**Contents**

## List of abbreviations

| | |
|---|---|
| AFE: | Analog Front End |
| FFT: | Fast Fourier Transform |
| MQTT: | Message Queuing Telemetry Transport |
| SoC: | System on Chip |
| FreeRTOS: | Free Real Time Operating System |

# 1 Introduction

This report discusses the development of a real-time heart rate monitoring system based on the Photoplethysmography (PPG) principle. The project builds upon a PPG sensor prototype developed earlier in a sensor course and extends it into a more complete system with embedded processing and IoT capabilities. After reaching the stage of a functional analog circuit, this project focuses on implementing a digital system capable of real-time monitoring using an integrated microcontroller platform.
The motivation behind this work arises from the limitations of most commercial wearable devices, which typically provide heart rate and oxygen saturation updates at intervals of around one minute. Such latency prevents users from observing rapid changes in their cardiovascular condition, potentially missing critical short-term variations such as transient arrhythmic events or sudden spikes in heart rate. Moreover, while clinical-grade ECG systems can provide highly accurate cardiac monitoring, their installation and maintenance are complex and costly. They generally require multiple electrodes, trained personnel, and a controlled environment, making them unsuitable for continuous everyday use or home-based health tracking.
Therefore, the goal of the project is to design a high-precision, low-latency monitoring system capable of providing near real-time cardiovascular feedback suitable for medical diagnostics, sports monitoring, and general wellness tracking.

The system initially used a dual-microcontroller architecture—STM32F103C6T6 for sampling and digital signal processing, and ESP8266 for communication and display. However, the design was later simplified and fully migrated to the ESP32 microcontroller platform. The ESP32 provides sufficient processing power, integrated Wi-Fi, and multiple peripherals, enabling it to perform all data acquisition, processing, and communication tasks simultaneously.

The objectives of the project are as follows:

- To implement a reliable real-time heart rate monitoring system using PPG.

- To design and test firmware for acquisition, filtering, and heart rate computation.
- To integrate display and wireless features within a single ESP32-based system.
- To provide a modular foundation for future expansion, such as $SPO_2$ or cloud integration.

The scope of this project includes both hardware and software development, signal acquisition, processing, testing, and evaluation.

# 2 Background and System Overview

## 2.1 Photoplethysmography (PPG) Principle

PPG is an optical measurement technique that detects volumetric changes in blood flow. When light is emitted onto the skin, a portion of it is absorbed or reflected depending on blood volume within the tissue. By monitoring the periodic variations in light intensity, the system can estimate the heart rate of the subject. PPG sensors are commonly used in wearable health devices because they are compact, inexpensive, and non-invasive. However, the signal is often affected by motion artifacts, noise, and ambient light, requiring careful filtering and signal processing.

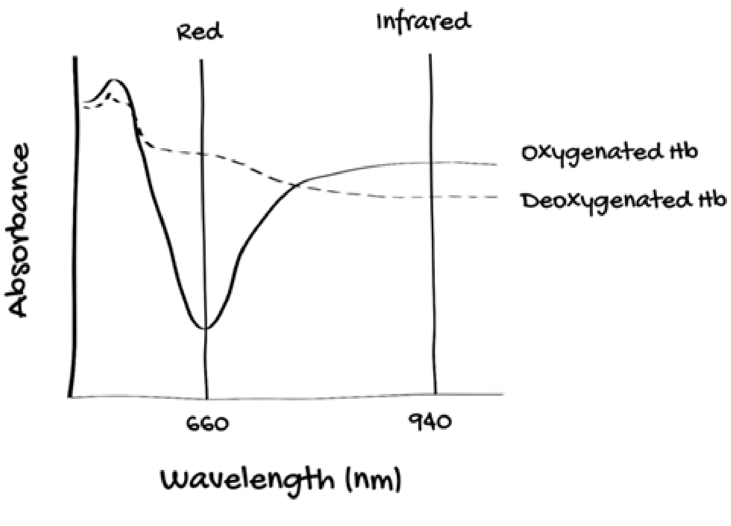


Figure 1. Absorbance of blood by light

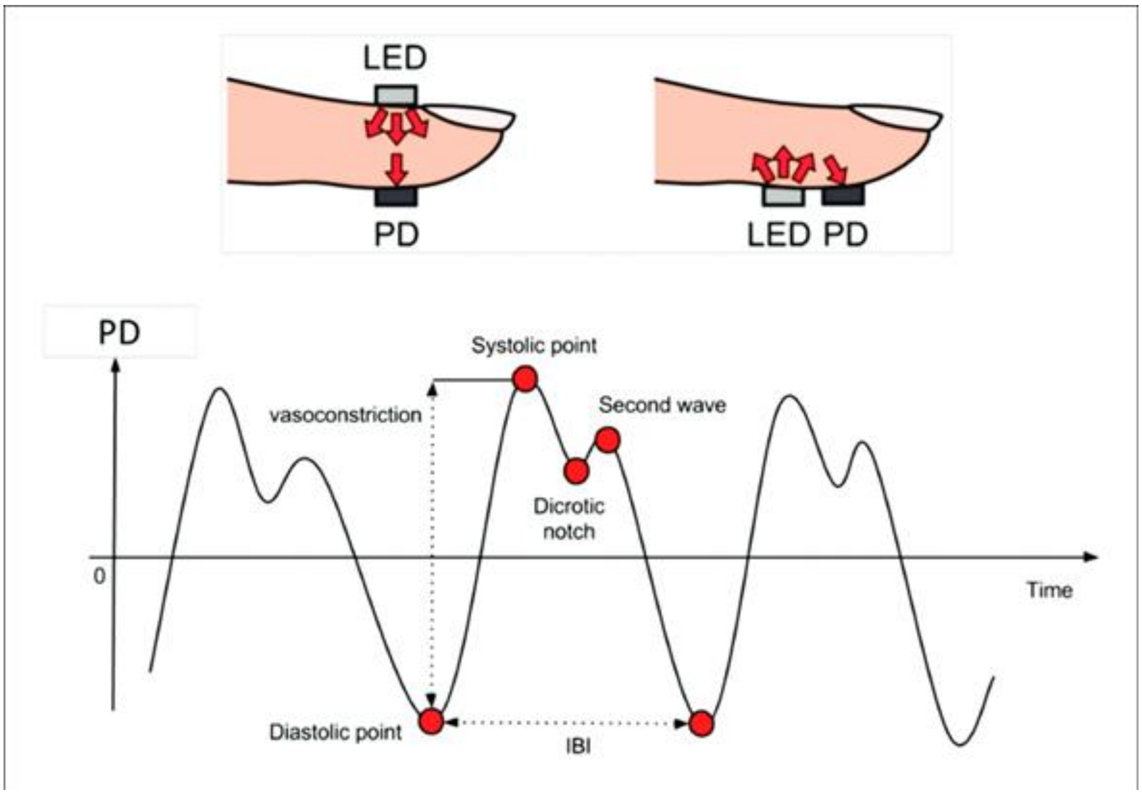
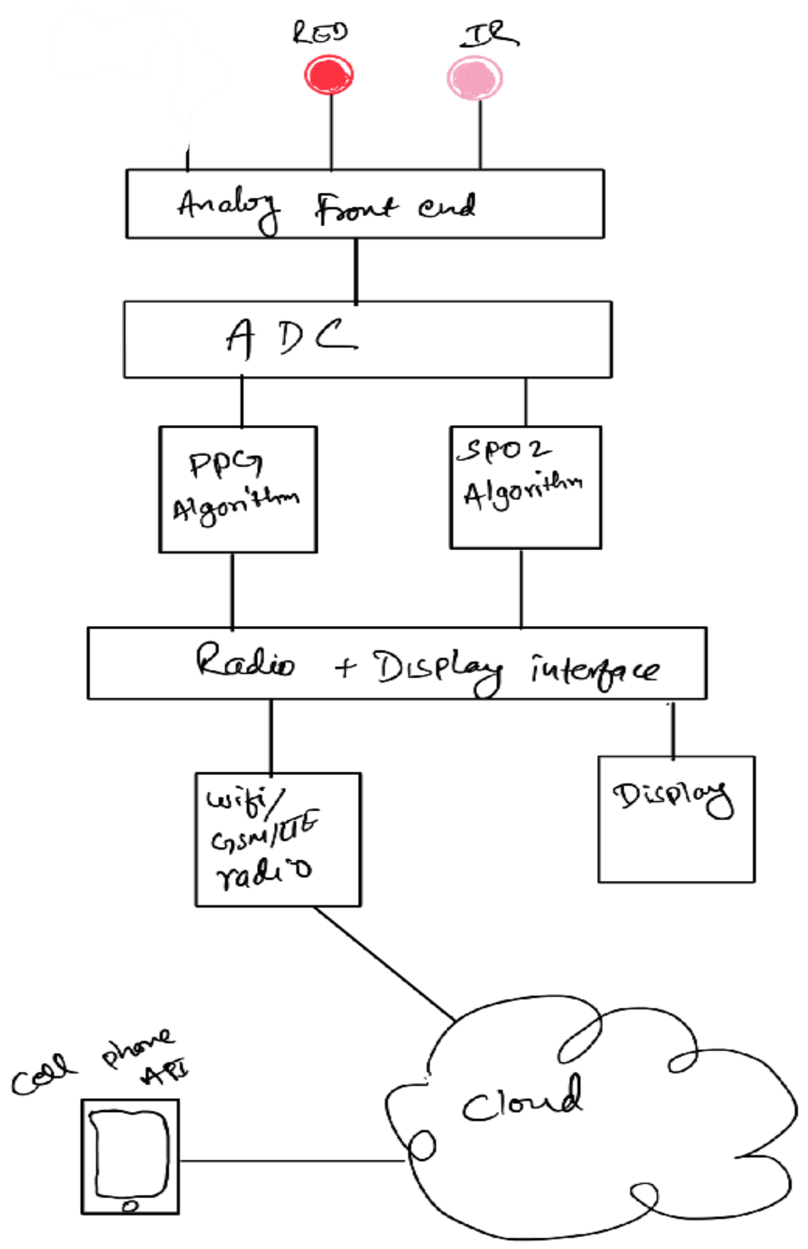


