Department of Electronic and Telecommunication Engineering University of Moratuwa

BM 2210 - Biomedical Device Design Final Report

Team Meditrones

Boralugoda.M.S - 220074B
Liyanage.D.L.B.B - 220362G
Madusanka.S.P.S - 220374U

Contents

1. Introduction

Team Meditrones, a startup dedicated to engineering solutions for the healthcare sector, is driven by a mission to deliver innovative, consumer-focused biomedical technologies that are both affordable and impactful. Grounded in our strategic focus, we aim to address critical healthcare challenges, particularly in underserved communities, while fostering economic growth and positioning Sri Lanka as a leader in medical innovation. Despite facing constraints such as limited resources, rising costs, and a demanding timeframe, our strengths in electronics, collaboration with healthcare professionals, and passion for problem-solving empower us to create solutions that align with our strategic objectives. This final report encapsulates our journey, challenges, and the outcomes of our efforts to develop accessible healthcare technologies.

2. Problem Statement

"There is a need for an affordable and reliable continuous ECG monitor with a mobile app interface in Sri Lankan hospitals to improve access to continuous heart monitoring, enhance cardiac care, and address the limitations of current costly devices."

Cardiovascular diseases (CVDs) remain the leading cause of mortality in Sri Lanka, highlighting the urgent need for accessible and affordable cardiac monitoring solutions. Current options, such as traditional Holter monitors, are prohibitively expensive for many resource-constrained healthcare settings, including government hospitals, limiting their availability and impact on patient care. The following challenges are prevalent in Sri Lanka regarding the Holter monitor:

1. **Limited Availability:** Many hospitals, especially in rural areas, have limited access to Holter monitors, leading to long waiting lists.
2. **Cost Constraints:** The devices and associated software are expensive, making them less accessible for low-income patients.
3. **Maintenance Issues:** High-end devices often require maintenance and calibration, which can be challenging in resource-limited settings.
4. **Patient Compliance:** Patients may find the device uncomfortable or challenging to use, particularly in hot and humid conditions.

In response to this critical gap, our start-up has developed a low-cost Holter monitor integrated with a mobile/desktop app interface, specifically designed to address the needs of Sri Lankan hospitals.

While the low-cost Holter monitor design may involve trade-offs in accuracy and usable time range, it offers a practical solution for preliminary screening. This innovation aims to significantly reduce the waiting list for Holter monitors in Sri Lanka, addressing the urgent need for accessible cardiac monitoring.

