.9
3	4.97×10^{-4}	1.25×10^{-4}	1607.7
4	6.63×10^{-4}	1.45×10^{-4}	1237.6
5	8.83×10^{-4}	1.66×10^{-4}	1001.0
6	9.99×10^{-4}	1.82×10^{-4}	846.7
7	1.163×10^{-3}	2.06×10^{-4}	730.5
8	1.332×10^{-3}	2.19×10^{-4}	644.7

Table 6: Table of Read and Write Times given number of LEDs

As seen in the Figure and the calculated gradients in Table 7, an increase in the number of LEDs sampled results in a much bigger increase in read times than write times. As the read times are already quite low, improving this may not be valuable. Improving read times, however, is more difficult than improving the write times as this requires further lower-level improvements with regard to the SPI communication library. Due to this, a different approach is implemented to maintain the high precision needed for blood pressure monitoring.

	Gradient
Read	1.666×10^{-4}
Write	1.871×10^{-5}

Table 7: Estimated Read & Write Gradients

Figure 23 shows the maximum operating frequency the microcontroller can achieve given the number with respect to the number of LEDs sampled. Although 1000Hz is necessary for good blood pressure estimation, other biomarkers, such as heart rate, do not need precision this high. Inferring blood pressure also requires only 2 LED samples (1 from each AFE chip); this is typically either LED 1 or LED 4. Therefore, instead of attempting to sample every LED at 1000Hz, the system can sample a single LED channel on both AFE chips (e.g., LED 4) at 1000Hz and sample the other LEDs at a lower rate.

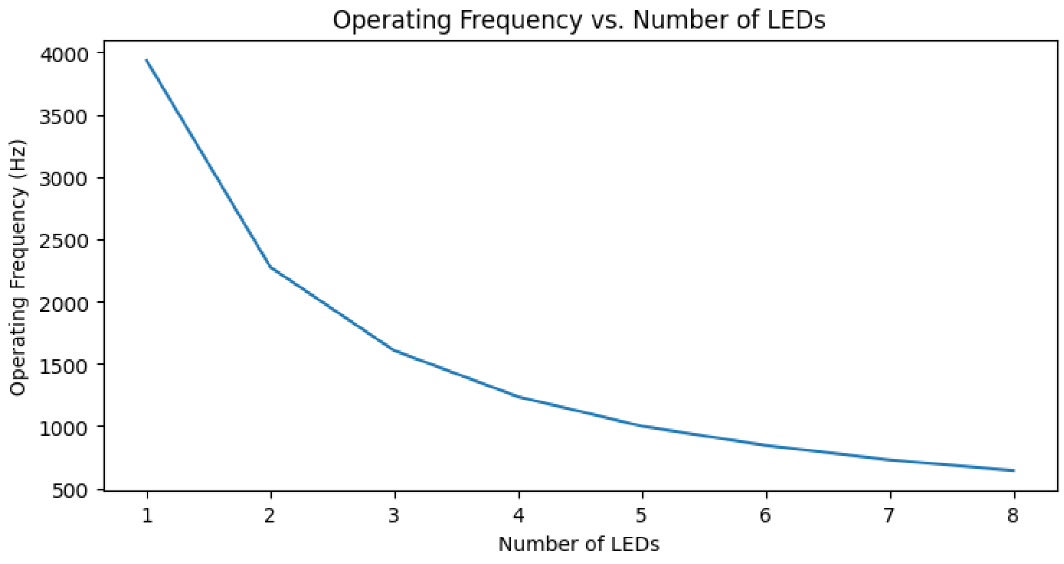


Figure 23: Number of LEDs To Maximum Potential Operating Frequency

If a targetted frequency of 1000Hz is to be achieved, a maximum of 5 LEDs can only be sampled at any given time. The newly implemented sampling method consists of sampling the same LED (LED 4) from both AFE chips at the targetted 1000Hz frequency and samples the remaining LEDs at a 3x lower sampling rate (around 333.3Hz). This is done by sending a sample of 4 LED values at every transmission iteration, 2 of which contain the LED 4 values of both chips and the remaining 2 containing LED 1-3. This can be depicted by the modified flowchart shown in Figure 24. This cyclic transmission pattern must also be captured within the serial monitor output. It is worth noting that 4 LEDs are used since 5 LEDs have a maximum operating frequency almost identical to 1000Hz. Leaving a slight margin to ensure consistent and robust sampling is typically much safer.

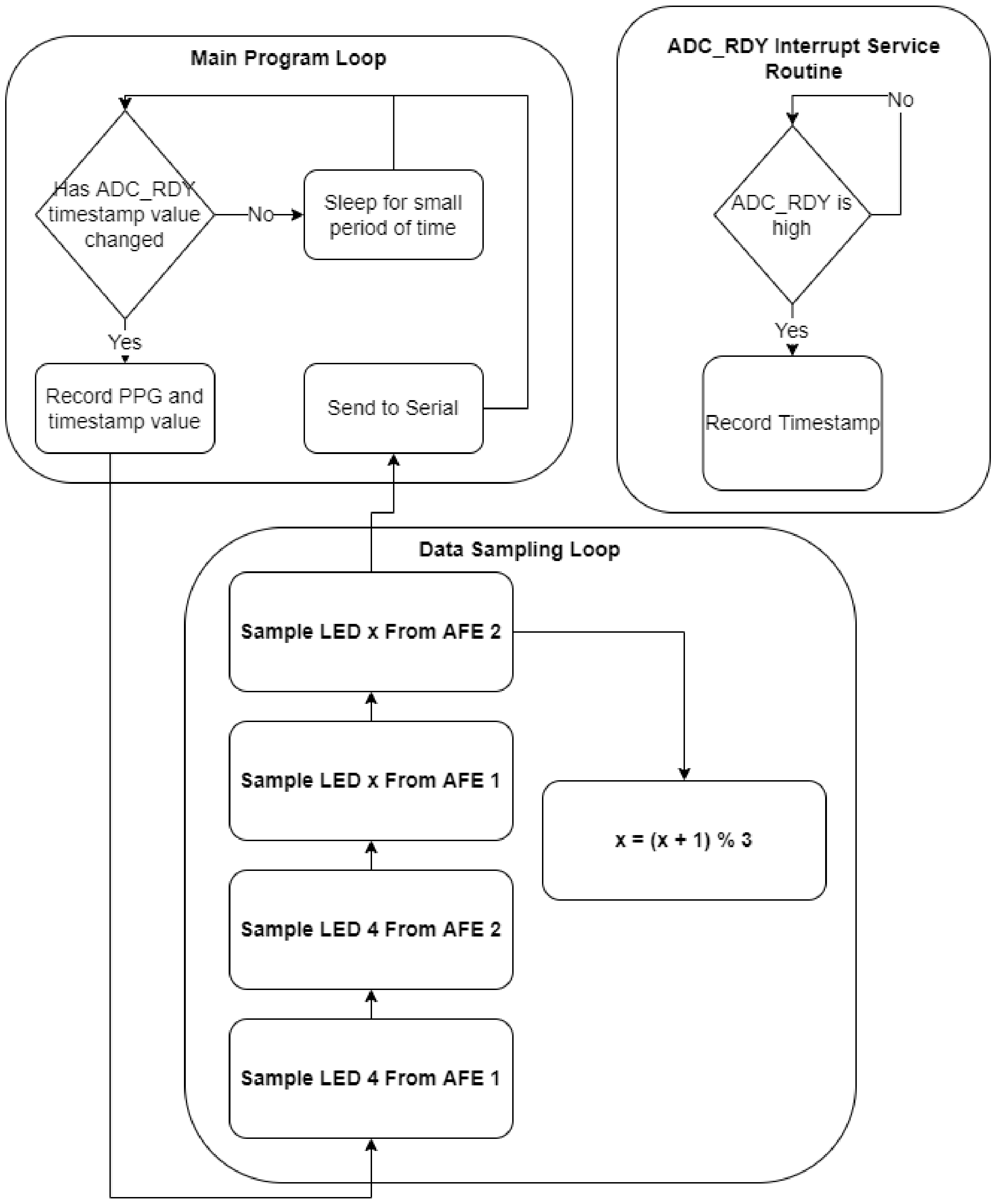


Figure 24: Number of LEDs To Maximum Potential Operating Frequency

5.4.6 Evaluating The New Sampling Method

Using this new method and a sample of **213304** data points, the mean consecutive timestamp between each timestamp difference is now approximately $9.77 \times 10^{-}4$s which gives approximately an operating frequency of 1023.1Hz. Additionally, the distribution of the different timestamp differences and their corresponding operating frequency is shown in Table 8. From the data, this shows that approximately **99.97**% of datapoints operate at a delay of $9.8 \times 10^{-4} \pm 0.1 \times 10^{-4}$ which achieves the estimated **1022.6**Hz (approximately 1000Hz).

Time (s)	Quantity (N)	Estimated Frequency (1.d.p) (Hz)	Percentage of Sampled Data (%)
9.8×10^{-4}	163809	1020.4	76.80
9.7×10^{-4}	49438	1030.9	23.18
0.0	54	NA	2.5×10^{-2}
9.5×10^{-4}	1	1052.6	4.7×10^{-4}

Table 8: Table of the difference between consecutive reads and quantity (213,304 samples)

With the implementation of this new algorithm, the microcontroller can be seen to achieve the desired 1000Hz sampling as well as provide very consistent performance. Furthermore, this algorithm can be tuned further, taking into account the minimum sampling each biomarker needs from each LED wavelength and optimising the best trade-off as a medical wearable. For example, currently, every transmission packet contains 4 LED values and 2 timestamp values. This can be further reduced to 3 and, using the LED to operating frequency graph, be estimated to have an operating frequency up to 1500Hz. This will, however, come at the cost of the other LEDs being further downsampled, which may still be appropriate depending on the biomarker.

5.5 Wrist Sensor & Final Results

With all the optimizations implemented, the targetted operating frequency of 1000Hz has been achieved. A wrist configuration has also been implemented, which will be used to test with the finger probe. A final sanity test is done by applying 2 finger probes with the same configuration to get the results shown in Figures 25 and 26.

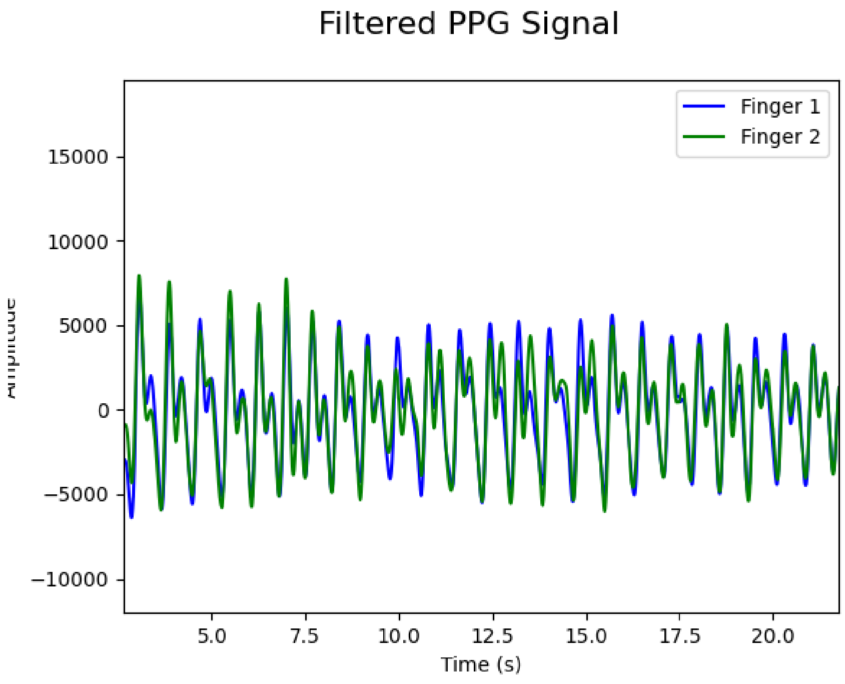


Figure 25: Finger + Finger Configuration Output (Zoomed out)

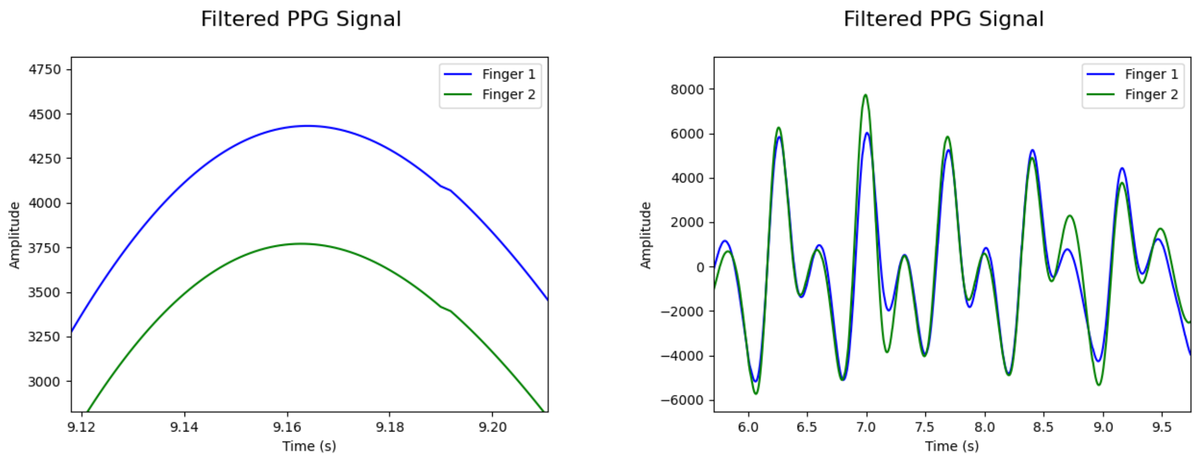


Figure 26: Finger + Finger Configuration Output (Zoomed in)

From the plot, an approximately similar signal can be seen within the two waveforms. Upon closer inspection, it can be seen that sometimes 1 signal can be slightly ahead of the other. This is most likely due to noise artefacts which is a common problem faced by PPG signals. Given a large enough sample, through the use of average, the noise should theoretically become more and more negligible given the relationship in (12) where σ^2 is the noise variance and N is the number of samples.

$$\frac{\sigma^2}{N} \tag{12}$$

Additionally, by manually calculating each delay between peaks in the waveform, there is an estimated -4.66×10^{-3}s mean delay between each waveform showing. By swapping the probe orders, a reverse sign of $2.67 \times 10^{-}3$s delay is measured, showing a potential propagation delay depending on the finger probes used. This could be because one of the finger probes uses a significantly much longer wire than the other. This, however, can be mitigated by the use of a simple constant calibration offset. It is worth noting that this change in sign could still be due to the previously specified noise artefacts, but it is still worth specifying.