Figure 2. Typical PPG sensor setup and waveform. (*https://www.researchgate.net/figure/PPG-sensor-including-the-light-source-LED-and-the-light-receiver-PD-two-different_fig2_331211370* )

## 2.2 System Overview

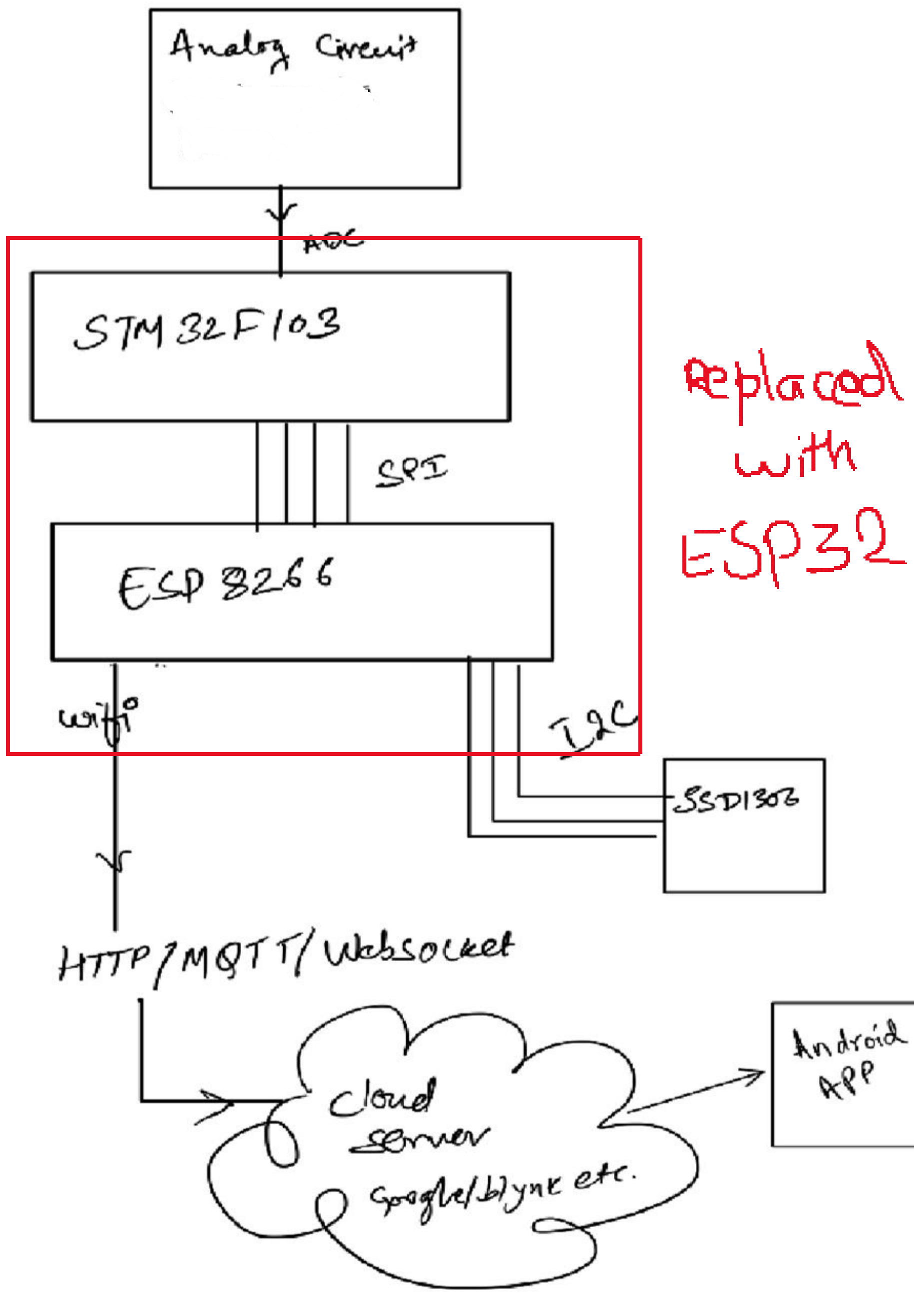
Overview of the project



The system acquires raw analog signals from the PPG sensor through the ADC of the ESP32, applies digital filtering, estimates peaks corresponding to heartbeats, and computes the heart rate in beats per minute (BPM) and publishes it to the remote android app.

# 3 Materials and Methods

## 3.1 Hardware Design



The final system is implemented entirely on the ESP32 microcontroller. It handles all tasks — data acquisition, processing, display, and wireless communication.

## 3.2 Hardware Components

- **ESP32** microcontroller: Dual-core MCU with Wi-Fi and Bluetooth.
- **PPG sensor module:** Infrared and red LED with photodiode receiver.
- **SSD1306 OLED display:** I2C interface for real-time output.
- **Power supply:** 5 V via USB.

### Analog Front END

The analog front end consisted of PIN Photodiode, a TIA, a bandpass filter and one more amplifier stage for amplification of the signal. The bandpass filter itself worked as both for noise removal and antialiasing filter. Filter response was simulated in LT-spice and was tested to get near result in Lab bench.

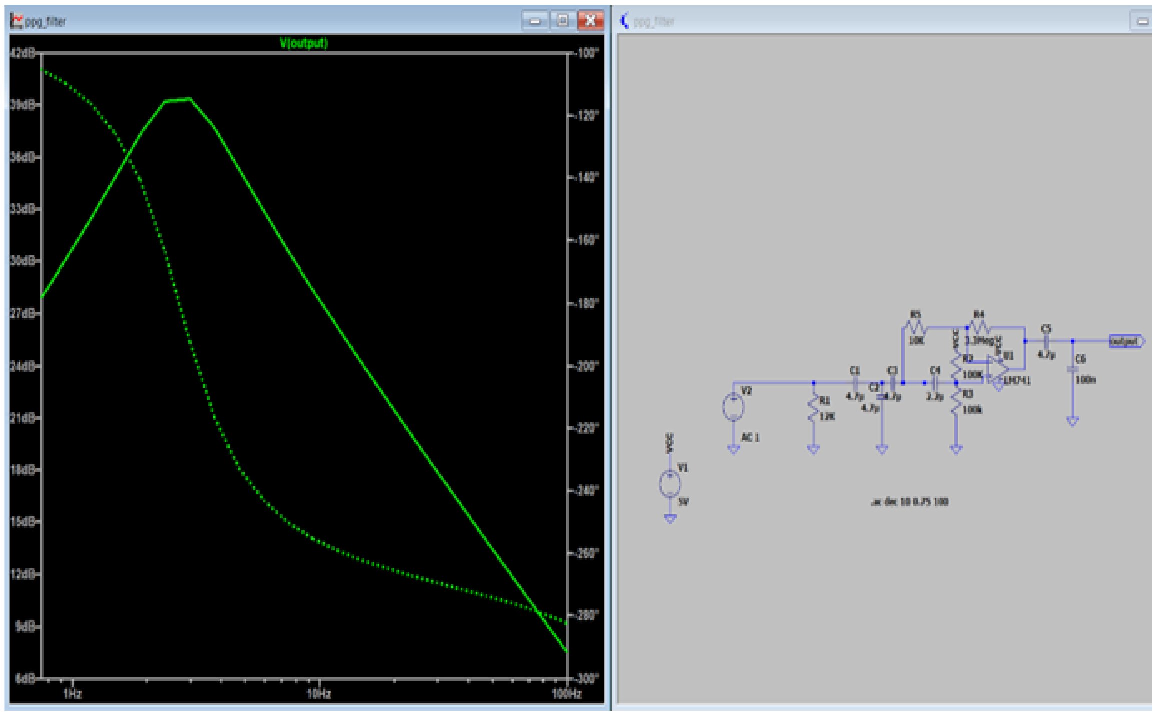
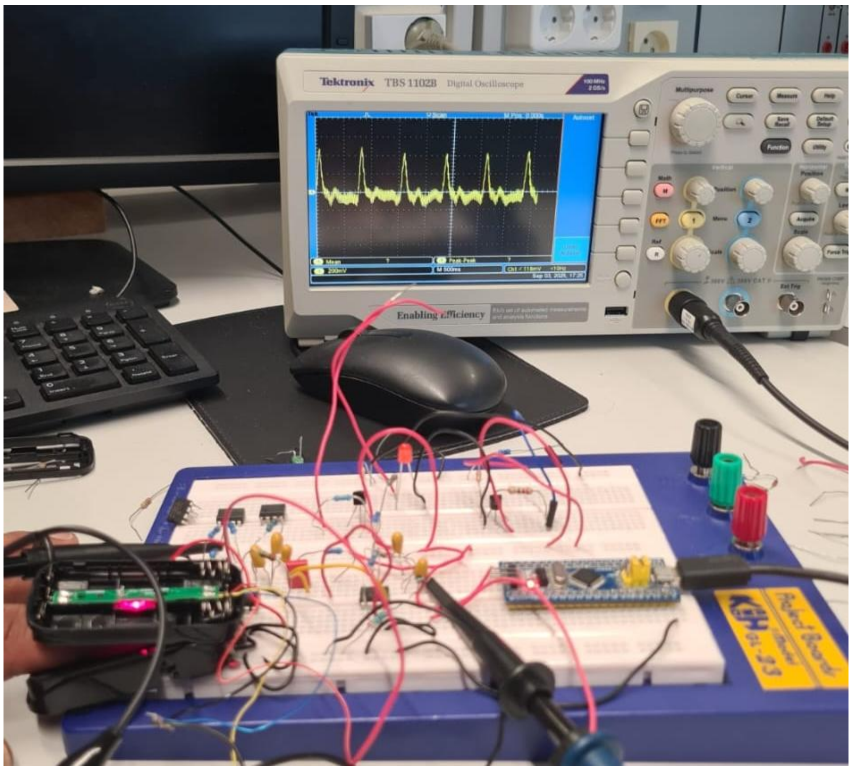
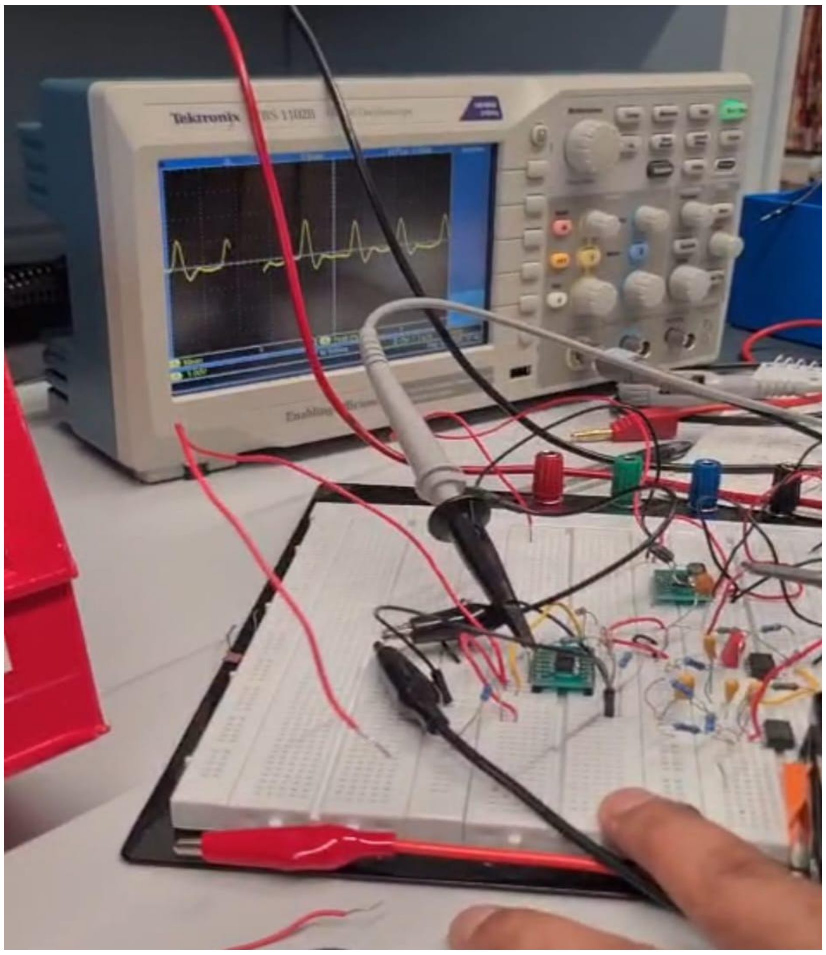