3. Methodology

3.1. Block diagram

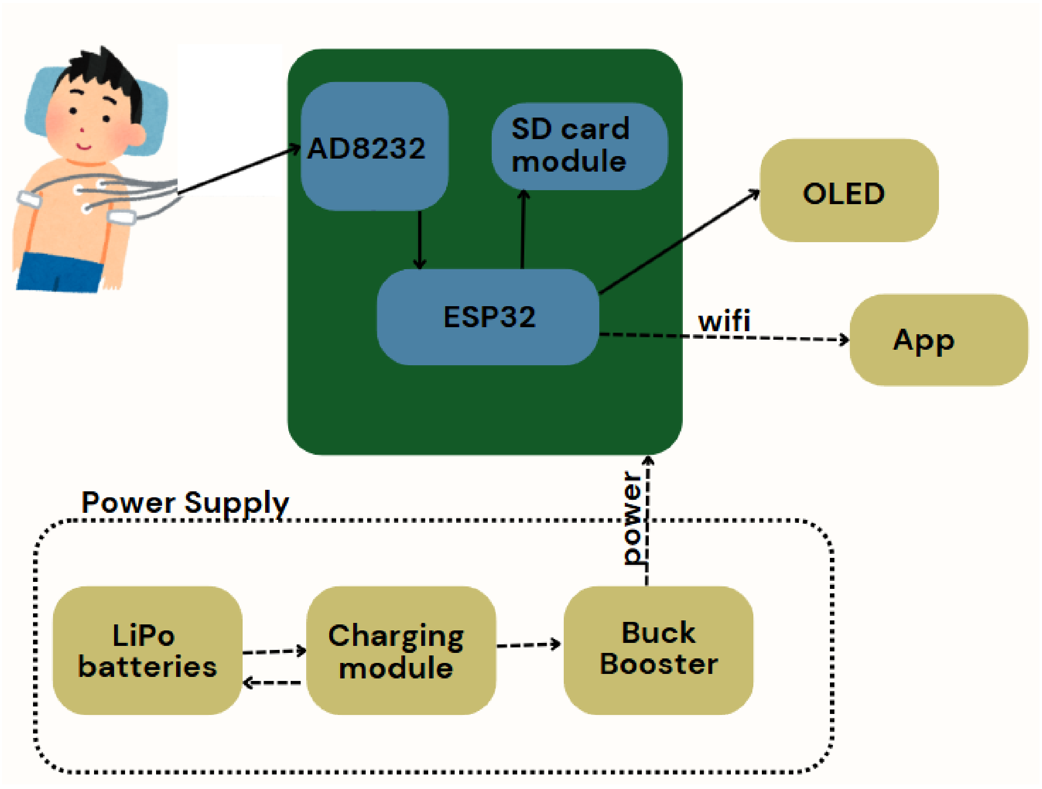


Figure 1: functional block diagram

To address the need for a low-cost Holter monitor, we employed a modular design approach, leveraging off-the-shelf components to minimize development time and production costs. The key components of our device include the AD8232 ECG module, ESP32 microcontroller, and an SD card module. This combination provides a reliable and scalable framework for continuous cardiac monitoring.

3.2. ECG Signal Acquisition

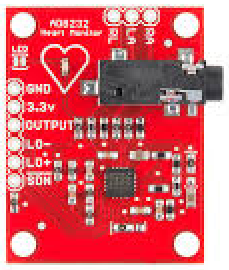


Figure 2: AD8232 module

The AD8232 module is used to capture and preprocess ECG signals, ensuring accurate and efficient detection of cardiac activity while maintaining a compact form factor.

3.2.1. Justification for Using the AD8232

- **Fully Integrated Single-Lead ECG Front End:** Simplifies the circuit design, making it compact and cost-effective for a low-cost Holter monitor.
- **Low Supply Current (170 μA typical):** Reduces power consumption, extending battery life and making the device suitable for continuous monitoring.
- **High Common-Mode Rejection Ratio (80 dB, dc to 60 Hz):** Ensures reliable ECG signal acquisition by effectively suppressing noise, which is critical in real-world hospital or home environments.
- **High Signal Gain (G = 100) with DC Blocking Capabilities:** Amplifies the weak ECG signals while eliminating baseline wander, ensuring clear and accurate waveform detection.
- **Adjustable High-Pass and Low-Pass Filters:** Enables tuning to remove motion artifacts and high-frequency noise, enhancing signal quality.
- **Fast Restore Feature:** Quickly stabilizes the signal after electrode reattachment or transient events, ensuring uninterrupted monitoring.

- **Accepts ± 300 mV of Half-Cell Potential:** Compensates for variations in electrode-skin interface potentials, ensuring consistent signal capture over time.
- **Leads-Off Detection (AC or DC Options):** Provides real-time notification of electrode disconnection, ensuring patient safety and continuous operation.
- **Integrated Right Leg Drive (RLD) Amplifier:** Minimizes common-mode interference, improving the quality of ECG signals.
- **Integrated Reference Buffer (Virtual Ground):** Simplifies circuit design and reduces external component count, contributing to a cost-effective solution.
- **Internal RFI Filter:** Mitigates interference from mobile phones and other RF sources, crucial for maintaining signal integrity in real-world conditions.
- **Compact Package (20-lead, 4 mm $\times$ 4 mm LFCSP):** Reduces PCB footprint, allowing for a portable and lightweight design essential for Holter monitors.
- **Qualified for Automotive Applications:** Demonstrates reliability and suitability for high-performance, mission-critical applications, ensuring dependability in medical-grade devices.