By applying a similar sanity test using the wrist and finger configuration, we obtain the plot shown in Figure 27. A constant delay between the waveforms is observed, showing very promising results. There is unexpected behaviour, however, in the PPG waveforms shown between the finger and wrist probes, showing an earlier peak on the finger waveform before the wrist one. This is unexpected as it would be assumed that the heartbeat to the finger would need to pass through the wrist first. However, a likely explanation for this is that the wrist probes are placed where it detects the reflected heartbeat (i.e., the one going from the hands towards the wrist).

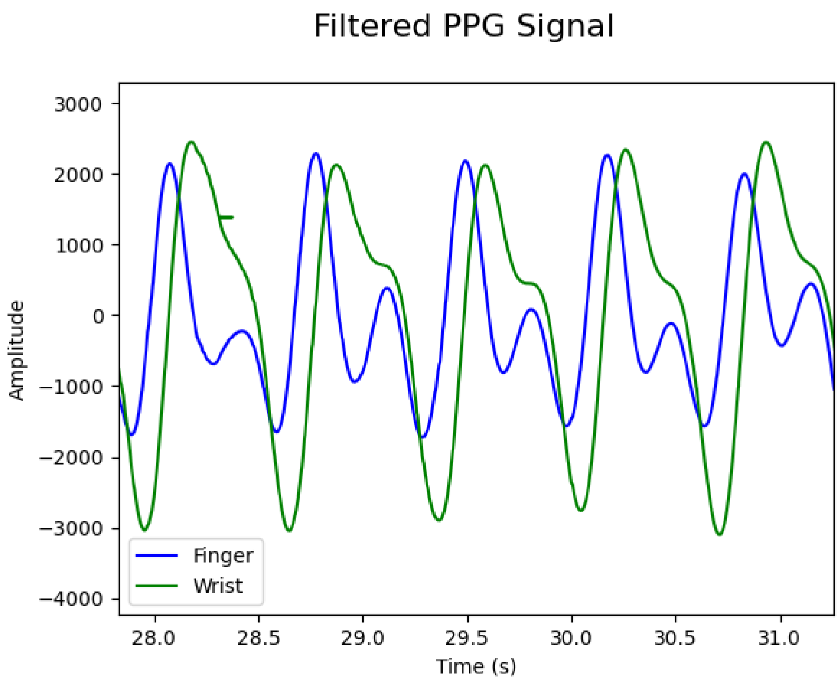


Figure 27: Finger + Wrist Configuration Output (Zoomed in)

Additional factors are also considered. Initially, propagation delay was ruled as a potential cause; however, as the delay is quite large, with results up to almost 0.1s, this is most likely not propagation delay. Another factor considered was also the reflective vs transmissive PPG acquisition properties each probe has. Additionally, the wrist probe uses green light, which has a relatively shallow penetration length compared to the red light the finger probe uses. Ultimately, however, the similarly constant delay between the wrist and finger configuration set-up in comparison to the approximately similar-looking waveforms between the finger and finger configuration shows interesting promise in being able to estimate the blood pressure of patients using this method. Therefore, the task of implementing a robust embedded system with the capabilities of monitoring crucial patient biomarkers with high enough precision and performance is successful.

6 Signal Processing

As the main goal of PPG signal acquisition has been achieved, additional work has been done to improve the signal processing pipeline. This work can be broken down into 2 sections, a peak detection algorithm implementation and attempting to perform filtering using the micro-controller (instead of a computer).

6.1 Peak Detection Algorithm

Pulse Wave Velocity (PWV) can be estimated by calculating the difference in the peaks between the wrist and finger probes using the formula $PWV = Distance/Time$. Therefore a simple peak detection algorithm was implemented to automate the process of peak detection for blood pressure estimation. This peak detection algorithm would be used after filtering since this would give the algorithm the best chance of success at detecting peaks.

The work in [39] was used as an initial starting reference in implementing a simple peak detection algorithm. The idea used was that a peak is considered an extreme value (outlier) when considering the context of the local window around it. Therefore utilising the common range of heart rate values should theoretically allow signal peaks to be detected fairly accurately, given an appropriate window size is chosen. To do this, let m and s denote the mean and standard deviation of the data points within the chosen window. Afterwards, the method uses 2 assumptions in determining whether a sample point x_i is a peak.

1. The sample point x_i is a peak if x_i is greater than the mean around the window (i.e., $x_i \geq m$) and its absolute mean subtracted value is greater than h standard deviations ($|x_i - m| \geq hs$). An example is, if the dataset is normally distributed then $h = 3 \rightarrow P[-3s < x_i - m < 3s] = 0.997$).

2. The second assumption is using the Chebyshev Inequality. For a random variable X with mean m and standard deviation s, for any positive number h, $[P|X - m| < hs] \geq 1 - \frac{1}{h^2}$. This can be rearranged in the form $P[|x_i - m| \geq hs] < \frac{1}{h^2}$.

A problem with these assumptions, however, is that since these assumptions use a range of values, multiple peaks within a small window can be detected. Additionally, this makes the parameter h and window length N difficult to select and can result in either no peaks detected or multiple peaks shown in Figure 28.

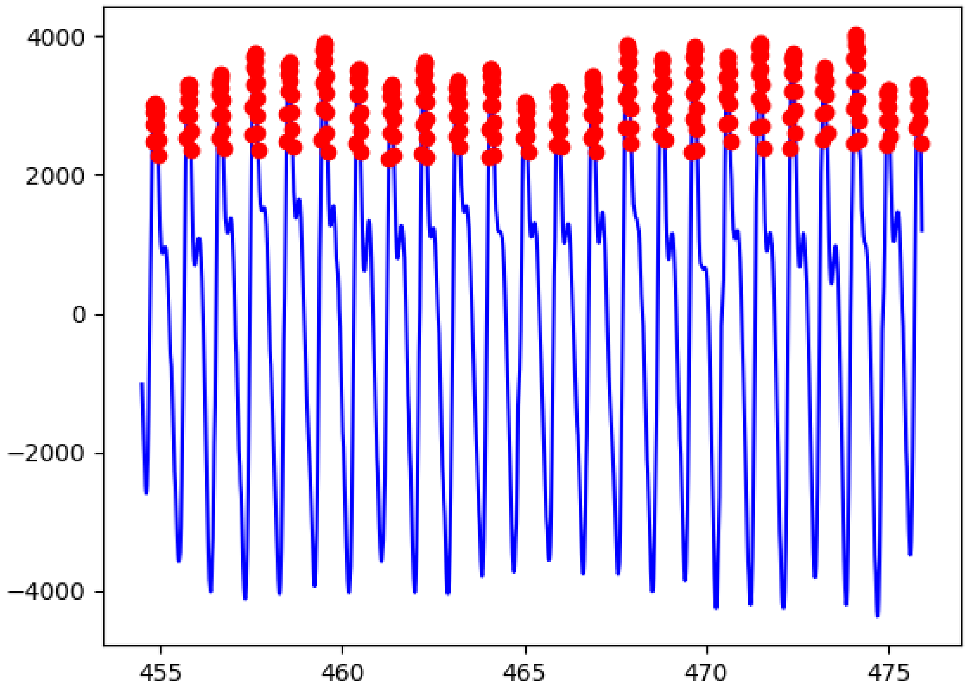


Figure 28: Initial Peak to Peak detection algorithm results (h = 1, window = 2s, fs = 100Hz)

By looking at the waveform, it is obvious why multiple peaks are detected. Given that the signal point x_i satisfies both conditions specified above and that a good window is provided, there will be a range of values near the peak, which will be detected as peaks. This can be visualised in Figure 29:

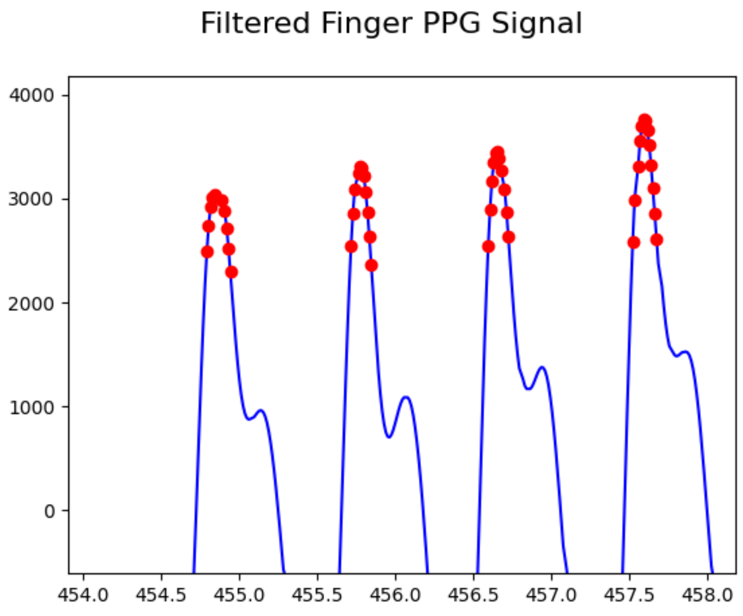


Figure 29: Peak to Peak detection algorithm results zoomed (h = 1, window = 2s, fs = 100Hz)

A simple solution to mitigate this is to use a condition to check whether the local peak detected is still increasing and, if so, update the peak detected data. If it has decreased, however, this indicates that the signal peak for that local window has now been passed and can be recorded. This is only possible due to the smoothing effect of the Chebyshev filter, which emphasises the crucial requirement to filter the signal before passing it through the peak detection algorithm. A flowchart visualisation of this can be seen in Figure 30 as well as the results in Figure 31.