Figure 3. Filter simulation


(Image: Testing of AFE **contains High Frequency noise)*


(Image: Testing of AFE *cleaner signal)*

The design of the AFE was done in KiCad and was milled in two-layer PCB. Main components were Osram photodiode, MCP6272 dual opamp.

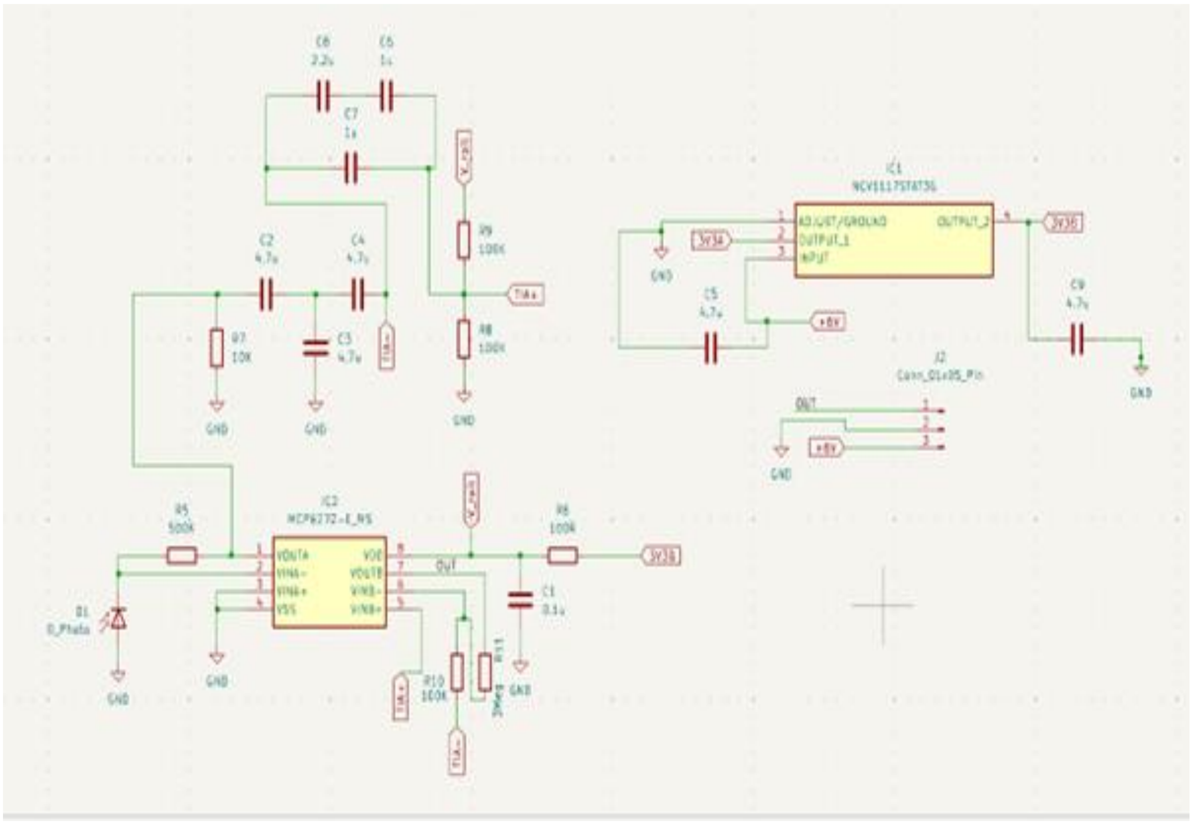
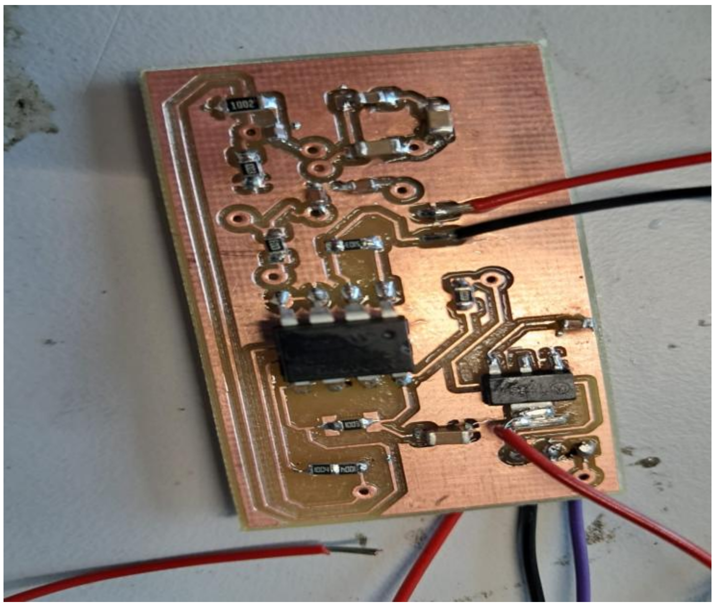


Figure 4. AFE schematic



(Image: AFE PCB)

## 3.3 Digital back end

Though the project was initially developed with STM32F103 and esp8266 due to possible complexity of PCB implementation, this was shifted entirely to single ESP32WROOM SoC.

The SoC comprised of 12-bit ADC resolution, dual CPU, I2C peripheral to connect to external display and radio unit (WIFI and Bluetooth) in a single package. The reference design of the SoC was studied from the manual provided by Vendor. The SoC is of SMD type and not breadboard friendly requiring a development PCB. Thus, two iterations of PCB were milled. The PCB designs were done in KiCAD and milled in the school lab.