By leveraging the AD8232's advanced features, we ensure that the low-cost Holter monitor provides accurate, reliable, and user-friendly cardiac monitoring, addressing the critical needs of resource-constrained healthcare facilities.

3.3. Data Processing and Transmission

Figure 3: ESP32 dev board

The ESP32 WROOM microcontroller serves as the core of the system, handling data processing and enabling wireless connectivity. Its built-in Wi-Fi and Bluetooth capabilities facilitate real-time data transmission to the accompanying mobile app, ensuring accessibility for healthcare providers.

3.3.1. Justification for Using the ESP32

- **Low Power Consumption:**
  - Deep sleep modes (10 µA) for long battery life.
  - Suitable for continuous, battery-powered ECG monitoring.
- **Dual-Core Processor:**
  - High processing power for real-time ECG signal processing.
  - Efficient data handling for ECG analysis and transmission.
- **Wireless Connectivity (Wi-Fi & Bluetooth):**
  - Wi-Fi for cloud data transmission (e.g., Ubidots, MQTT).
  - Bluetooth Low Energy (BLE) for smartphone or device communication.

- **Multiple I/O Pins & Communication Interfaces:**
  - 18 ADC channels for analog ECG signal input.
  - SPI, I2C, UART for peripheral integration.
- **High Integration:**
  - Built-in flash memory (4MB+) and RAM, reducing external component needs.
  - Simplifies design and reduces costs.
- **Security Features:**
  - Secure boot and flash encryption for data integrity and privacy.
  - SSL/TLS for secure communication.
- **Compact Form Factor:**
  - Small and lightweight, ideal for portable ECG monitoring devices.
- **Cost-Effective:**
  - Affordable solution, aligning with the goal of a low-cost Holter monitor.
- **Cloud Integration:**
  - Easy integration with cloud platforms for real-time remote monitoring.

By integrating the ESP32 into our Holter monitor, we ensure reliable, real-time data acquisition and transmission at an affordable price point. Its combination of high performance, connectivity, and low cost aligns perfectly with our goal of creating an accessible and efficient cardiac monitoring solution.

3.4. Data Storage

For offline analysis and redundancy, an SD card module is integrated into the system to store ECG data locally. This ensures uninterrupted monitoring even in areas with unstable network connectivity.

3.5. Power Supply

The power supply design for the Holter monitor uses **2 LiPo 1200mAh batteries**, a **USB micro charging module**, and a **buck-boost converter**, each selected for their key benefits.

The **LiPo batteries** are chosen for their **high energy density**, providing a compact and lightweight power solution with sufficient capacity for extended operation. Being **rechargeable**, they are both cost-effective and environmentally friendly.

The **USB micro charging module** enables **easy and convenient charging** using common USB ports, which is highly user-friendly for both patients and healthcare providers.

The **buck-boost converter** ensures **stable voltage regulation**, providing consistent power even as the battery voltage fluctuates. This is crucial for the reliable operation of the monitor's components. The converter is also **energy efficient**, reducing power losses and extending battery life, and its **wide input voltage range** (from 3.0V to 4.2V) accommodates the varying charge levels of the LiPo batteries. This setup minimizes the need for multiple regulators, making the system more compact and efficient.

Overall, this power supply configuration ensures long-lasting, reliable, and easy-to-manage power for the Holter monitor, making it ideal for continuous, portable use in healthcare settings.