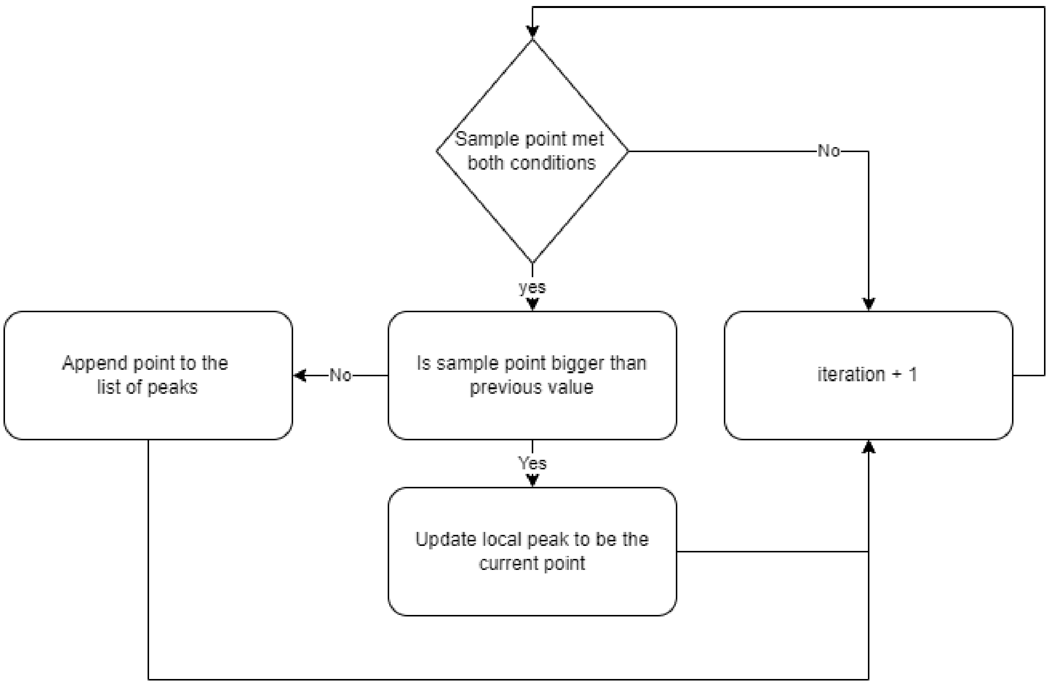

Sample point met both conditions
No
yes
Append point to the list of peaks
No
Is sample point bigger than previous value
iteration + 1
Yes
Update local peak to be the current point

Figure 30: Finalised peak detection flowchart

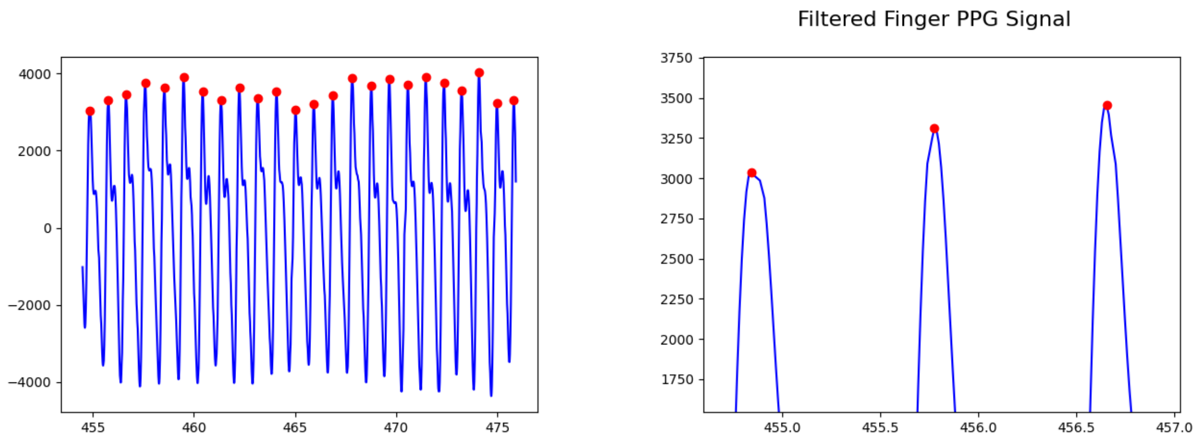


Figure 31: Modified peak detection algorithm results zoomed

Furthermore, by upscaling to 1000Hz and using a finger-finger configuration dataset, we get promising results as shown in Figure 32.

The algorithm proposed clearly shows very good consistent accuracy given the dataset ranges provided. Although the method can suffer from a wide range of variability of heart rate values, a calibration period can be implemented in conjunction with this method to estimate the appropriate window function needed. Provided a calibration period can be implemented, this

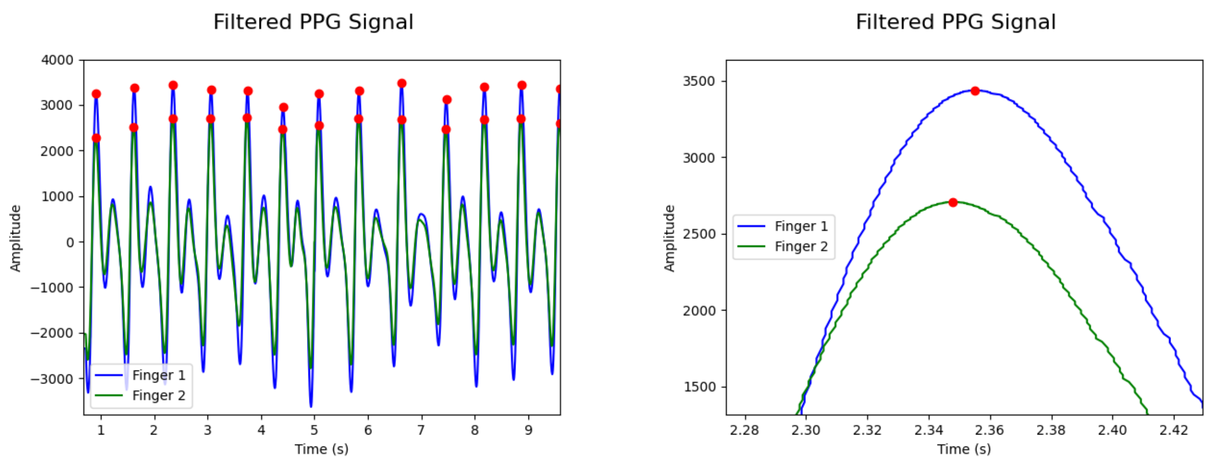


Figure 32: Modified peak detection algorithm results zoomed

method could provide very good results in peak detection even with the variable range of heart rate values.

6.2 Signal Processing within the Microcontroller

6.2.1 C Code vs Python Implementations

After a simple peak detection algorithm was implemented, the next investigation conducted was whether the signal processing algorithm could be implemented within the microcontroller itself. This is because, currently, the program outputs a minimum of 256000 bits per second into serial (4xLED + 2xTimestamps). As investigated previously, this puts a big strain on the microcontroller's performance. Therefore the idea of this investigation is to determine whether the signal processing pipeline (especially for blood pressure estimation) can be done within the microcontroller instead of sending all the data to serial to be processed. This means the microcontroller can transfer a downsampled amount (for example, at 100Hz) for signal visualisation and can additionally transfer the signal features such as timestamp peaks of each signal or even the Pulse Wave Velocity itself. Assuming a regular heart rate of 60bpm, alongside a downsampled 100Hz, this would result in 25696 bits per second which is almost 10 times smaller than the original signal transfer.

So far, the signal processing work done in the project has been used in conjunction with the Scipy library inside Python. This library allows the use of easy-to-use functions which correspond to the different filters used thus far. Being able to transfer this into the microcontroller requires the implementation of this in C code from scratch requiring significantly more work to be able to do. The work done by Disi A in [40] alongside the derivations in Appendix B has been crucial resources in implementing the Butterworth and Chebyshev type I filters that can be used within the microcontroller. Upon testing the Chebyshev type I filter alongside a C-Code version implementation of the peak detection on the pre-saved 100Hz dataset in Figure 16 we can see its corresponding output plot in Figure 33. Its corresponding Chebyshev type II

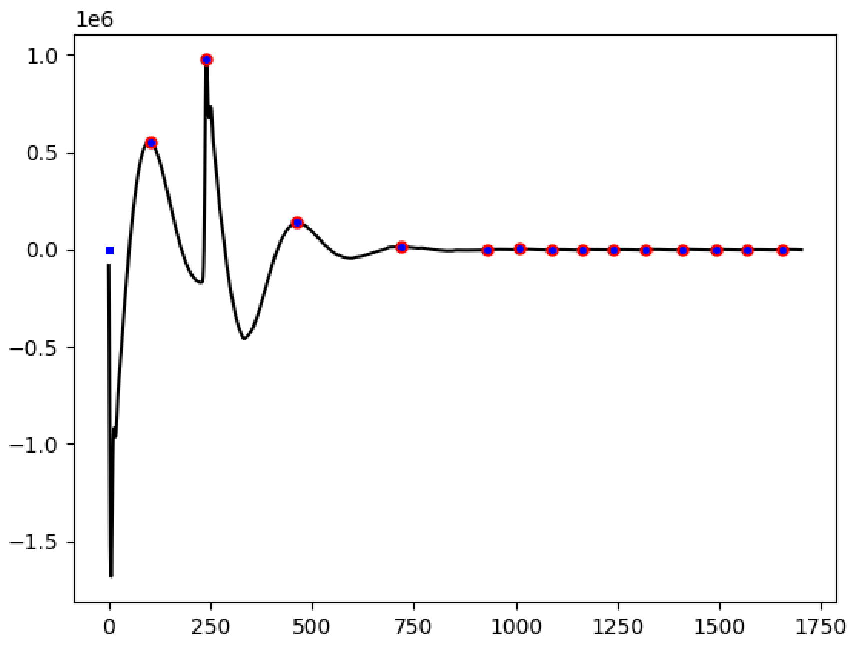


Figure 33: C Code Chebyshev type I filtering (red dots - C Code version peak detection, Blue dots - Python version peak detection)

plot, seen in previous sections, is shown in Figure 34.

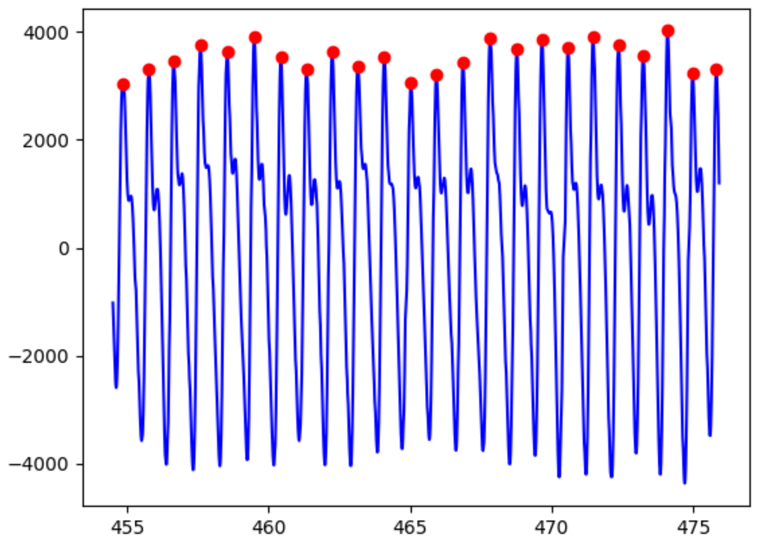


Figure 34: Python version Chebyshev type II sample comparison

From the plots, it can be seen that a big difference is that the C Code filter needs to take some time to converge, which is expected since an IIR requires the feedback loop of previous output values in order to update its weights. This behaviour is not observed when using the Python implementation, as the signal processing library most likely uses different techniques to mitigate this. This is, however, not an issue with the medical wearable as within a couple of

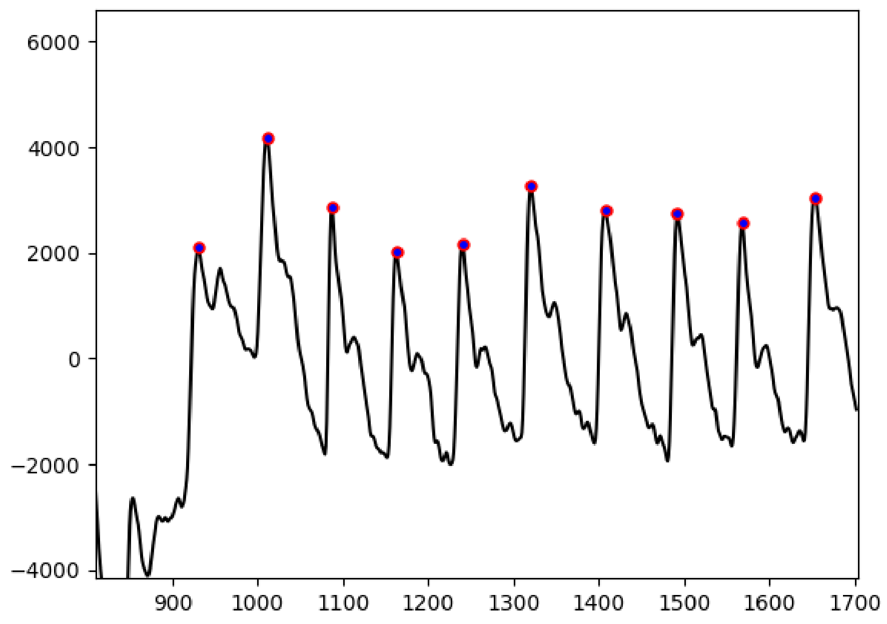


Figure 35: Zoomed in C Code Chebyshev type I filtering (red dots - C Code version peak-to-peak detection, Blue dots - Python version peak-to-peak detection)

seconds, the C Code filter appears to converge, shown in Figure 35. Additionally, as the medical wearable will be monitoring patients over a long period of time, the filter should similarly be able to converge in a short period of time.

Upon closer inspection, it can also be seen that the filtered PPG waveform using the Chebyshev type I C-Code filter contains more noise artefacts in comparison to the filtering done using the Chebyshev type II filter in Python. Looking at the frequency response plots of each filter in Figure 36 there is an observed ripple effect on each of the Chebyshev filter responses. In the type I filter, the observed ripple effect is primarily in the pass-band frequencies whereas in the type II filter, the ripple effect occurs primarily in the attenuated frequencies. This most likely explains the additional noise artefacts upon using the Chebyshev type I filter in comparison to the type II.

Lastly, upon using the same parameters on the Butterworth filter apart from the filter order (n=5 since this provided better results), the corresponding plot output can be seen in Figure 37. Increasing the filter order by 1 provided better results and faster convergence since a higher filter order results in a faster drop-off in the Butterworth filter frequency response. Additionally, from the plot, similar to the Chebyshev type I filter, noise artefacts appear more present, although the same parameters are used. From its frequency response plot, it can be observed that the Butterworth filter has a much slower frequency cut-off drop than the other two filters. This results in more frequency noise contents ”leaking” into the pass-band since the power of those frequency components are not attenuated to a low enough value.