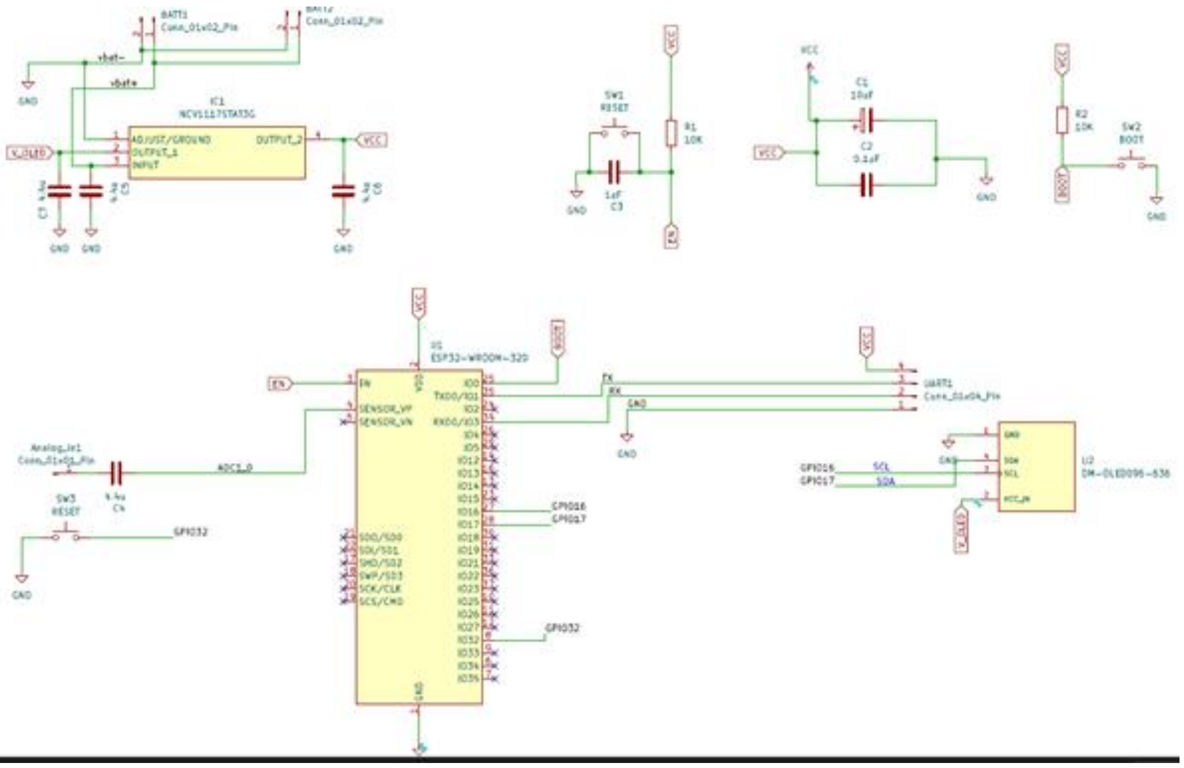
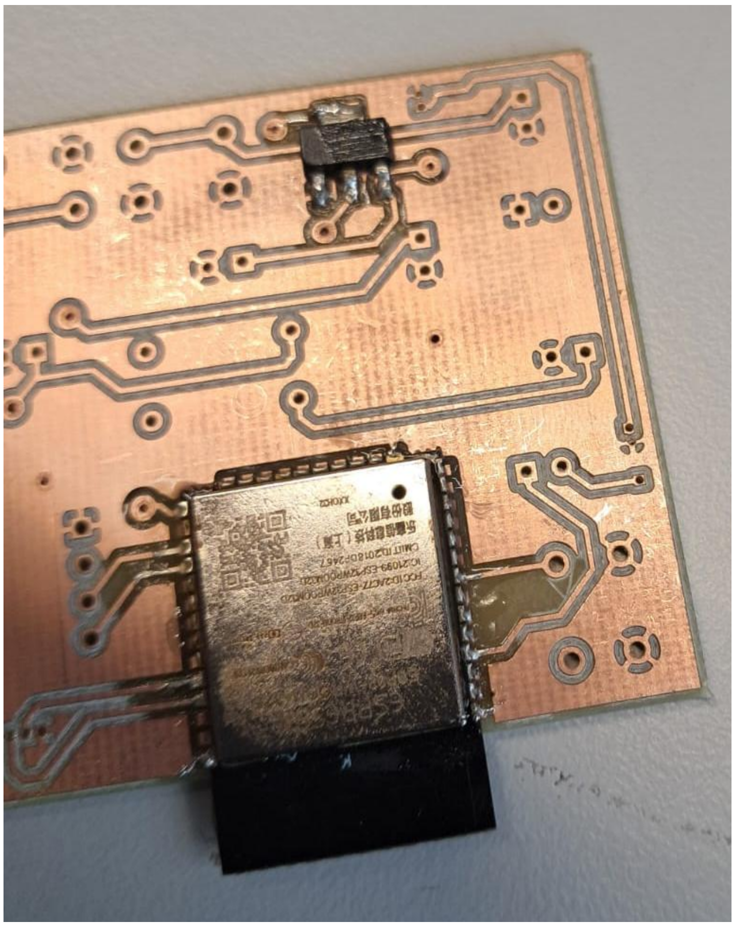


Figure 5. Digital back-end schematic



(Image: Digital back-end PCB)

## 3.4 Software and Algorithm Design

Firmware development was done using the **ESP-IDF** framework with FreeRTOS. Tasks are divided as follows:

- **Acquisition Task:** Samples analog data at a fixed rate using timer interrupts.
- **Processing Task:** Applies digital filtering and peak detection to compute heart rate.
- **Display Task:** Updates the OLED display periodically with the calculated value.
- **Wi-Fi Task:** Provides wireless data transmission for remote monitoring.

## 3.5 Signal Processing

Large collection of PPG signal was sampled at 50HZ for research, testing algorithm building, and simulation from the developed AFE.

**Initial Observations and Noise Characteristics**

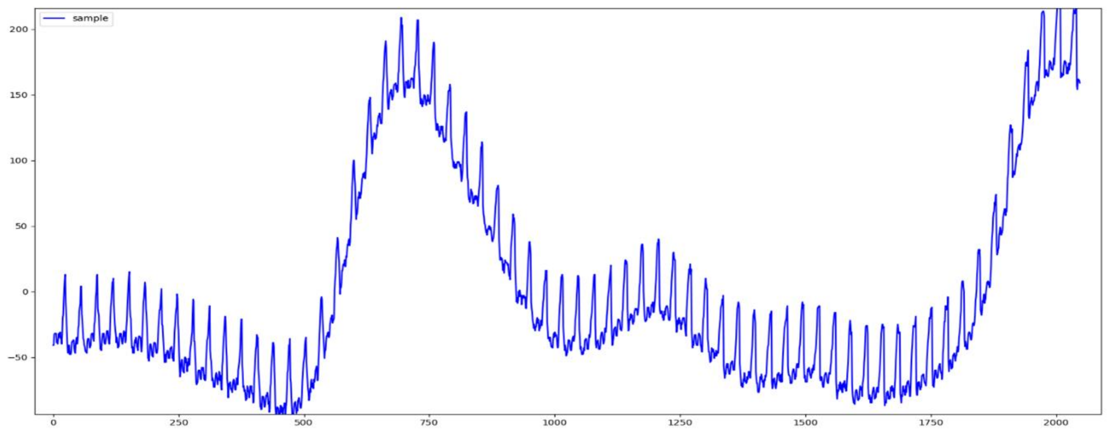


Figure 6. Samples collected from PPG sensor

The dataset consists of 2048 samples collected from the PPG sensor, sampled at 50 Hz. A visual inspection of the raw signal reveals the presence of mixed low-frequency noise, which contributes to baseline wander.

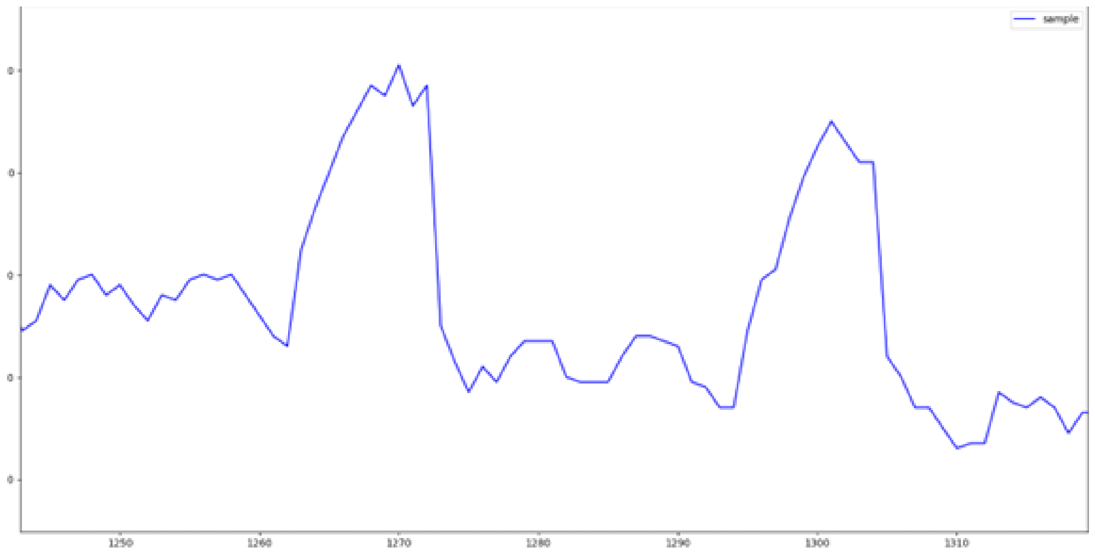
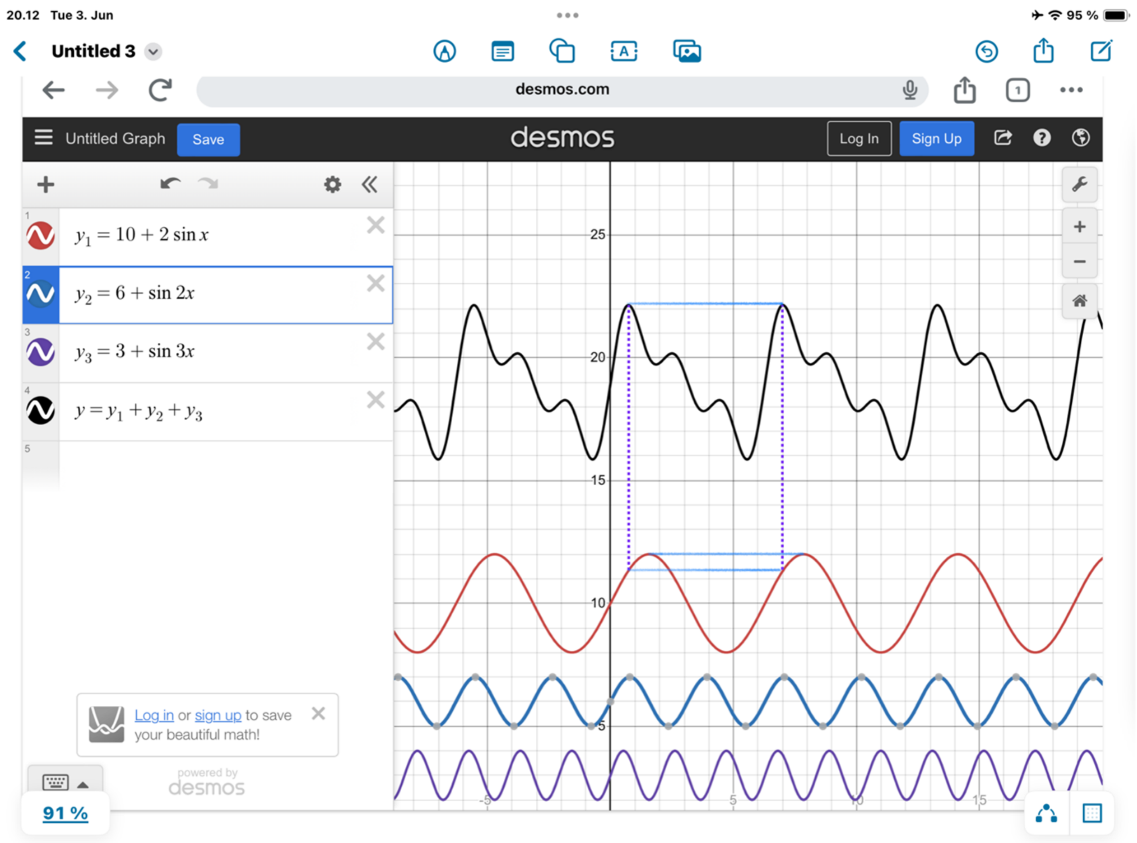


Figure 7. Shows peak when zoomed in

Additionally, when zooming in on individual peaks, high-frequency noise components become apparent.
An interesting observation is the unpredictability of these noise frequencies, some of which lie very close to the fundamental signal frequency, making them particularly difficult to filter out without distortion. While it's theoretically possible to design an analog filter to isolate the signal components, such precise filtering is non-trivial. It becomes even more challenging when dealing with a handmade circuit in a basic school lab environment, where component tolerances, power supply noise, and layout constraints introduce their own complications.