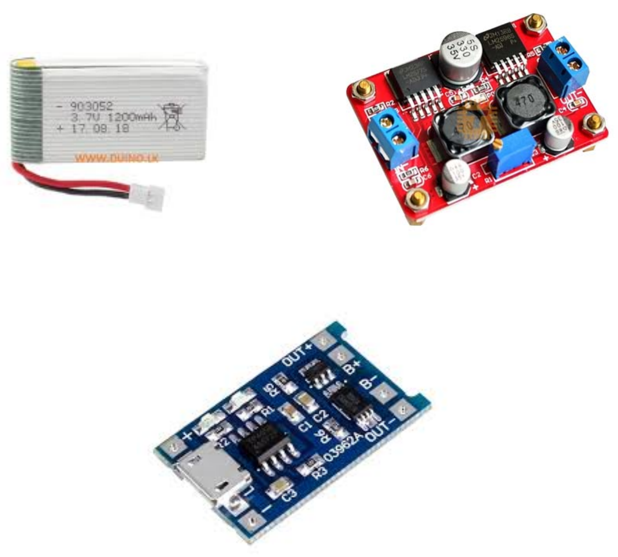


Figure 4: Power supply components

3.6. Mobile Application for the User

We created a simple and a user friendly application for the health professionals and the user to easily observe the ECG waveform after a certain time period and this received data is stored in the cloud for future usage.
This app is made using a popular app development language called ”React” and the user interface of the app is shown below: (Note that all the software codes are attached to the appendix at the end of the report)