A solution that was implemented to mitigate this is to use tighter bounds when filtering the

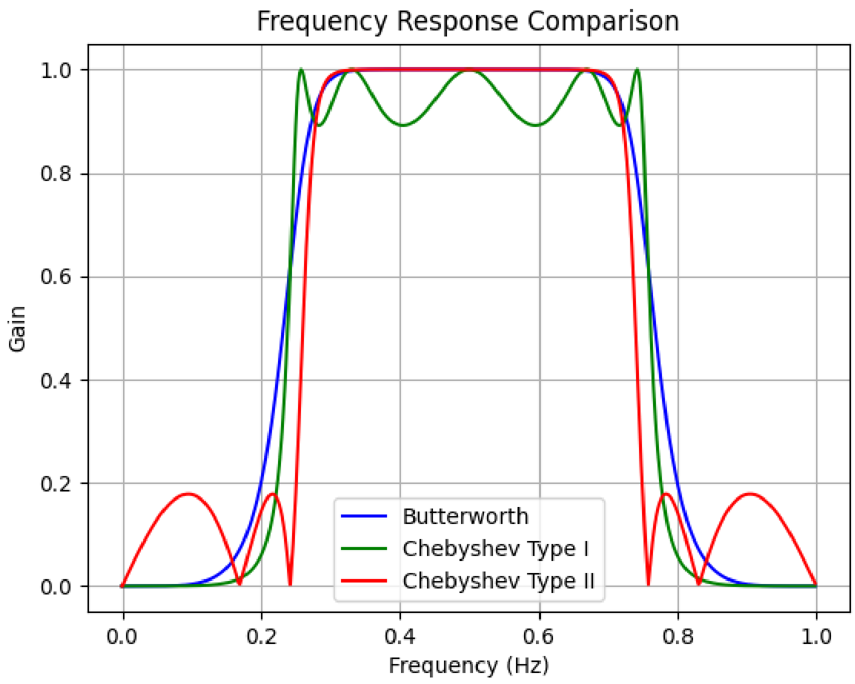


Figure 36: Example visualisation of different bandpass filters with cutoff frequency 0.25-0.75Hz (Parameters: Cheby1 - [n=5, rs = 1], Cheby2 - [n=5, rs = 15], Butterworth - [n=5]

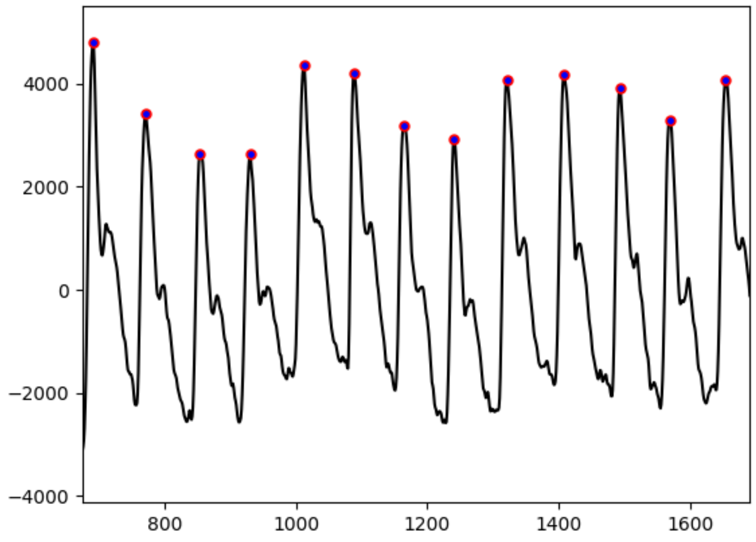


Figure 37: Butterworth Filtered signal with original parameters (n=5, highpass cutoff = 0.35, lowpass cutoff = 10)

signal. This should, in theory, provide similar attenuation to the unwanted frequency components. Although not an ideal solution, Figure 38 shows a better-filtered signal with visual improvements on the ”jaggedness” nature of the signal. Since this appeared to provide the best visual filtering of the signal, this is the filter of choice with regard to the implementation.

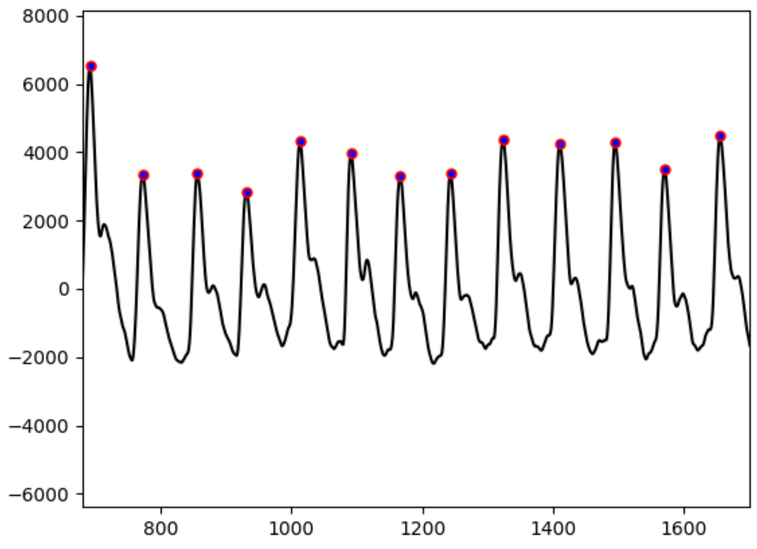


Figure 38: Butterworth Filtered signal (n=5, highpass cutoff = 0.5, lowpass cutoff = 6)

6.2.2 Performance Drawbacks

Both C-Code filter and peak detection versions are ported into the microcontroller and performance tested. Upon testing, the peak detection performed relatively slowly due to its use of floats and double-precision floating points. As stated in [38], there are no double precision floating point hardware units within the ESP32. This means all calculations involving doubles are emulated in software which causes a very significant drop in performance. There is also only 1 floating point hardware unit within the microcontroller. Since the peak detection algorithm heavily relies on floats, this can also cause a performance bottleneck. Thus, executing the peak detection algorithm results in excessive time consumption and continuously triggers the watchdog timer.

On the other hand, the C-Code version of the Chebyshev Type I and Butterworth Filters performs significantly better as it does not rely on doubles when performing calculations. Measuring the average time it takes to perform each task (**10000 samples**) results in the Table shown in Table 9. Although filtering itself does not appear to cause a performance bottleneck, showing an average speed of 2×10^{-6}s per transmission cycle, it appears to cause a significant reduction in reading speeds (17 times slower), potentially due to the use of additional computing resource. Furthermore, as mentioned previously, using floating point arithmetic in calculations can reduce performance speeds. Lastly, some optimisations described in Section 5.4 are also reverted as floating point arithmetic in conjunction with the optimisations are not supported. Some of these include:

1. Removal of the targetted optimisations.
2. Removal of priority tasks.

3. Removal of multi-processing.

	Original Implementation (s)	C-Code Filtering Implementation (s)
Read	6.6×10^{-4}	1.1×10^{-3}
Filtering	N/A	2.0×10^{-6}
Write	1.47×10^{-4}	1.5×10^{-4}

Table 9: Table of Time Taken to Perform Each Task (s) Between Original and C-Code Filtering Implementations (10000 samples)

6.2.3 Evaluation and Further Improvements

Using both filtering and peak detection within the microcontroller is currently not feasible without further optimisations due to the heavy performance drawbacks it could cause on BP estimation. However, the C Code implementations can be substituted for the current Python version as it performs significantly faster, increasing the current system's performance speeds. Additionally, suppose certain biomarkers that require very high precision, such as BP estimation, are excluded from the system (as they might not be needed for other disease monitoring). In that case, this means the integration of filtering within the microcontroller serves as an ideal choice to improve the computational and power efficiency of the system. Ultimately, the decision to migrate some of the signal processing within the microcontroller relies on the system's performance requirements.

If the system proposed in this subsection can be optimised further, then this could open up the possibility for a highly advanced and very efficient medical wearable. Optimising the functions to use 64-bit integers instead of doubles and floats could allow both the peak detection and filtering algorithms to perform better at BP estimation. Additionally, some additional work can be done to mitigate the filters' initial transient behaviour and allow them to converge quicker. Lastly, from the frequency response plot, it is clear that a more optimal filter than both filters used in this section is the Chebyshev Type II Filter. The filter's fast drop-off and flat pass-band characteristics make it the most optimal among the 3.

7 Additional Development and Implementation

This section explains the additional work performed in addition to the main core and signal processing work mentioned above. Similar to Section 6, this section aimed to add some potential quality-of-life improvements to the medical wearable and attempt to get it closer to a fully working product.

7.1 Temperature Sensing

From the clinical review of managing patients with dengue fever [41], temperature changes in patients' current body temperature changes typically vary depending on the stage of the illness. During the early stages of Dengue fever, a common symptom that typically manifests in patients is increased body temperature. This symptom is typically remedied with the use of paracetamol with no more than four doses in 24 hours. However, complications in dengue fever can arise, and this can sometimes result in dengue haemorrhagic fever, which is a more severe case of the illness. As mentioned previously, this severe case of dengue typically results in plasma leakage and decreased blood volume. An additional symptom accompanied by this, however, is also a sudden decrease of body temperature or hypothermia. Tracking the extremities of patients should increase robustness in monitoring patients with severe dengue fever.

7.1.1 TMP117 Library

The TMP117 is a high-precision digital temperature sensor with characteristics to meet ASTM E1112 and ISO 80601 requirements for electronic patient thermometers. The next objective was to utilise this device to provide additional patient data by creating an additional embedded application. The Adafruit implementation for the Arduino TMP117 library [42] is used as a starting reference for useful functions to implement and port/address values when creating the library. Similar to the AFE library, additional helper functions and data types would also have to be created.

All the function signatures defined in the header file take in TMP117_cfg parameters. The TMP117_cfg is simply an extra datatype created for the purpose of storing the configuration handler for the specific TMP117 instance. The data structure definition, as well as an example of how this can be instantiated, is shown in Appendix C.1 Listings 2 and 3.

7.1.2 Testing

Testing the TMP117 library was done incrementally by testing all functions individually, progressing from the least dependent function to the most dependent. As almost every function in the library was dependent on the basic read and write functions, this, alongside the instantiate function, was tested first. The serial output in Appendix D Listing 6 shows the sample output at 100Hz of the temperature sensor with default configuration values and under room conditions.

7.1.3 Integration with ADXL343

After the TMP117 library was confirmed to be working, this was integrated with an ADXL343 sensor. The ADXL343 is a 3-axis accelerometer with the purpose of improving the D-SCAPE power efficiency. By monitoring whether the patient is moving, the wearable can determine whether it should or should not record data. Since a moving patient will most likely provide PPG signal data with noise artefacts, the wearable could either, not record signals at any point the patient is moving or use the accelerometer data to remove noise artefacts.

The two sensors are integrated together by allocating the two I2C interfaces of the ESP32-S3 to each sensor. A sample log output of the two sensors working simultaneously at 100Hz is shown in Appendix D Listing 7.