## 3.6 Foundational Principle of algorithm

Fifi


$y_1 = 10 + 2\sin x$

$y_2 = 6 + \sin 2x$

$y_3 = 3 + \sin 3x$

$y = y_1 + y_2 + y_3$

Figure 8. PPG signal modelled using Demos online software

We model our PPG signal as The Graph $y$. We can see that $y$ is the linear sum of $y_1, y_2 and\ y_3$ . And the period of the largest peak in $y$ corresponds to the period of the largest frequency component $y_1$. We can think of our PPG sinal as a sum of such several components. And now, here's the interesting bit: the biggest peak in the PPG signal corresponds to the frequency component with bigger amplitude (or the proportional constants in Fourier series $f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$ ).

That component, a pure, clean sinusoid is the one that carries the heartbeat. So, if one can extract it properly and measure its period, one can estimate the heart rate accurately.

Since our samples are in discrete form we use DFT

$\boldsymbol{X}[\boldsymbol{k}] = \frac{1}{N}\sum_{\boldsymbol{n}=0}^{N-1} \boldsymbol{x}[\boldsymbol{n}]\boldsymbol{e}^{-i\frac{2\pi}{N}\boldsymbol{kn}}$ to extract the frequency content of the signal.

Below are the sampled Signal from the AFE and its spectral content for window of 256 samples.

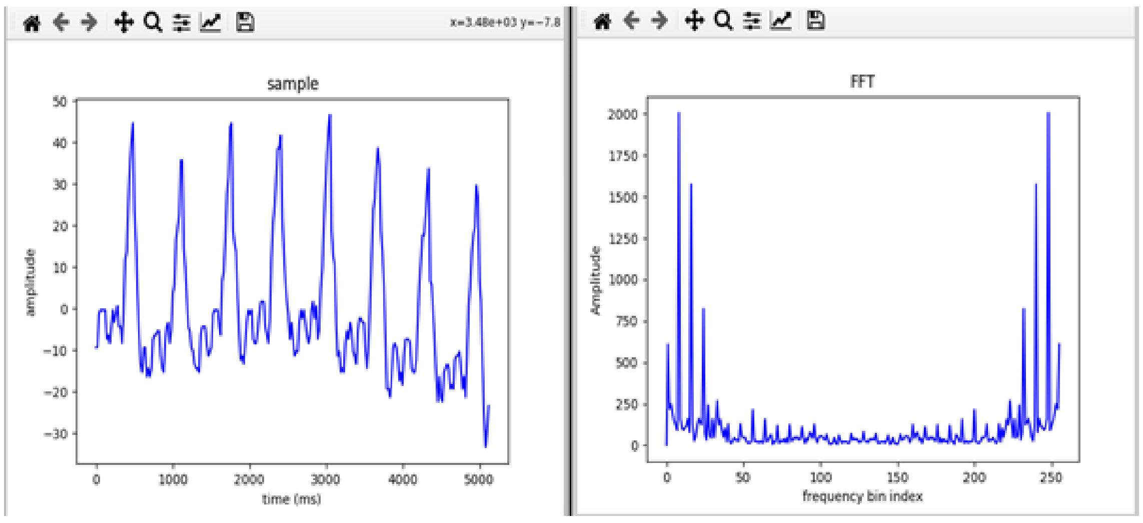


Figure 9. PPG signal and FFT

Firstly, you can see that the FFT plot is symmetrical about point N/2. It is always true for real valued signals like ours. So, one can discard the spectral content beyond N/2. So, the spectral content we are only interested in becomes.

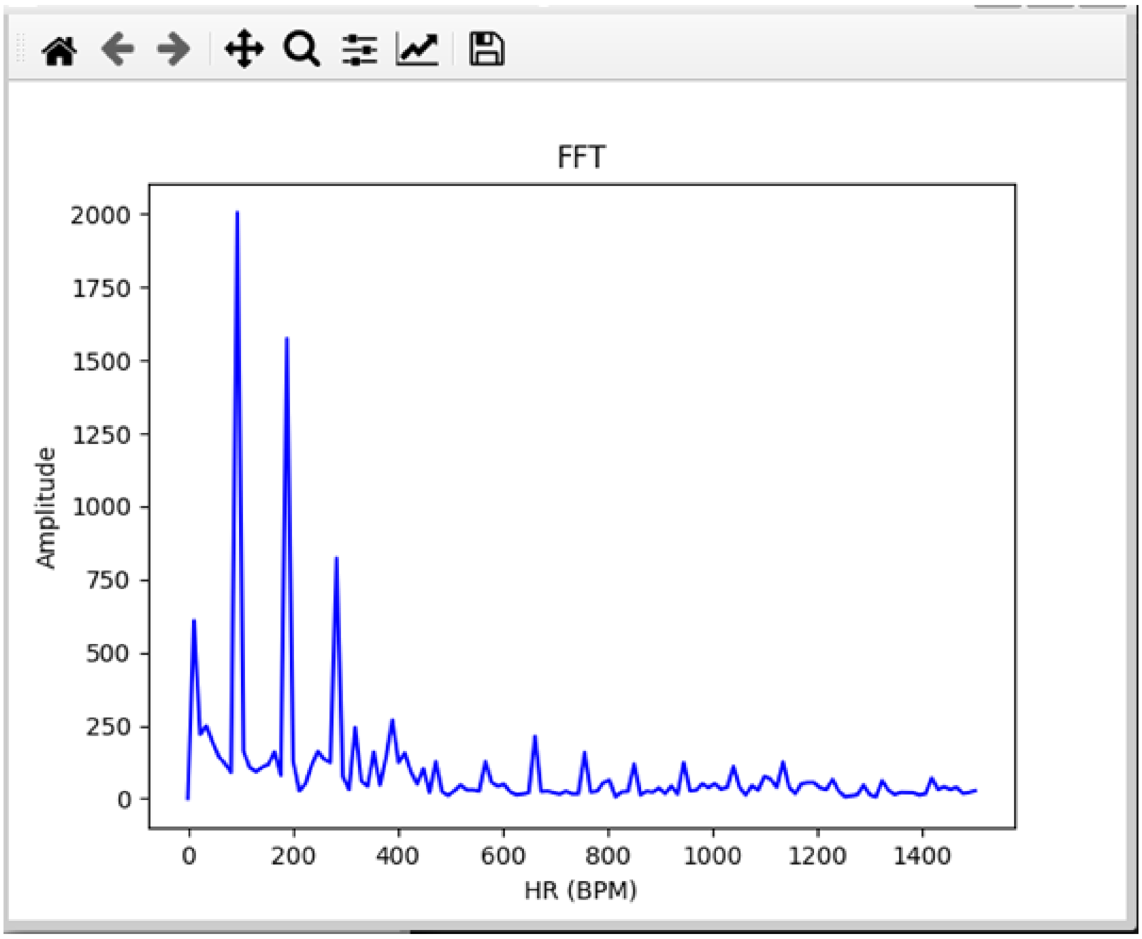


Figure 10. N/2 FFT plot

**Adaptive filtering in Frequency domain**

Now that we have information about the frequency content we can proceed with filtering. And it is very easy in frequency domain since convolution (filtering operation) is just a multiplication here. But in FFT it gets even better, if one wants to get rid of the frequency, “Just clear the bins”).

Okay, first we clear out the extreme frequency contents. Heart rate of humans recorded from lowest to highest is ~28 to ~300BPM, both of which are medical emergencies. So, the frequency would be ~0.467 to 5 Hz. And bins corresponding those would be 2.40 and 25.6, but we don’t have fractional bins, so we take the nearest bins 3 and 26 and keep only the bins between those, clearing all other. After removing the frequencies, we confidently don’t need. We still must take the problem that noises appear in the selected band. For this we assume that our SNR is high (signal is stronger than noise) so we can check between those bins with highest amplitude and keep only that removing all others.

*(Note* The frequency corresponding to the bins are dependent on sampling frequency, here this is based on test sampling frequency 50Hz, implementation has changed accordingly)*

**Estimation of actual Frequency**

Here, the Heartbeat is clearly in the position where the peak is highest. But there's an important observation.

256 samples at 50Hz, corresponding to $256 \times 20ms = 5120ms$. So, our frequency resolution for each bin will be $\frac{50Hz}{256} = 0.1953125Hz$, i.e 11.71875 BPM. This has serious implication; we can only get results accurately for beats in difference of 11.71875 BPM. To reduce that we clearly need to increase the sample time but then it will not be as real time system as it would be in the objective of the project.

The problem was recognized and studied already (as far as we came across) in the early 90s. One such interesting and valuable piece of work on it for building low order complexity algorithm was done by BG. Qininn in his Cambridge book. Although the analysis of the problem was published in IEEE paper.

Python simulations between different interpolation techniques were done for Frequency estimation based on spectral coefficients and compared based on the errors produced by each technique. Although Quinn's estimator defines the problem and is widely used considering for low minimum square error. The Log-parabolic interpolation produced better results for us in the simulation and thus it was adopted. Here we list only the Quinn's and Log-parabolic interpolation since other interpolation techniques were not considered because of large errors.

Quinn's second estimator:

$$\alpha = \Re\frac{X_{k-1}}{X_k}, \quad \beta = \Re\frac{X_{k+1}}{X_k}, \quad \Delta_{-1} = \frac{\alpha}{1-\alpha}, \quad \Delta_{+1} = \frac{\beta}{1-\beta}, \quad \delta = \begin{cases} \Delta_{+1}, & \Delta_{+1} > 0, \\ -\Delta_{-1}, & \Delta_{+1} \leq 0 \text{and} \Delta_{-1} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Log-parabolic interpolation:

$$\begin{aligned} P_k &= 20\log_{10}\left(\sqrt{(\mathrm{Re}[X_k])^2 + (\mathrm{Im}[X_k])^2 + 10^{-9}}\right) \\ p &= \frac{1}{2} \cdot \frac{P_{k-1} - P_{k+1}}{P_{k-1} - 2P_k + P_{k+1}} \\ k_{\text{peak}} &= k + p \\ f_{\text{peak}} &= \frac{k_{\text{peak}}\, f_s}{N} \\ \mathrm{HR}_{\text{est}} &= 60 \times f_{\text{peak}} \end{aligned}$$