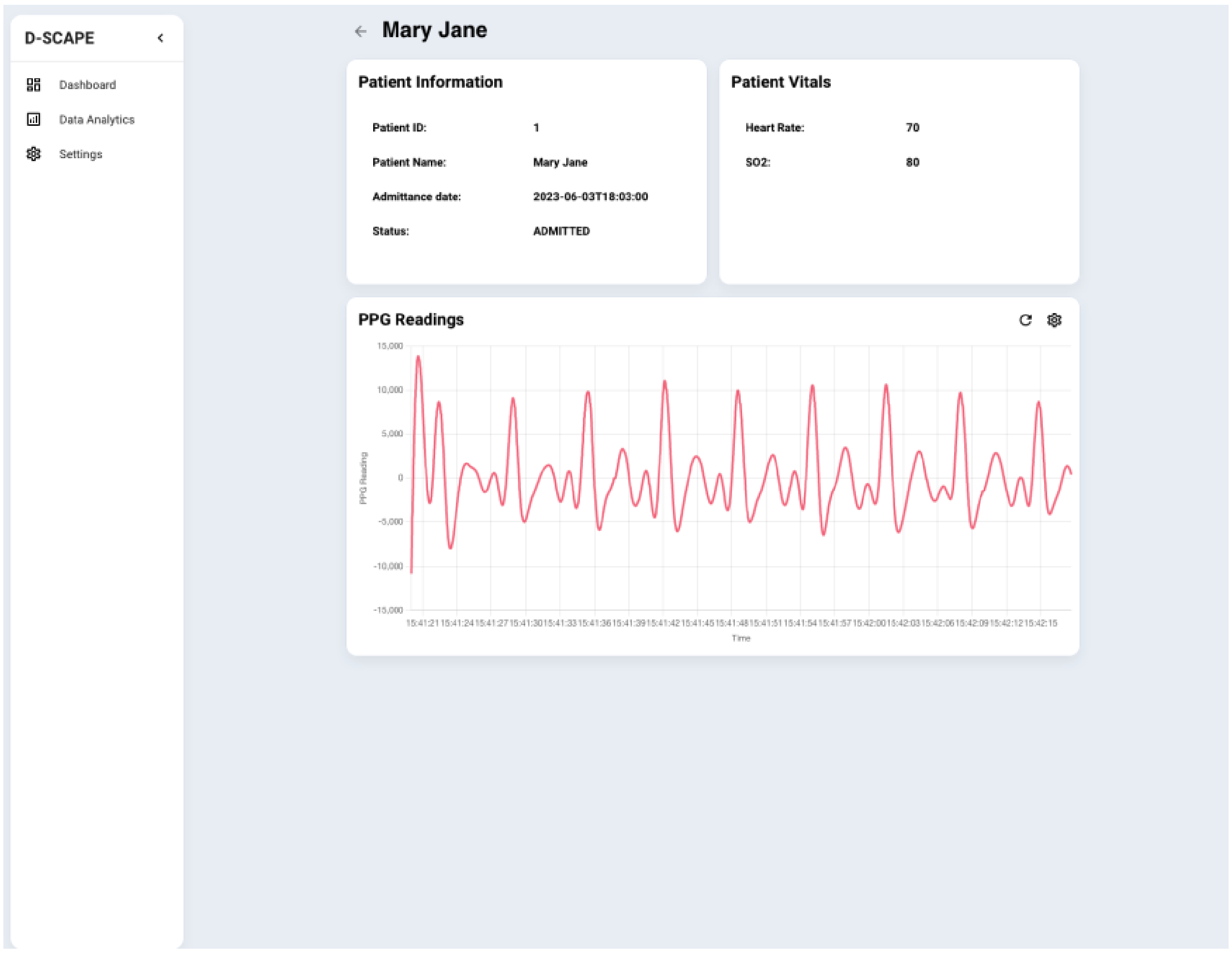
Although these additional sensors appear to increase the additional overhead of the D-SCAPE wearable, these signals will be sampling at a much lower rate than the AFE chips. This means the sensors will most likely increase the efficiency and sensing capabilities of the D-SCAPE.

7.2 Integration with System Design

As stated at the beginning Sections of this report, the D-SCAPE-v2 is a multi-part project consisting of multiple sections, with this report primarily focusing on the embedded system and signal processing implementations. Xin Wang has created another project consisting of the system design implementation of the D-SCAPE-v2 under the same primary Supervisor, Prof. Georgiou P. As a short summary, this system design implementation consists of a web app to which the D-SCAPE wearable will be connected and contains numerous tools and features that easily allow patient monitoring from a computer system.

The system design implementation is primarily optimised for data transmission through a wireless medium (WiFi). In comparison, the embedded system application described in this work is more heavily tailored towards serial monitoring. Therefore, this last implementation addition consists of attempting to combine both projects together with the aim of getting a minimum viable product that can perform PPG acquisition and can be monitored through the web app.

The integration objective was to transmit the PPG waveforms using WiFi, which would be verified by the real-time waveform seen through the web app. After some code modifications, the final product can be seen in Figure 39 and 40. This version of the D-SCAPE-v2 was entered in the ISCAS 2023 competition under the contestants: Khayle Torres, Xin Wang and Yuting Xu and won regionally (Region 8 - Europe, Middle East, Africa). The video and report submitted can be seen in Appendix A



Mary Jane

Patient Information

Patient ID:	1
Patient Name:	Mary Jane
Admittance date:	2023-06-03T18:03:00
Status:	ADMITTED

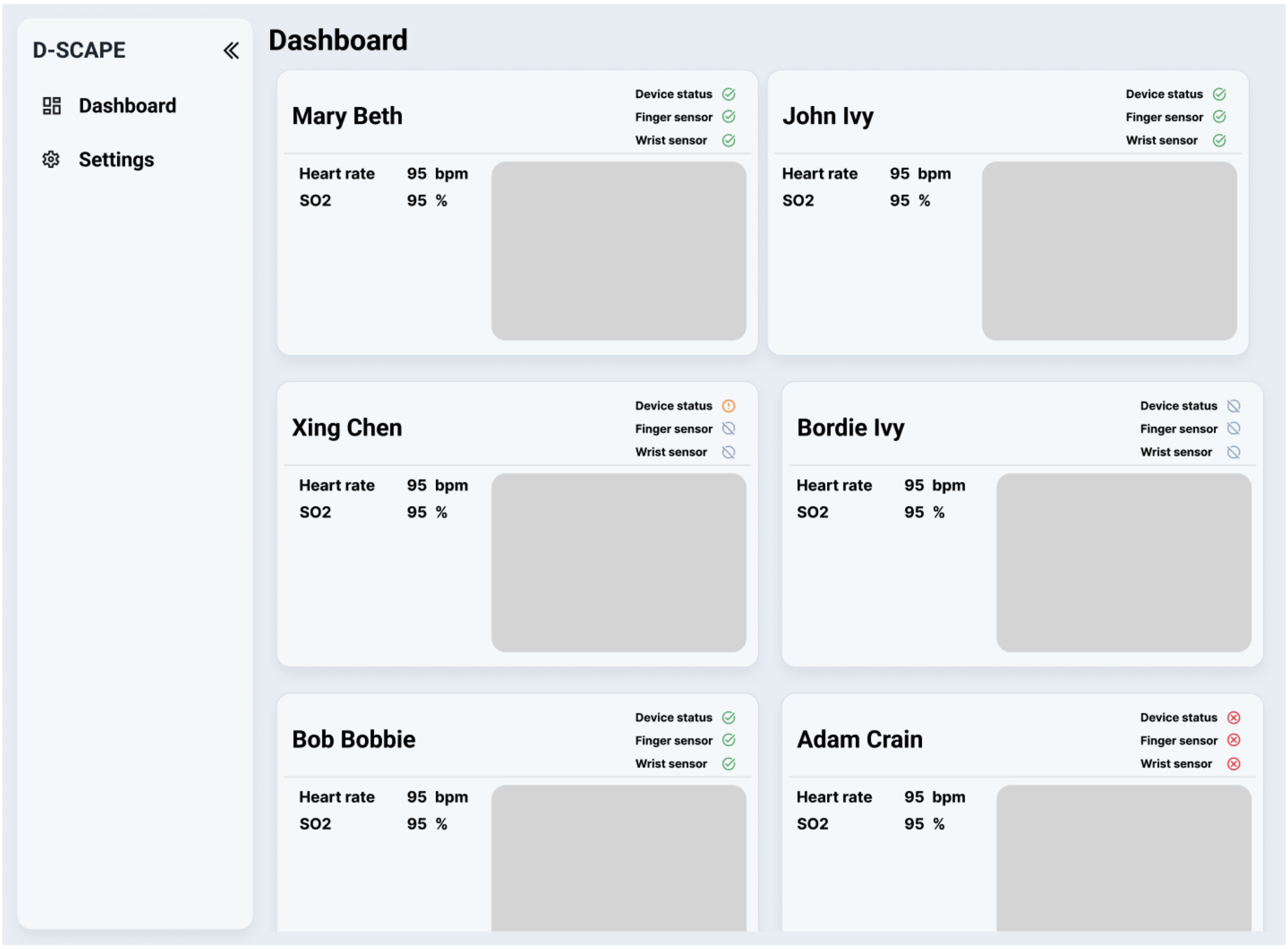
Patient Vitals

Heart Rate:	70
SO2:	80

Figure 39: Integrattion Web App (1)

7.2.1 Brief Integration Evaluation

Although PPG waveforms can be observed through the front end, this was done under heavy time constraints in addition to the regular project workload. As such, this has not been thoroughly tested. The operating frequency, primarily tested and used under the demo in the ISCAS 2023 competition, had been at **100Hz** and would occasionally experience reductions in throughput. This was primarily seen occasionally through a continuously increasing delay between the PPG recording and the visual signal output waveform on the web app. This typically happens under **non-optimal conditions** such as using slower and less reliable mobile data instead of WiFi. This appears to indicate a performance bottleneck in the process of sending packets. Further testing must be conducted to investigate the impact of unreliable connections on the speed of data transmission and different ways to mitigate this.



Dashboard

Mary Beth — Device status ✓, Finger sensor ✓, Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

John Ivy — Device status ✓, Finger sensor ✓, Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Xing Chen — Device status !, Finger sensor ⊘, Wrist sensor ⊘
Heart rate 95 bpm
SO2 95 %

Bordie Ivy — Device status ⊘, Finger sensor ⊘, Wrist sensor ⊘
Heart rate 95 bpm
SO2 95 %

Bob Bobbie — Device status ✓, Finger sensor ✓, Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Adam Crain — Device status ⊗, Finger sensor ⊗, Wrist sensor ⊗
Heart rate 95 bpm
SO2 95 %

Figure 40: Integrattion Web App (2)

This is, however, to be expected as a transition from a Serial-based communication link to a WiFi one typically slows down the rate of transmission and decreases reliability. Additionally, since the rate of transmission while using WiFi is variable under the speed of the connection, this can serve as a critical bottleneck in the signal processing pipeline. For future development, an integration using the same web app, however, using Serial instead of WiFi should allow for more consistent real-time tracking and monitoring of critical patient biomarkers.

8 Conclusion

At the beginning of this work, we set out to develop a real-time embedded system capable of improving and solving some of the issues the original v1 had. By implementing the numerous optimisation techniques investigated and discussed above, the v2 can reach higher performance than its predecessor, improving upon its PPG-based sensing capabilities and increasing robustness.

The enhancements implemented in version 2 of the device enable more accurate estimation of patient biomarkers. They are expected to substantially address the common challenge of blood pressure estimation faced by PPG-based sensors. This improvement holds considerable promise for enhancing overall performance. Furthermore, the inclusion of an embedded system framework will enhance the capabilities and potential for expanding the functionalities and features of the system. This framework opens up new avenues for future development and growth.

In addition, this work also investigated signal processing techniques aimed at extracting a very useful PPG signal feature used for blood pressure estimation and improving the wearable's efficiency. The peak detection algorithm showed very good promise in consistently detecting the peaks of signals within a common range of heart rates. Although the investigation upon integrating signal processing techniques within the microcontroller was concluded too slow to implement, given this can be improved and optimised for performance, there might be some promise in potentially integrating this within the microcontroller.

Lastly, this work also provides further improvements, such as integration with a system design architecture and the addition of a temperature sensor library which can be used to build upon for future iterations of the medical wearable. This allows the D-SCAPE-v2 to become a complete medical wearable that can be used in clinical facilities.

8.1 Further Development

As the research performed on the D-SCAPE is quite novel, the potential for further improvement is very large. Although some of these have been mentioned throughout the report, this section will explain more concisely some potential improvements which can be implemented. for future versions.

Optimization of Signal Processing For The Microcontroller: One area of future work is to explore and optimize the signal processing techniques implemented in C-Code up to a stage that can be efficiently used within the microcontroller. This would involve streamlining the algorithms and reducing computational complexity while maintaining the accuracy of the PPG measurements.

Integration of Accelerometer and Temperature Sensors: Another avenue for future work is the integration of the additional ADXL343 and TMP117 sensors to be used with alongside the AFE4900, previously discussed in Section 7.1. This integration aims to optimize power consumption and enhance the accuracy of biomarker readings. An example stated previously is utilizing the accelerometer data in order to either remove noise artefacts within the data or stop recording while the patient is moving. Additionally, incorporating temperature readings from extremities could provide valuable contextual information. By integrating these sensors effectively, the overall performance and reliability of the PPG system can be improved.

Implementation of Adaptive Signal Processing for PPG Acquisition: Throughout the development of this project, it has been observed that the optimal LED intensity values between individuals can encompass a wide range of values. Some of these factors, such as skin colour, hair density, skin density, and nail colour, have been observed to affect signal readings significantly and have required the tuning of LED intensity values in order to get valid PPG signals. Future iterations of the D-SCAPE system could therefore incorporate adaptive signal processing techniques as a part of its signal processing pipeline. Currently, the intensity values for PPG measurements are constant. However, the system can provide improved PPG readings by developing algorithms that can automatically tune the ideal intensity values based on individual patient characteristics. This adaptive signal-processing approach can potentially enhance the accuracy and reliability of PPG measurements across a diverse range of users.

8.2 Final Remarks

Throughout the course of this project, I have acquired invaluable insights and hands-on experiences that helped me better understand the complexities of developing a wearable medical device. A notable key takeaway from this revolves around the challenge posed by the necessity to perform additions while ensuring the device's robustness, performance, and efficiency are not compromised. Developing solutions in a resource-constrained environment requires careful consideration of trade-offs and prioritization to ensure optimal functionality within limited resources.

Undoubtedly, the most significant challenge encountered was the realization that initial implementations often fell short of expectations. It became apparent that finding creative and innovative solutions was essential to overcome the hurdles encountered along the way. This iterative process of trial and error has been crucial in finding successful working solutions for this project.

Finally, through the challenges faced and the determination to overcome setbacks, I have developed a deeper understanding and appreciation of the complexities inherent in developing a successful medical wearable device.