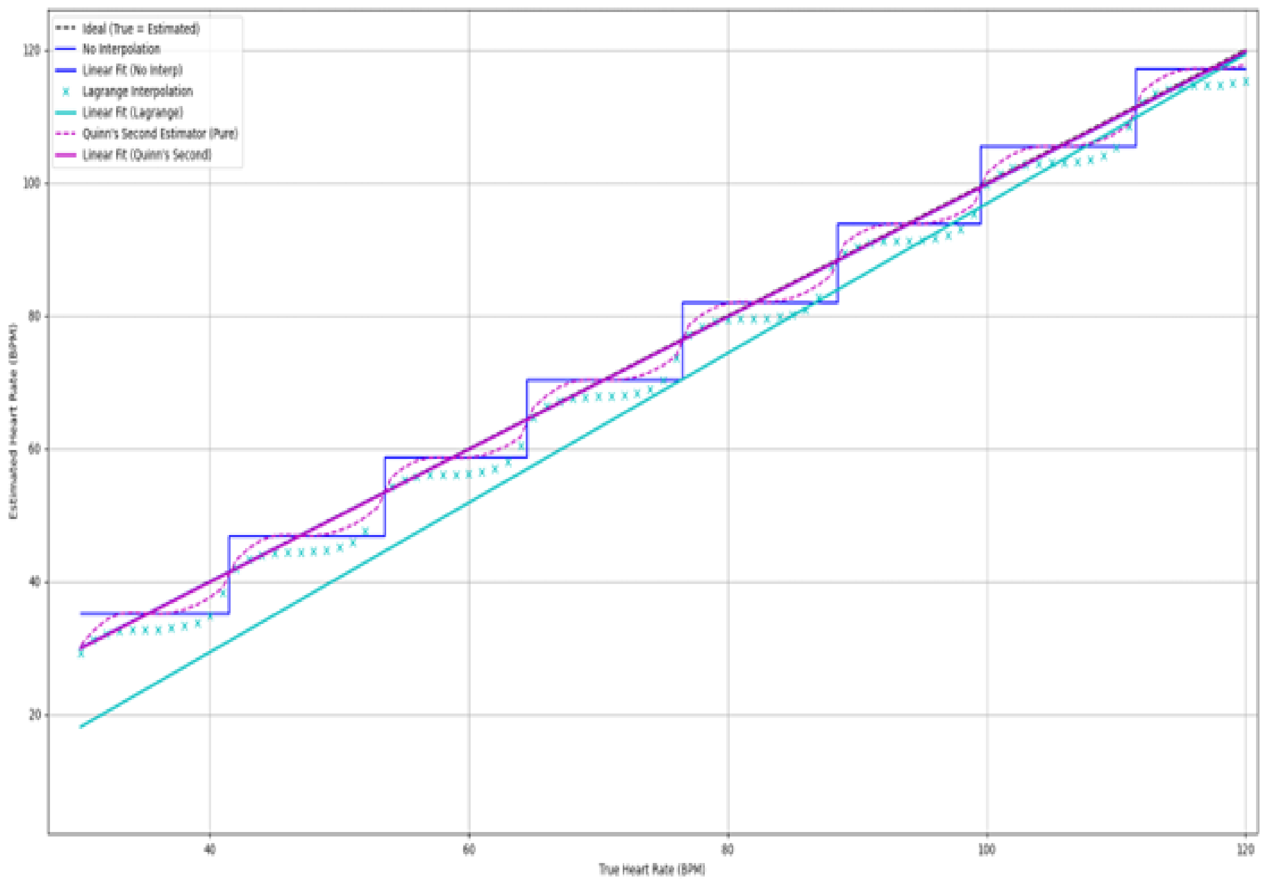


Figure 11. The comparison between different techniques

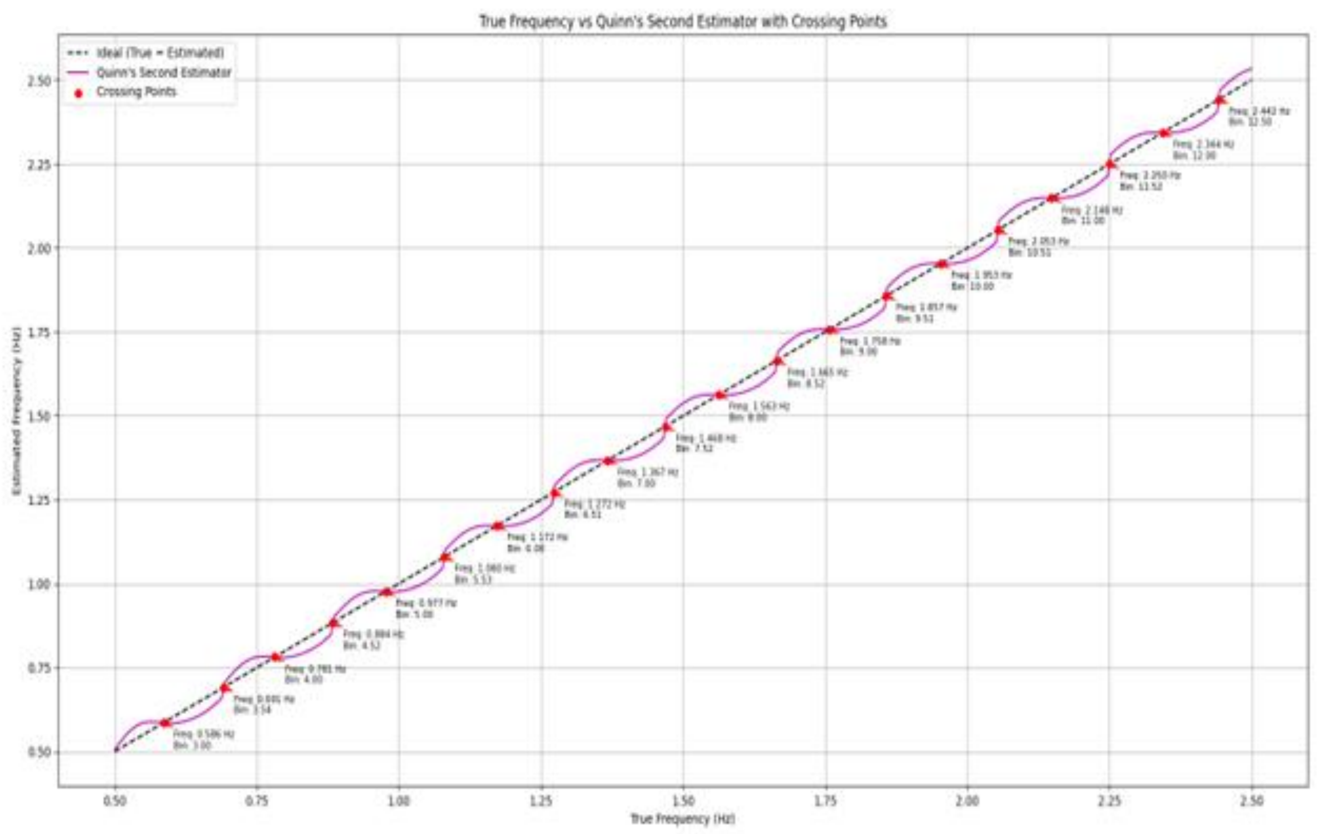


Figure 12. Quinn’s Second estimator

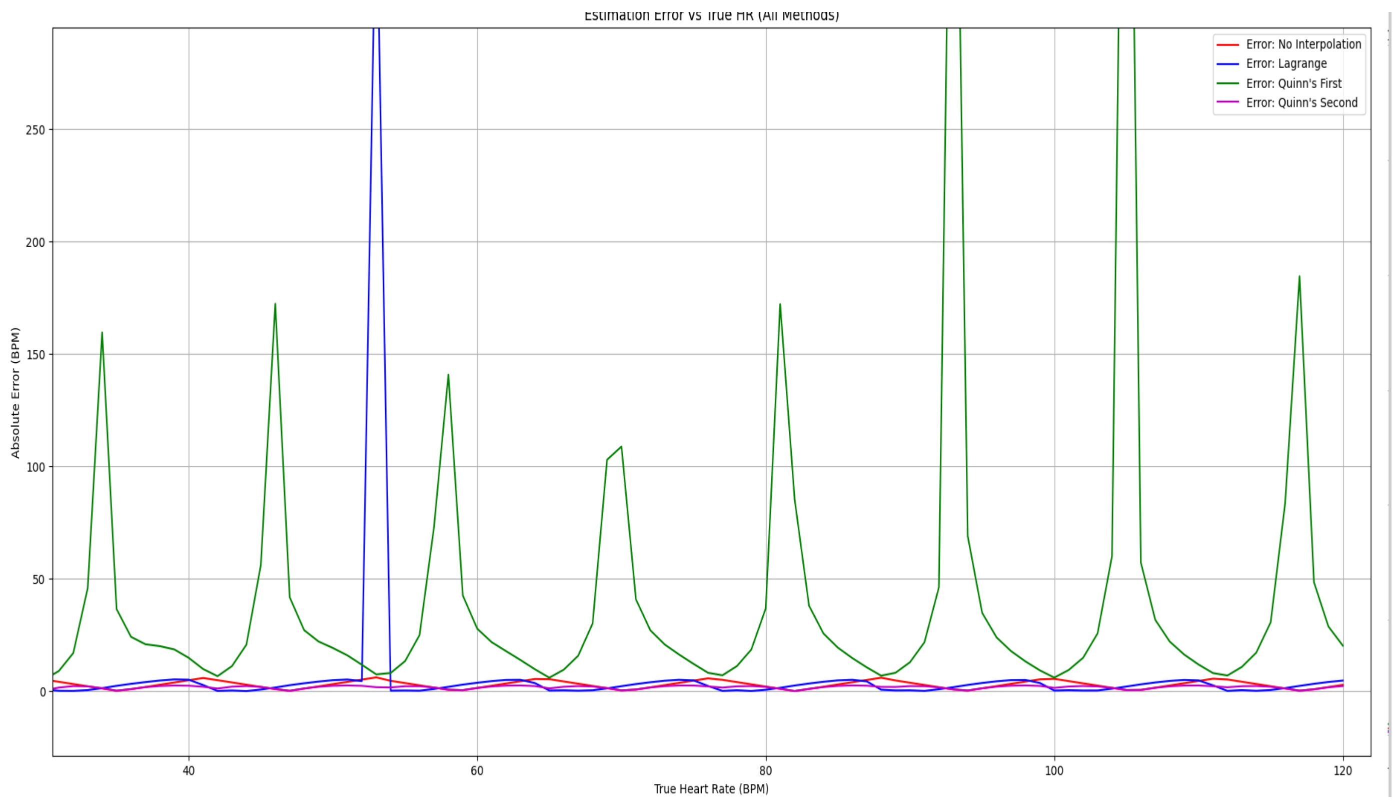


Figure 13. Errors of different techniques

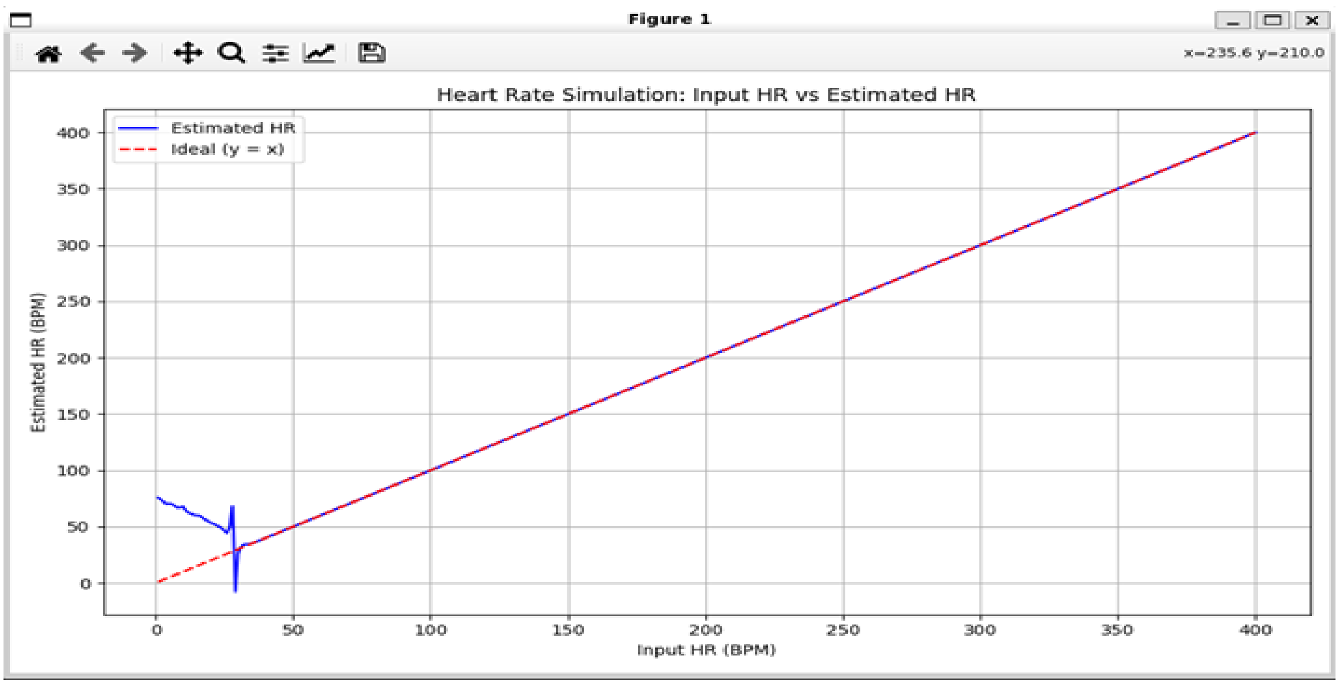


Figure 14. The simulation result of Log-parabolic interpolation

Due to the broadness of the topic this report does not discuss other relevant parts such as noise behavior modelling and simulations.

The software and algorithm were then implemented with following components:

1. **Sampling:** The ADC samples at 222.22 Hz.
2. **Filtering:** A digital bandpass filter isolates heart rate frequency (~0.4–5 Hz).
3. **Frequency Detection:** Fixed point FFT library from CMSIS_DSP was used for testing, while final product used esp_dsp library for faster processing.
4. **Computation:** Heart rate (BPM) = 60 * estimated frequency.
5. **Version control:** The code base and simulations are maintained in GitHub.

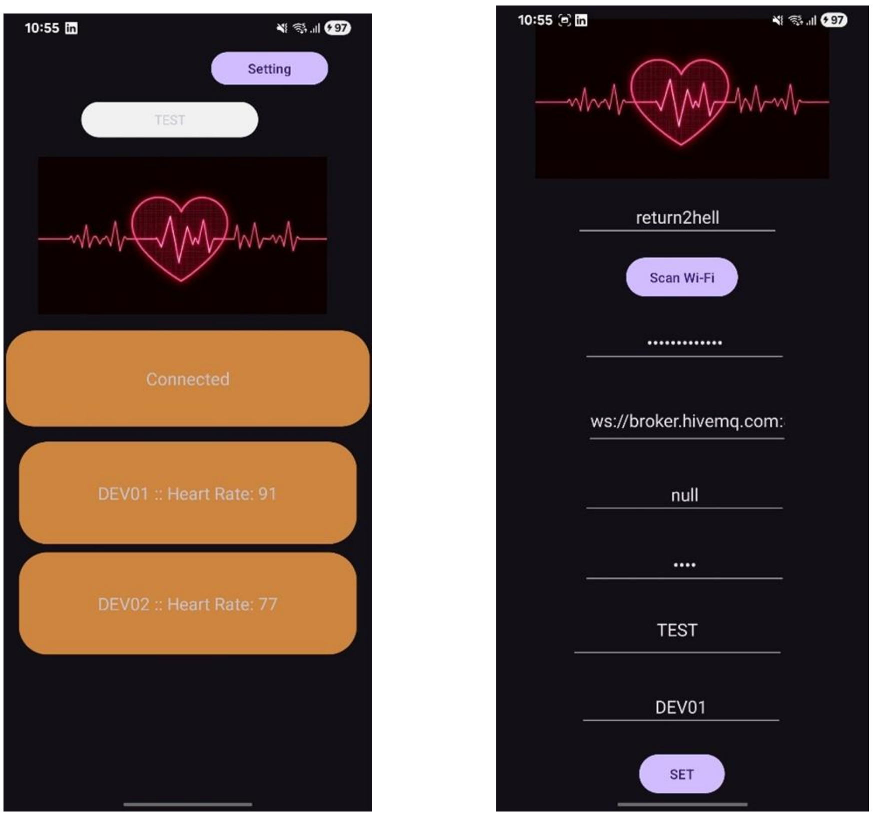
## 3.7 Development Tools

- ESP32WROOM SoC

- ESP-IDF SDK
- Python (for signal simulation and data analysis)
- LTspice (for analog front-end modeling)
- GitHub (for version control)

### Network

The acquired result from the system is then visualized on the OLED display and published to a remote android app using MQTT protocol. A simple android app interface was developed for this matter which was also able to set the device with proper connection settings. And used HTTPS and MQTT-TLS for secured connection. The system also featured the capability of creating a network of these devices so that multiple monitoring can be done at once.



DEV01 :: Heart Rate: 91

DEV02 :: Heart Rate: 77

return2hell

•••••••••••••

ws://broker.hivemq.com:

null

••••

TEST

DEV01

(Image: Android Application)

# 4 Workflow And Development process

The project was carried out in several phases:

1. Initial Concept and Research:
   A dual-microcontroller design (STM32 + ESP8266) was first implemented to separate sampling and communication.
2. **Architecture Simplification:**
   Due to complexity and synchronization challenges, the design was migrated to a single ESP32 platform.
3. **Firmware Implementation:**
   Tasks were developed incrementally — acquisition, filtering, display, and Wi-Fi modules.
4. **Testing and Validation:**
   Real PPG signals were tested under different lighting and motion conditions. Data were analyzed using Python to verify algorithm accuracy.
5. **Optimization:**
   CPU load and task timing were optimized using FreeRTOS scheduling.

# 5 Results And Observations

The ESP32 system successfully acquired and processed real-time PPG signals. The heart rate output was displayed on the OLED screen with updates every ~2.6-3s seconds, depending on the averaging window.

**Performance Metrics**

- Sampling Rate: 222.22 Hz
- Heart Rate Update Interval: ~2.6 s
- Power Consumption: < 150 mA
- ADC Resolution: 12-bit
- Processing Delay: < 1 s

The signal remained stable under static conditions but degraded under strong motion, indicating sensitivity to motion artifacts — a known PPG limitation. The accuracy of the device was compared with the cheaply available device, the difference between two were miniscule but couldn’t be stated exactly since the measurement update of the developed device was fast (as planned).