References

[1] Caitlin AM Van Dodewaard and Stephanie L Richards. “Trends in dengue cases imported into the United States from Pan America 2001–2012”. In: *Environmental Health Insights* 9 (2015), EHI–S32833.

[2] WHO. *Dengue and severe dengue*. Jan. 2022. URL: https://www.who.int/en/news-room/fact-sheets/detail/dengue-and-severe-dengue (visited on 01/23/2023).

[3] Trinh Manh Hung et al. “Productivity costs from a dengue episode in Asia: a systematic literature review”. In: *BMC Infectious Diseases* 20.1 (2020), pp. 1–18.

[4] Stefan Karolcik et al. “A multi-site, multi-wavelength PPG platform for continuous non-invasive health monitoring in hospital settings”. In: *IEEE Transactions on Biomedical Circuits and Systems* (2023).

[5] Junyung Park et al. “Photoplethysmogram Analysis and Applications: An Integrative Review”. In: *Frontiers in Physiology* 12 (2022), p. 2511.

[6] Denisse Castaneda et al. “A review on wearable photoplethysmography sensors and their potential future applications in health care”. In: *International journal of biosensors & bioelectronics* 4.4 (2018), p. 195.

[7] Jiřı Přibil, Anna Přibilová, and Ivan Frollo. “Comparative measurement of the PPG signal on different human body positions by sensors working in reflection and transmission modes”. In: *Engineering proceedings* 2.1 (2020), p. 69.

[8] Mohamed Elgendi. “On the analysis of fingertip photoplethysmogram signals”. In: *Current cardiology reviews* 8.1 (2012), pp. 14–25.

[9] Yoshifumi Kishimoto, Yasunari Kutsuna, and Koji Oguri. “Detecting motion artifact ECG noise during sleeping by means of a tri-axis accelerometer”. In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2007, pp. 2669–2672.

[10] Reza Rawassizadeh, Blaine A Price, and Marian Petre. “Wearables: Has the age of smartwatches finally arrived?” In: *Communications of the ACM* 58.1 (2014), pp. 45–47.

[11] SM Riazul Islam et al. “The internet of things for health care: a comprehensive survey”. In: *IEEE access* 3 (2015), pp. 678–708.

[12] TI. *Texas Instruments AFE4900*. Jan. 2023. URL: https://www.ti.com/product/AFE4900 (visited on 01/23/2023).

[13] Devin B McCombie, Andrew T Reisner, and H Harry Asada. “Adaptive blood pressure estimation from wearable PPG sensors using peripheral artery pulse wave velocity measurements and multi-channel blind identification of local arterial dynamics”. In: *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2006, pp. 3521–3524.

[14] Charalambos Vlachopoulos, Michael O'Rourke, and Wilmer W Nichols. *McDonald's blood flow in arteries: theoretical, experimental and clinical principles*. CRC press, 2011.

[15] Jehill D Parikh et al. "Measurement of pulse wave velocity in normal ageing: comparison of Vicorder and magnetic resonance phase contrast imaging". In: *BMC cardiovascular disorders* 16.1 (2016), pp. 1–7.

[16] Yadong Li et al. "Research based on OSI model". In: *2011 IEEE 3rd International Conference on Communication Software and Networks*. IEEE. 2011, pp. 554–557.

[17] Frederic Leens. "An introduction to I2C and SPI protocols". In: *IEEE Instrumentation Measurement Magazine* 12.1 (2009), pp. 8–13.

[18] Dvijen Trivedi et al. "SPI to I2C Protocol Conversion Using Verilog". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–4.

[19] Gunther Gridling and Bettina Weiss. "Introduction to microcontrollers". In: *Vienna University of Technology Institute of Computer Engineering Embedded Computing Systems Group* (2007).

[20] Toshiyo Tamura et al. "Wearable photoplethysmographic sensors—past and present". In: *Electronics* 3.2 (2014), pp. 282–302.

[21] Byung S Kim and Sun K Yoo. "Motion artifact reduction in photoplethysmography using independent component analysis". In: *IEEE transactions on biomedical engineering* 53.3 (2006), pp. 566–568.

[22] KW Chan and YT Zhang. "Adaptive reduction of motion artifact from photoplethysmographic recordings using a variable step-size LMS filter". In: *SENSORS, 2002 IEEE*. Vol. 2. IEEE. 2002, pp. 1343–1346.

[23] M Raghu Ram et al. "A novel approach for motion artifact reduction in PPG signals based on AS-LMS adaptive filter". In: *IEEE Transactions on Instrumentation and Measurement* 61.5 (2011), pp. 1445–1457.

[24] Boreom Lee et al. "Improved elimination of motion artifacts from a photoplethysmographic signal using a Kalman smoother with simultaneous accelerometry". In: *Physiological measurement* 31.12 (2010), p. 1585.

[25] Hayato Fukushima et al. "Estimating heart rate using wrist-type photoplethysmography and acceleration sensor while running". In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2012, pp. 2901–2904.

[26] Sang Hyun Kim, Dong Wan Ryoo, and Changseok Bae. "Adaptive noise cancellation using accelerometers for the PPG signal from forehead". In: *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2007, pp. 2564–2567.

[27] Parvez Ahmmed et al. “A wearable wrist-band with compressive sensing based ultra-low power photoplethysmography readout circuit”. In: *2019 IEEE 16th international conference on wearable and implantable body sensor networks (BSN)*. IEEE. 2019, pp. 1–4.

[28] Gabriel Chan et al. “Multi-site photoplethysmography technology for blood pressure assessment: challenges and recommendations”. In: *Journal of clinical medicine* 8.11 (2019), p. 1827.

[29] Juan C Ruiz-Rodrıguez et al. “Innovative continuous non-invasive cuffless blood pressure monitoring based on photoplethysmography technology”. In: *Intensive care medicine* 39 (2013), pp. 1618–1625.

[30] Gašper Slapničar, Nejc Mlakar, and Mitja Luštrek. “Blood pressure estimation from photoplethysmogram using a spectro-temporal deep neural network”. In: *Sensors* 19.15 (2019), p. 3420.

[31] Nabil Ibtehaz et al. “PPG2ABP: Translating Photoplethysmogram (PPG) Signals to Arterial Blood Pressure (ABP) Waveforms”. In: *Bioengineering* 9.11 (2022), p. 692.

[32] Grzegorz Bilo et al. “Validation of the Somnotouch-NIBP noninvasive continuous blood pressure monitor according to the European Society of Hypertension International Protocol revision 2010”. In: *Blood pressure monitoring* 20.5 (2015), p. 291.

[33] Eoin O’Brien et al. “Working Group on Blood Pressure Monitoring of the European Society of Hypertension International Protocol for validation of blood pressure measuring devices in adults”. In: *Blood pressure monitoring* 7.1 (2002), pp. 3–17.

[34] Chern Pin Chua and Conor Heneghan. “Continuous blood pressure monitoring using ECG and finger photoplethysmogram”. In: *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE. 2006, pp. 5117–5120.

[35] Geerthy Thambiraj et al. “Investigation on the effect of Womersley number, ECG and PPG features for cuff less blood pressure estimation using machine learning”. In: *Biomedical Signal Processing and Control* 60 (2020), p. 101942.

[36] Amir Hosein Afandizadeh Zargari et al. “An accurate non-accelerometer-based ppg motion artifact removal technique using cyclegan”. In: *ACM Transactions on Computing for Healthcare* 4.1 (2023), pp. 1–14.

[37] Espressif. *SPI Master Driver*.

[38] Espressif. *Maximizing Execution Speed*.

[39] Girish Palshikar et al. “Simple algorithms for peak detection in time-series”. In: *Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*. Vol. 122. 2009.

[40] Disi A. *C Filters Github*. 2023. URL: https://github.com/adis300/filter-c.

[41] Maria Glória Teixeira and Mauricio L Barreto. “Diagnosis and management of dengue”. In: *Bmj* 339 (2009).

[42] AF. *Adafruit TMP117 Arduino Github*. Jan. 2023. URL: https://github.com/adafruit/Adafruit_TMP117 (visited on 01/23/2023).

[43] Clemens Lode. *Recursive Digital Filters: A Concise Guide*. Abrazol Publishing, 2014.

Appendix A ISCAS 2023

A.1 Youtube Video

The Youtube video submission of the D-SCAPE-v2 is currently **unlisted** however can be found by using this link: "https://youtu.be/8lk5BZSLvvQ"

A.2 Conference Report

This section shows the report that was submitted for the ISCAS2023 competition. This report was a collaborative effort made by the Region 8 team specified on the first page. As each page in the report requires the full page, the first page will be displayed on the next page.

IEEE CASS - Student Design Competition Report
Region 8

A MULTI-SITE MULTI-WAVELENGTH PPG DEVICE FOR DENGUE MONITORING

Wang Xin, Khayle Torres*, Yuting Xu, Stefan Karolcik, and Pantelis Georgiou*

Department of Electrical and Electronic Engineering, Imperial College London

{xin.wang19, khayle.torres19, yuting.xu18, s.karolcik, pantelis}@imperial.ac.uk

ABSTRACT

This report presents a multi-site multi-wavelength PPG device platform capable of synchronous data acquisition and wireless transfer. The device can be used to assist in providing personalised point-of-care (PoC) monitoring to dengue patients in Low- and Middle- Income Countries (LMICs).

1. INTRODUCTION

Dengue poses significant healthcare risks globally, putting 3.9 billion people at risk [1]. One in 20 who contract dengue develop severe symptoms that can be fatal within a few hours [2]. There is no specific treatment for dengue, the WHO advises the best approach is through continuous monitoring of patients to detect the onset of severe symptoms and provide timely clinical interventions such as organ support and fluid therapy. This can reduce the mortality of severe dengue from 20% to less than 1% [3][4]. With approximately 96 million cases of severe dengue recorded per year that require clinical interventions [5], dengue imposes a major strain on healthcare systems, especially on low and middle-income countries (LMICs) [6]. As these healthcare systems get strained, misdiagnosis of dengue patients occurs more frequently [7]. Missing or not identifying the warning signs of severe dengue could result in fatal outcomes for patients [8] and increases the economic costs [9] with one study estimating that dengue in Vietnam alone represents an economic burden of US$94.87 million per year (2016 prices) [10].

To alleviate this issue, the team at Imperial College London's Centre for Bio-inspired Technology has built and successfully validated a portable, laptop-operated multi-site multi-wavelength device [11]. The portable device has been validated in a clinical environment, enrolling 50 dengue patients in Vietnam as part of an observational study. Although this device was well-received by the patients, clinicians, and hospital officials, it has not achieved all its goals, notably the wearable aspect and the ability to non-invasively estimate blood pressure. Therefore, we have built a second prototype of the PPG device that aims to solve the limitations of the first version and transition towards a wearable form factor. In this report, we go into detail on the hardware and software architecture of the device platform.

2. SYSTEM AND STRUCTURE

I. Sensing Methodology

Photoplethysmography (PPG) is a non-invasive, low-cost technique of measuring the physiological state of the body. By shining light onto the skin and measuring the reflected or passed through light, one can reconstruct the changes in blood volume and extract biological marker parameters such as heart rate, oxygen saturation, respiratory rate, etc. In addition, by leveraging the different absorption rates of wavelengths, presence of specific blood content can also be estimated.

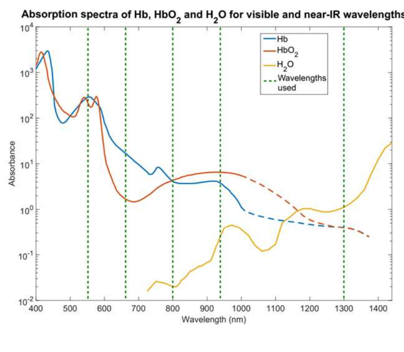


Figure 1. Light absorption properties for different blood content [6].

*: These authors contributed equally.

Even though PPG reflects the changes in blood volume instead of blood pressure (BP), it has been show in literature that the latter can be estimated from PPG [12]. By measuring the time blood travels from one site to another, the pulse wave velocity can be estimated, which is found to be directly correlated to age and systolic blood pressure (SBP) [13]. However, clinical validation of such algorithms has remained sparse due to the many practical difficulties present: general lack of training data, synchronisation between waveforms, noise-cancellation filters changing shape of waveforms, etc. We seek to tackle these problems with our system architecture detailed in the following sections.

II. Hardware

The device consists of three main parts, with an overview of the device shown in Figure 1. The analogue front-end chip used is AFE4900 from Texas Instruments, capable of driving 4 LEDs and sampling 3 photodiodes (PD) up to 1000 times per second. Two AFEs were used to acquire PPG from two body sites: wrist and finger. A standalone clock oscillator was used as the clock input to both AFEs with the track lengths carefully matched to ensure synchronous operation. Fast switching LED driver tracks were routed on the opposite side of the sensitive PD tracks to minimise interference and cross-talk. Both AFEs are currently placed on a small breakout board and plugged into the motherboard to reduce manufacturing costs and increase flexibility.

The two AFEs are connected to an MCU, ESP32-S3 in our current setup. The ESP32-S3 a low-cost, low-power dual-core microcontroller by Espressif that has built-in high-speed wireless transmission over Wi-Fi or Bluetooth. It also comes with large a flash and RAM that can be used to store the large incoming data stream. The ESPs is used to control both AFEs, battery gas gauge and the power block. In addition to the AFE, accelerometers and temperature sensors can be connected to the ESP32 using the headers via I2C buses. The accelerometer is used to estimate user motion and cancel motion artifacts from the PPG signal. Temperature sensors can provide additional information for the clinicians to make better decisions and used as a feature for our severity prediction algorithms.

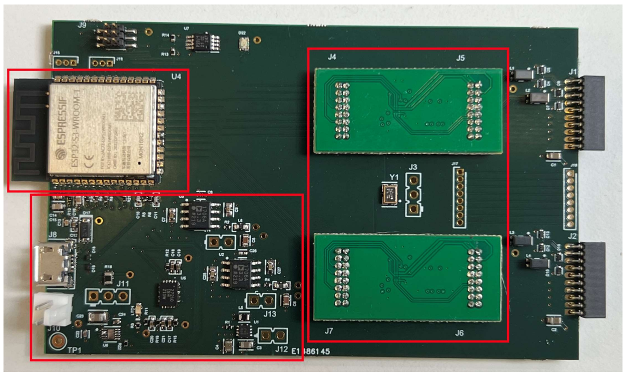


Figure 2. Overview of hardware device. Two AFE breakout boards are plugged into the motherboard and connected to a ESP32-S3 MCU. The system can be powered using a Li-ion battery and charged via USB.

A total of 8 LED channels were used, split equally between the finger and wrist probe. Five different wavelengths: 525nm, 660nm, 800nm, 940nm, 1300nm were used to perform pulse oximetry, haematocrit and PWV estimation. Both probes are connected to the motherboard using medical grade wires.

III. Software

The software aspect of the device enables two main capabilities for the device:

- Wireless communication between ESP32 and server using the message queuing telemetry transport (MQTT) protocol and optimised with Protobuf encoding.
- A dedicated user interface platform for clinicians to view patient PPG data.

The ESP32, through the MQTT broker, connects to a Python-based backend server and transmits the PPG data as shown in Figure 3. Using Protobuf encoding, there is a 3x reduction in a sample MQTT payload of 10 PPG readings, from on average 1.5 kB to 500 B.

The system is designed such that the backend services and user interface can run locally on a computer, eliminating the need for cloud computing which would add additional data privacy concerns.

*: These authors contributed equally.

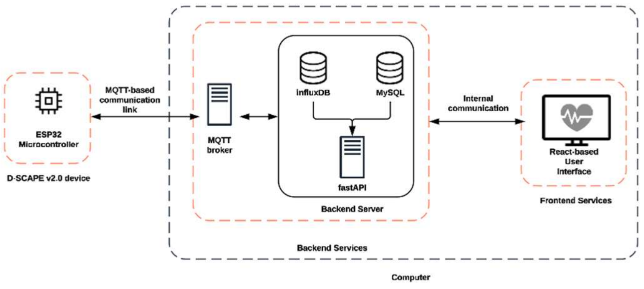
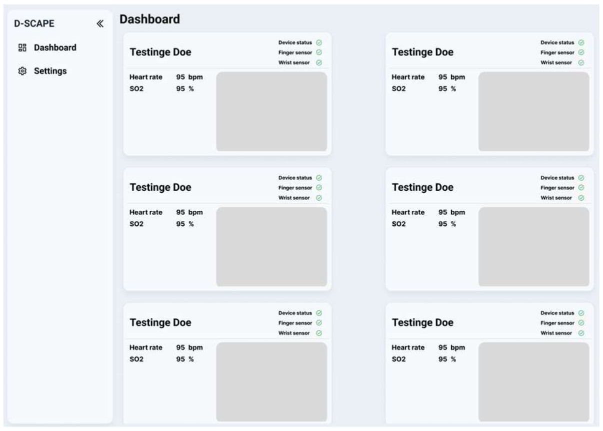


Figure 3. Backend system design.

The server performs processing and stores the data in a dedicated time-series database. In addition, the backend server enables an intuitive user interface for clinicians to manage devices and view stored patient data as shown in Figure 4. Key physiological parameters and device status are clearly presented to clinicians, along with a graph displaying trends or PPG waveforms. Clicking on a patient card will allow the user to view history data and advanced statistics. The dashboard can be further customised to display different parameters for each individual patient.



Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Testinge Doe
Device status ✓
Finger sensor ✓
Wrist sensor ✓
Heart rate 95 bpm
SO2 95 %

Figure 4. User Interface Currently Under Development

3. RESULTS

To verify the new prototype has similar capabilities to its predecessor, we used a test setup shown in Fig. 5. The setup consists of a finger probe, a ESP32-S3, and an AFE chip attached to the customized AFE board.

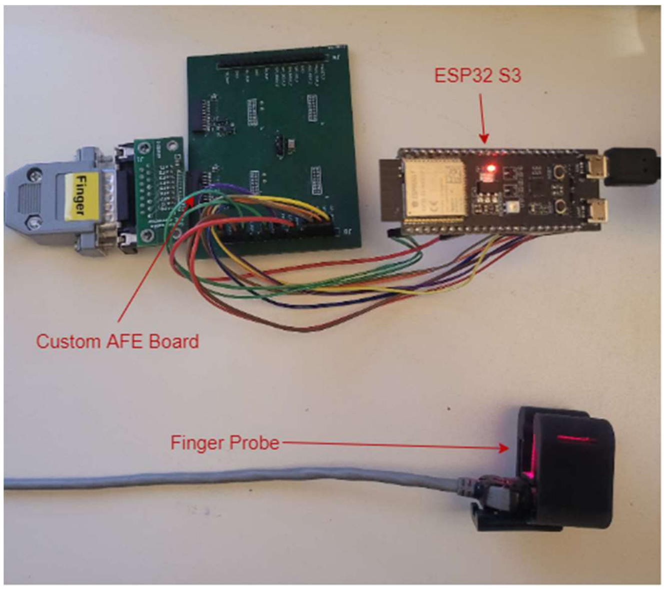


Figure 5. Simple hardware configuration setup used for testing. A single finger probe is connected to the AFE situated on a custom breakout board.

Using the previously shown setup, we were able to validate the functionality of our custom C library for AFE4900. We were also able to verify that a PPG signal can be extracted to estimate biological markers (e.g, heart rate) using our new and improved portable design. The raw and filtered PPG readings acquired from the device are shown in Fig. 6, showing a clear heart rate pattern with visible systole and diastole phases.

In addition, we were also able to verify that the device was operating at the desired frequency specified by the common clock oscillator. Therefore, the sampling rate discrepancies between two AFEs can be minimised, allowing for accurate PWV extraction and therefore, blood pressure tracking.

*: These authors contributed equally.

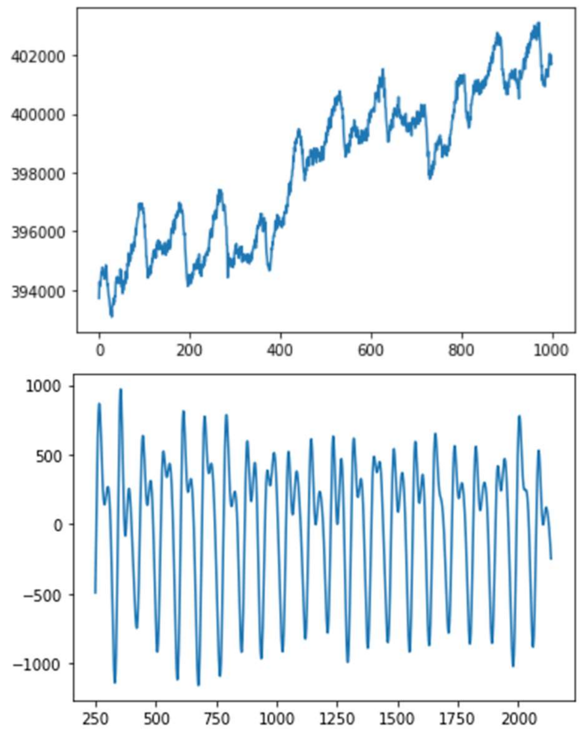


Fig. 6 Raw PPG waveform and filtered PPG waveform from our device. The filtered waveform contains periodic components which frequencies agree with the heart rate.

4. FUTURE WORK

While we have successfully validated the baseline functionality of our device, we have not yet been able to fully characterise the performance of our device. One area of future work is to validate the BP estimation in lab by comparing it to traditional cuff-based blood pressure monitors.

Another area of research is to minimise power usage to increase the portability and battery life of the device. This would highly benefit the clinicians, reducing the need for frequent check-ups.

Finally, there is a need to validate the performance of the device in a large-scale clinical trial. By comparing it to existing continuous monitoring products, if it is shown to be effective, it could have a significant impact even beyond the management of dengue, particularly in a resource-limited setting.

5. CONCLUSION

We have built a wearable device that utilises multi-wavelength PPG to carry out continuous monitoring for patients with dengue fever. Our device capable of sampling two PPG sites synchronously, processing the measured waveforms and uploading them to the cloud server for storage and visualization. Although this device is specifically tailored for monitoring dengue fever patients, it is important to note that the PPG-based methodology used for this medical wearable can be extended for numerous other diseases outside of dengue fever.

REFERENCES

[1] O. J. Brady et al., “Refining the Global Spatial Limits of Dengue Virus Transmission by Evidence-Based Consensus,” PLoS Neglected Tropical Diseases, vol. 6, no. 8, p. e1760, Aug. 2012, doi: https://doi.org/10.1371/journal.pntd.0001760.
[2] N. Alexander et al., “Multicentre prospective study on dengue classification in four South-east Asian and three Latin American countries,” Tropical Medicine & International Health, vol. 16, no. 8, pp. 936–948, May 2011, doi: https://doi.org/10.1111/j.1365-3156.2011.02793.x.
[3] T. P. Monath, “Dengue: the risk to developed and developing countries.,” Proceedings of the National Academy of Sciences, vol. 91, no. 7, pp. 2395–2400, Mar. 1994, doi: https://doi.org/10.1073/pnas.91.7.2395.
[4] G. Hentzy Moraes et al., “Determinants of Mortality from Severe Dengue in Brazil: A Population-Based Case-Control Study,” The American Journal of Tropical Medicine and Hygiene, vol. 88, no. 4, pp. 670–676, Apr. 2013, doi: https://doi.org/10.4269/ajtmh.11-0774.
[5] Samir Bhatt et al., “The global distribution and burden of dengue”. en. In: Nature 496.7446 (Apr. 2013), pp. 504–507. doi: 10.1038/nature12060.
[6] Kate Mulligan et al., “Is dengue a disease of poverty? A systematic review”. In: Pathogens and global health 109 (Dec. 2014), 2055033214Y0000000009. doi: 0.1179/2047773214Y.0000000168.
[7] Harapan Harapan et al., “Covid-19 and dengue: double punches for dengue-endemic countries in Asia”. In: Reviews in medical virology 31.2 (2021), e2161.
[8] Mervyn Singer et al., “The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3)”. In: JAMA 315.8 (Feb. 2016), pp. 801–810. issn: 0098-7484. doi: 10.1001/jama.2016.0287. url: https://doi.org/10.1001/jama.2016.0287

*: These authors contributed equally.

[9] William R. Judd et al., “Clinical and Economic Impact of a Quality Improvement Initiative to Enhance Early Recognition and
Treatment of Sepsis”. In: Annals of Pharmacotherapy 48.10 (2014). PMID: 24982314, pp. 1269–1275. doi: 10.1177/1060028014541792. url: https://doi.org/10.1177/1060028014541792.
[10] Trinh Manh Hung et al., “The estimates of the health and economic burden of dengue in Vietnam”. In: Trends in parasitology 34.10 (2018), pp. 904–918

[11] S. Karolcik et al., "A multi-site, multi-wavelength PPG platform for continuous non-invasive health monitoring in hospital settings," in IEEE Transactions on Biomedical Circuits and Systems, doi: 10.1109/TBCAS.2023.3254453.
[12] Elgendi, M. et al., “The use of photoplethysmography for assessing hypertension”. npj Digit. Med. 2, 60 (2019). https://doi.org/10.1038/s41746-019-0136-7
[13] M. Nitzan et al., “The difference in pulse transit time to the toe and finger measured by photoplethysmography,” Physiological Measurement, vol. 23, no. 1, pp. 85–93, 2001.

*: These authors contributed equally.

Appendix B Derivations

The work done by Stefan Hollos and J. Richard Hollos [43] has provided valuable insights into filter design principles and equation derivations. This section is meant to provide additional background on the different filter derivations as it is used in Section 6.2.