$SPO_2$ measurement was planned but not implemented in this version. However, the system architecture supports future extension to dual-wavelength analysis for oxygen saturation estimation.

# 6 Review Of Results

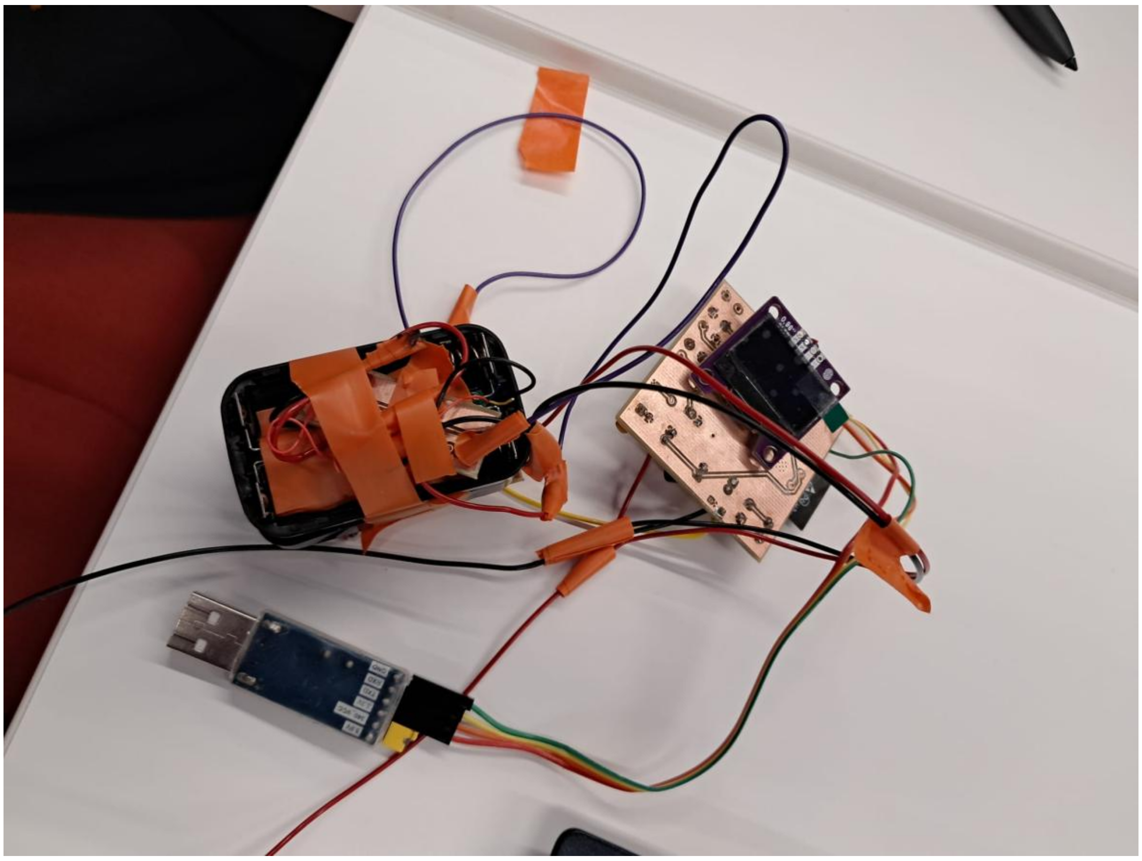
The project achieved its core objectives:

- Real time Heart Rate monitor(~2.6S)
- Successful migration from a multi-board system to a single ESP32 controller.
- Real-time heart rate monitoring with reliable performance.
- Integrated local display and wireless connectivity.

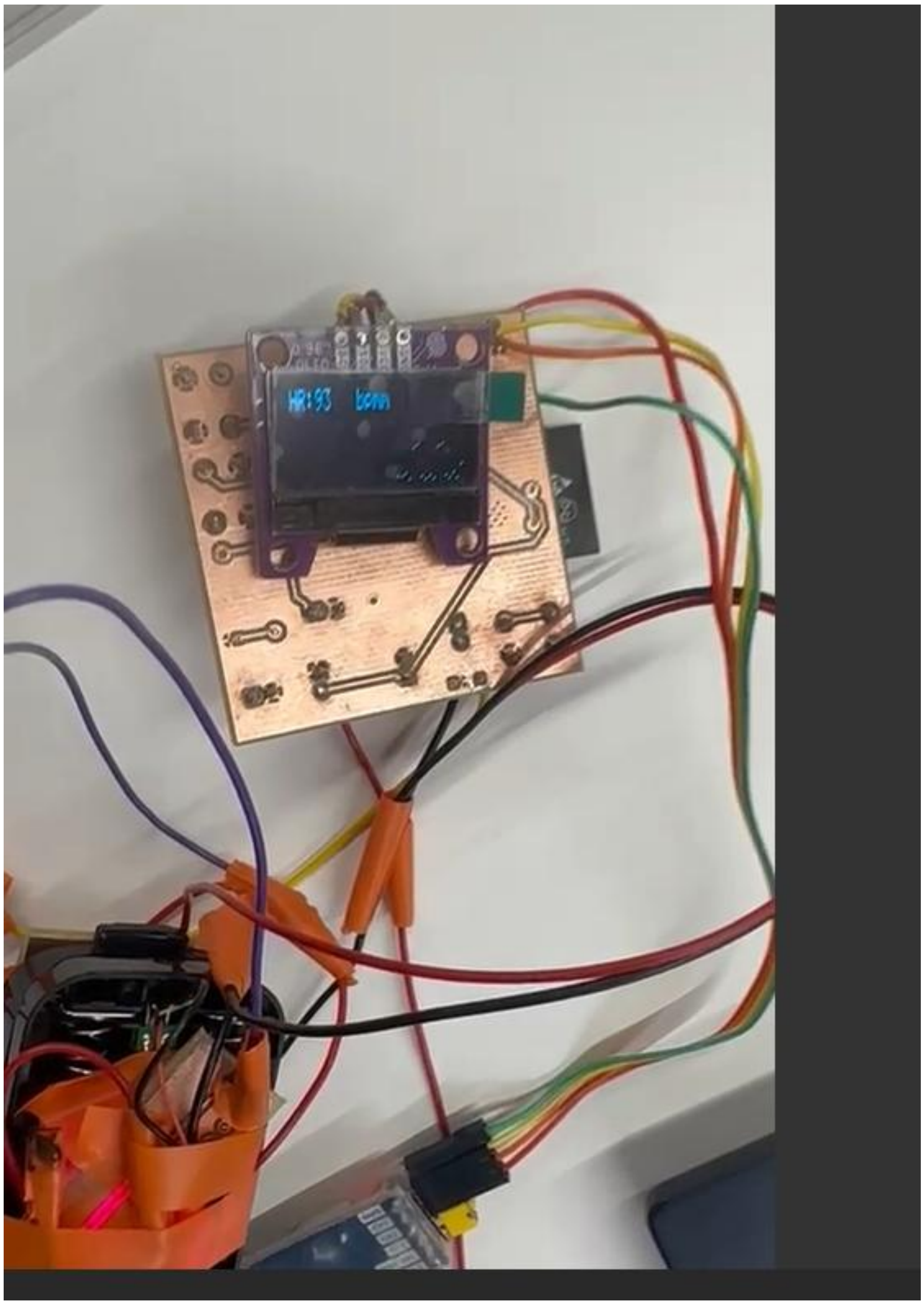
**Technical Evaluation**

- **Reliability:** Consistent readings under stable conditions; minor noise under motion.
- **Usability:** Compact, low-cost prototype suitable for wearable or portable use.
- **Limitations:** No $SPO_2$ feature implemented; signal filtering could be improved.
- **Scalability:** Modular design allows future addition of cloud integration or multi-sensor support.

The ESP32 proved to be an efficient and capable platform for handling both data acquisition and signal processing, eliminating the need for a separate DSP unit.



(Image: Built functional prototype)



(Image: Functioning of the device)

## 7 Summary

The goal of this project was to design and implement a real-time heart rate monitoring device using PPG. Initially designed as a dual-controller system with

STM32 and ESP8266, the final version was implemented entirely on ESP32 to simplify the architecture.

**Key Achievements**

- Complete real-time heart rate monitoring using a single ESP32 board.
- Real-time OLED display and Wi-Fi communication capability.
- Stable PPG data acquisition and filtering pipeline.

**Conclusions**

The system demonstrates how low-cost embedded platforms can effectively perform biomedical signal acquisition and processing in real time. Simplified architecture reduces development complexity and cost.

**Future Work**

- Implementation of $SPO_2$ algorithm.
- Improved motion artifact reduction.
- Integration with mobile/cloud health monitoring platforms.
- PCB miniaturization and wearable form factor design.

# 8 References

. **Journal Article:**

Quinn, B.G., 1994. Estimation of frequency, amplitude, and phase from the DFT of a time series. IEEE Transactions on Signal Processing, 42(11), pp. 3279–3291. Available at: https://ieeexplore.ieee.org/document/558515

. **Book:**

Quinn, B.G., 2001. *The Estimation and Tracking of Frequency*. Cambridge University Press. Available at: https://doi.org/10.1017/CBO9780511609602

. **GitHub Repository:**

pointertoaperson, 2025. *sensor_ppg*. GitHub repository. Available at: https://github.com/pointertoaperson/sensor_ppg

## . Similar works:

**a. Journal Article:**

B.M., J. and Holi, M.S., 2015. An estimation technique using FFT for heart rate derived from PPG signal. *Global Journal of Research in Engineering*, 15(F7), pp. 45–51. Available at: https://engineeringresearch.org/index.php/GJRE/article/view/1364

**b. Journal Article:**

Jhuma, F.A., 2024. A hybrid photoplethysmography (PPG) sensor system for health monitoring. *Sensors*, 24(23), pp. 7634. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC11644845/