B.1 Band-pass Butterworth Filter Derivation

An ideal lowpass Butterworth filter is designed to have a flat passband and also attenuates all the frequencies outside the passband. This can be accomplished by defining the frequency response to have the following form:

$$G_n(j\omega) = \frac{Y(j\omega)}{X(j\omega)} \tag{13}$$

$$= \frac{1}{\sqrt{1+\epsilon^2\omega^{2n}}} \tag{14}$$

where $Y(j\omega) = 1$ and $X(j\omega) = 1 + \epsilon^2\omega^{2n}$ and ϵ is the maximum band pass ripple. For the sake of simplicity, we set $\epsilon = 1$. We then attempt to find the poles for the filter by setting $s = j\omega$.

$$1 + \epsilon^2\omega^{2n} = 0 \tag{15}$$

$$1 + \omega^{2n} = 0 \tag{16}$$

$$1 - s^{2n} = 0 \tag{17}$$

This is clearly a roots of unity equation to be solved.

$$1 = s^{2n} \tag{18}$$

$$e^{2\pi jk} = s^{2n} \tag{19}$$

$$je^{\frac{\pi j(2k-1)}{2}} = s^{2n} \tag{20}$$

$$je^{\frac{\pi j(2k-1)}{2}} = s^{2n} \tag{21}$$

$$\tag{22}$$

We then use the trigonometric identities to rewrite and get the roots of the poles to be:

$$p_k = -sin\frac{\pi(2k-1)}{2n} + jcos\frac{\pi(2k-1)}{2n}, \qquad k = 0, 1, 2, ..., n-1 \tag{23}$$

We can then write the transfer function using the poles in the generalised form:

$$G_n(s) = \prod_{k=0}^{n} \frac{1}{s - p_k} \tag{24}$$
$$= \frac{1}{D_n(s)}, \qquad n = 1, 2, 3... \tag{25}$$
$$\tag{26}$$

where

$$D_n(s) = \begin{cases} (s+1)\prod_{k=0}^{\frac{n-3}{2}}(s^2 + 2r_k s + 1), & \text{if n is odd} \\ \prod_{k=0}^{\frac{n-2}{2}}(s^2 + 2r_k s + 1), & \text{if n is even} \end{cases} \tag{27}$$
$$r_k = sin(\frac{\pi(2k+1)}{2n}) \tag{28}$$

Since this project uses digital values, this means the filter must be transformed to a digital version using the bilinear transform. Additionally, the project also requires a band-pass filter with cutoff frequencies ω_1 and ω_2 (upper and lower respectively). This can be combined into a single substitution.

$$s \rightarrow \frac{z^2 - 2az + 1}{b(z^2 - 1)} \tag{29}$$
$$a = \frac{cos(\frac{\theta_1 + \theta_2}{2})}{cos(\frac{\theta_1 - \theta_2}{2})} \tag{30}$$
$$b = tan(\frac{\theta_1 - \theta_2}{2}) \tag{31}$$
$$\theta_1 = \omega_1 T \tag{32}$$
$$\theta_2 = \omega_2 T \tag{33}$$

An example of this is for a 1^{st} order Butterworth filter, this would be as follows:

$$G_1(s) = \frac{1}{s+1} \rightarrow \frac{b(z^2 - 1)}{(1+b)z^2 - 2az + 1 - b} \tag{34}$$

B.2 Band-pass Chebyshev Type I Filter Derivation

A Chebyshev Type I filter is similar to the Butterworth filter with the exception of a steeper roll-off and a passband ripple. By using the definition of its frequency response

$$G_n(j\omega) = \frac{Y(j\omega)}{X(j\omega)} \tag{35}$$

$$= \frac{1}{\sqrt{1+\epsilon^2 T_n^2(\omega)}} \tag{36}$$

Where $T_n(\omega)$ is defined as a Chebyshev Polynomial of degree n. This can be defined as:

$$T_0(\omega) = 1 \tag{37}$$

$$T_1(\omega) = \omega \tag{38}$$

$$T_2(\omega) = 2\omega T_{n-1}(\omega) - T_{n-2}(\omega) \tag{39}$$

A visualisation of the first 4 Chebyshev Polynomials can be seen in Figure 41

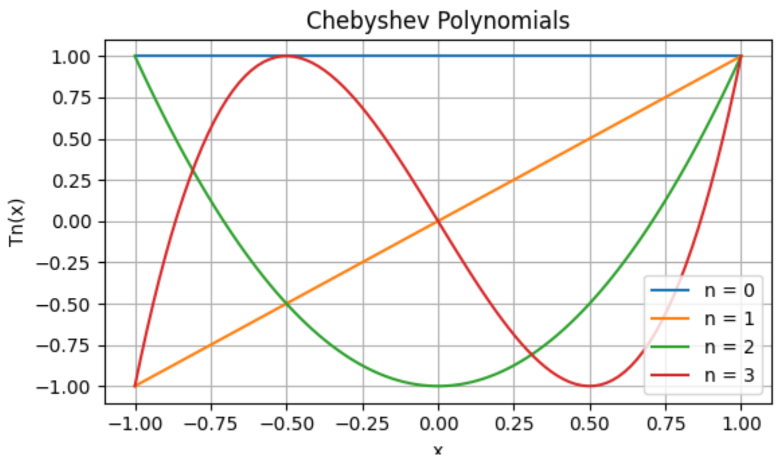


Figure 41: Chebyshev Polynomials up to degree n = 4

Additionally Chebyshev Polynomials of the first kind can also be defined as:

$$T_n(cos\theta) = cos(n\theta) \tag{40}$$

$$T_n(\omega) = = cos(n\theta) \tag{41}$$

Therefore by solving for the poles, we get the following equation:

$$1 + \epsilon^2 T_n^2(\omega) = 0 \tag{42}$$
$$1 + \epsilon^2 cos^2(n\theta) = 0 \tag{43}$$
$$cos^2(n\theta) = \frac{-1}{\epsilon^2} \tag{44}$$
$$cos(n\theta) = \pm\frac{j}{\epsilon} \tag{45}$$
$$\theta = \frac{1}{n} arccos(\pm\frac{j}{\epsilon}) + \frac{k\pi}{n} \tag{46}$$
$$\tag{47}$$

Using this solution of θ the Chebyshev Type I poles are therefore:

$$s_{pm} = jcos(\frac{1}{n} arccos(\pm\frac{j}{\epsilon}) + \frac{k\pi}{n}) \tag{48}$$

By using the properties of hyperbolic and trigonometric functions, we can express this equivalently as

$$s_{pm} = \pm sinh(\frac{1}{n} arsinh(\frac{1}{\epsilon})) sin(\theta_m) + jcosh(\frac{1}{n} arsinh(\frac{1}{e})) cos(\theta_m) \quad with\ k = 1, 2, 3, ...n \tag{49}$$

s_{pm} yields the poles both with positive and negative real parts. Since the transfer function must be stable, only the poles with negative real parts are taken. The transfer functions is given by:

$$\frac{1}{2^{n-1}\epsilon} \prod_{k=1}^{n} \frac{1}{s - s_{pm}^{-}} \tag{50}$$

By grouping the complex conjugate pairs between poles, we can express the system function in the following form.

$$G_n(s) = \frac{1}{D_n(s)} \qquad\qquad n = 1, 2, 3... \tag{51}$$
$$D_n(s) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases} \tag{52}$$

where

$$r_{pm} = Re(s_{pm}) \tag{53}$$

$$c_{pm} = |s_{pm}|^2 \tag{54}$$

In order to convert to this filter to a band-pass with cutoff frequencies ω_1 and ω_2 (upper and lower respectively) as well as transforming the filter into a digital one, this follows the same substitution in the Butterworth filter defined in (29). This then converts the second-order terms as shown below

$$\frac{1}{s^2 + 2rs + c} \rightarrow \frac{b^2(z^2-1)^2}{(b^2+2br+c)z^4 - 4a(1+br)z^3 - 2(b^2c - 2a^2 - 1)z^2 - 4a(1-br)z + b^2c - 2br + 1} \tag{55}$$

Appendix C Sample Code Listings

C.1 Project Github Page

All the code described here can be found in the GitHub link: "https://github.com/YutingXu/D-SCAPE_V2". This section is meant to give examples of setting up each library as well as the configuration definitions.

C.2 AFE4900

```c
AFE4900_cfg afe1 = {0};
afe1.spi_settings = (AFE4900SPISettings) {
    .miso = 21,
    .mosi = 13,
    .cs = 47,
    .sclk = 14,
    .busno = 2,
    .clock_freq = 4000000 // 4MHz
};
afe1.intr_settings = (AFE4900IntrSettings) {
    .prog_out_pin = 38,
    .adc_rdy_pin = 48,
    .prog_out_isr = NULL,
    .adc_rdy_isr = NULL
};
afe1.reset_pin = 39;

AFE4900_initialise(&afe1);
AFE4900_128kHz_load_1000Hz_finger_config(&afe1);

while(1) {
    uint32_t afe1_led1 = AFE4900_read_reg(afe1, AFE4900_REG_ALED1VAL);
    ESP_LOGI(TAG, "Led 1: %lu", afe1_led1);
    vTaskDelay(100 / portTICK_PERIOD_MS);
}
```

Listing 1: AFE4900 communication example

C.3 TMP177

```
typedef struct TMP117I2CSettings
{
  uint8_t scl_pin;
  uint8_t sda_pin;
  uint8_t addr; // 7-bit I2C addresss
  uint8_t port_num;
  uint32_t freq;
} TMP117I2CSettings;

typedef struct TMP117_cfg
{
  TMP117I2CSettings interface;
  uint32_t sensor_id;
} TMP117_cfg;
```

Listing 2: TMP117 Data Structures

```
////////////////// TMP117 //////////////////////

// Initialise TMP117 I2C
TMP117_cfg acc2 = {0};
acc2.interface = (TMP117I2CSettings) {
      .scl_pin = 1,
      .sda_pin = 2,
      .addr = 0x48,
      .port_num = 1,
      .freq = 100000
};
acc2.sensor_id = 2;
TMP117_initialise(&acc2);

int16_t tmp;
ESP_LOGI(TAG, "Got ID TMP117: %u", TMP117_get_device_id(&acc2));

while(1) {
      tmp = TMP117_get_temp(&acc2);
      ESP_LOGI(TAG, "Temp: %.4fC", tmp * TMP117_RESOLUTION);
      vTaskDelay(100 / portTICK_PERIOD_MS);
}
```

Listing 3: TMP117 Initialisation & Sample Program

Appendix D Log Output Results

D.1 128KHz Configuration Example

Listing 4: 128KHz Configuration Example

```
LED1LEDSTC = 26, LED1STC = 28, LED1CONVST = 44
LED1LEDENDC = 38, LED1ENDC = 38, LED1CONVEND = 59
LED2LEDSTC = 0, LED2STC = 3, LED2CONVST = 13
LED2LEDENDC = 12, LED2ENDC = 12, LED2CONVEND = 25
ALED1LEDSTC = 39, ALED1STC = 40, ALED1CONVST = 60
ALED1LEDENDC = 50, ALED1ENDC = 50, ALED1CONVEND = 75
ALED2LEDSTC = 13, ALED2STC = 15, ALED2CONVST = 27
ALED2LEDENDC = 25, ALED2ENDC = 25, ALED2CONVEND = 43

TG_PD1STC = 0, TG_PD1ENDC = 38
TG_PD2STC = 39, TG_PD2ENDC = 50

PRPCOUNT = 127
```

D.2 D-SCAPE-v1 Sample Reading

Listing 5: D-SCAPE-v1 Sample Reading

```
1676481171869626 84772 146406 -11290 -1916894
0 198752 146389 -8956 -1776642
0 265189 351453 80552 -1697626
0 250508 375644 88429 -1525463
0 211314 349463 75661 -983272
0 138305 271896 31238 540352
0 110438 191690 6244 1833428
0 103266 171908 974 1553156
0 99559 161974 -819 1225436
0 100856 161316 -1809 984452
...
```

D.3 TMP117 Sample Log

Listing 6: TMP117 Sample Log

```
(33424) main: Temp: 26.2188C
(34424) main: Temp: 26.2109C
(35424) main: Temp: 26.1953C
(36424) main: Temp: 26.1797C
(37424) main: Temp: 26.1641C
(38424) main: Temp: 26.1484C
(39424) main: Temp: 26.1328C
(40424) main: Temp: 26.1328C
```

D.4 TMP117 & ADXL343 Sample Log

Listing 7: TMP117 & ADXL343 Sample Log

```
(30434) main: X: 8.512172m/s^2, Y: 0.353039m/s^2, Z: 5.021005m/s^2
(30434) main: Temp: 27.9062C
(31434) main: X: 4.118793m/s^2, Y: -0.392266m/s^2, Z: 8.825985m/s^2
(31434) main: Temp: 27.9141C
(32434) main: X: 0.509946m/s^2, Y: -0.549172m/s^2, Z: 11.924887m/s^2
(32434) main: Temp: 27.9297C
(33434) main: X: -4.432606m/s^2, Y: 0.078453m/s^2, Z: 8.394493m/s^2
(33434) main: Temp: 27.9375C
(34434) main: X: 0.078453m/s^2, Y: 0.745305m/s^2, Z: 9.061345m/s^2
(34434) main: Temp: 27.9766C
(35434) main: X: 0.117680m/s^2, Y: 0.117680m/s^2, Z: 9.571291m/s^2
(35434) main: Temp: 28.0391C
```