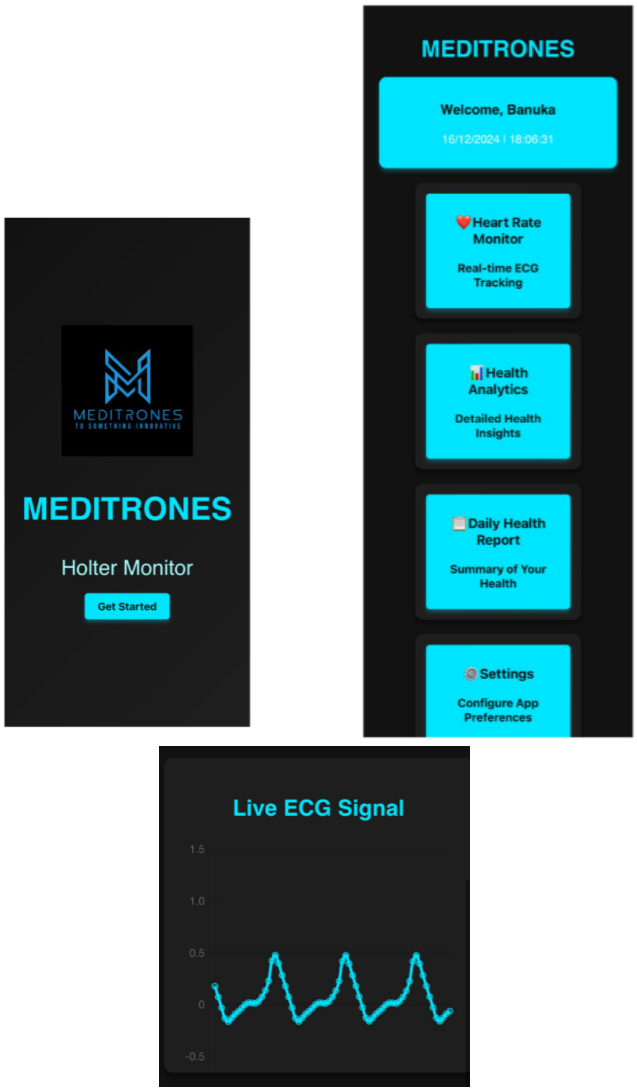


Figure 5: Mobile App UI

4. Schematics and Design

4.1. Schematic diagram

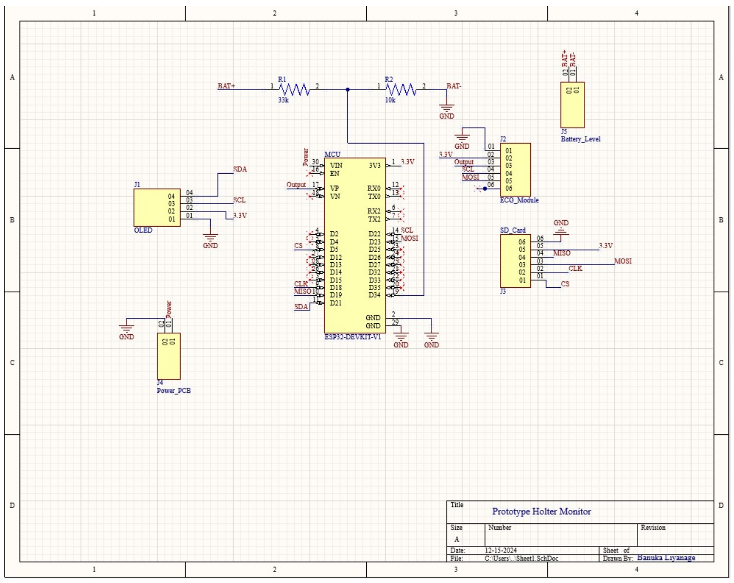


Figure 6: Schematic diagram

4.2. PCB design

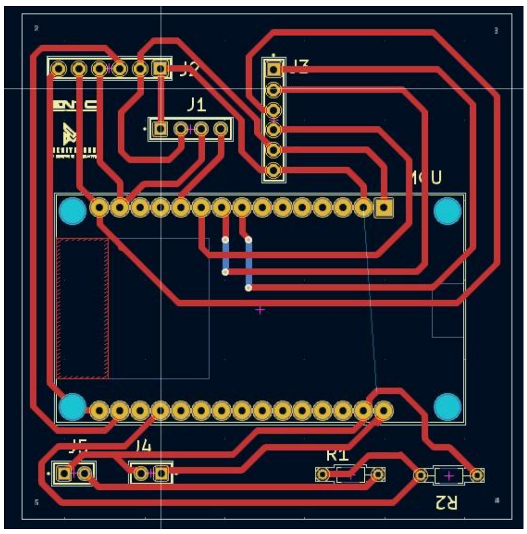


Figure 7: PCB routing

4.3. Enclosure design

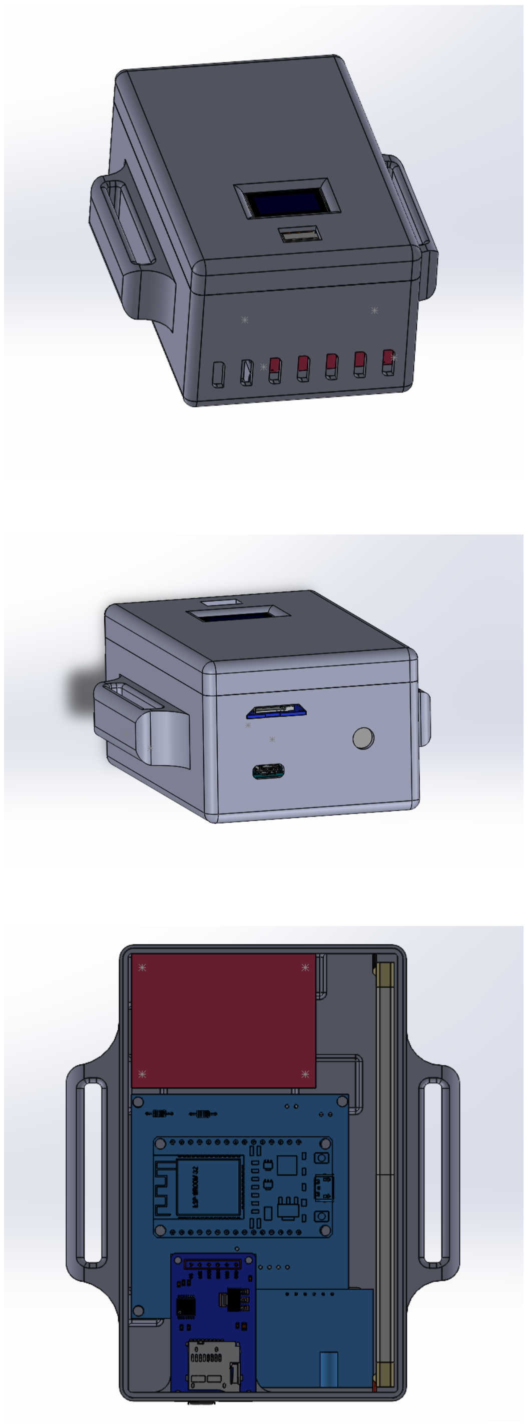


Figure 8: Enclosure

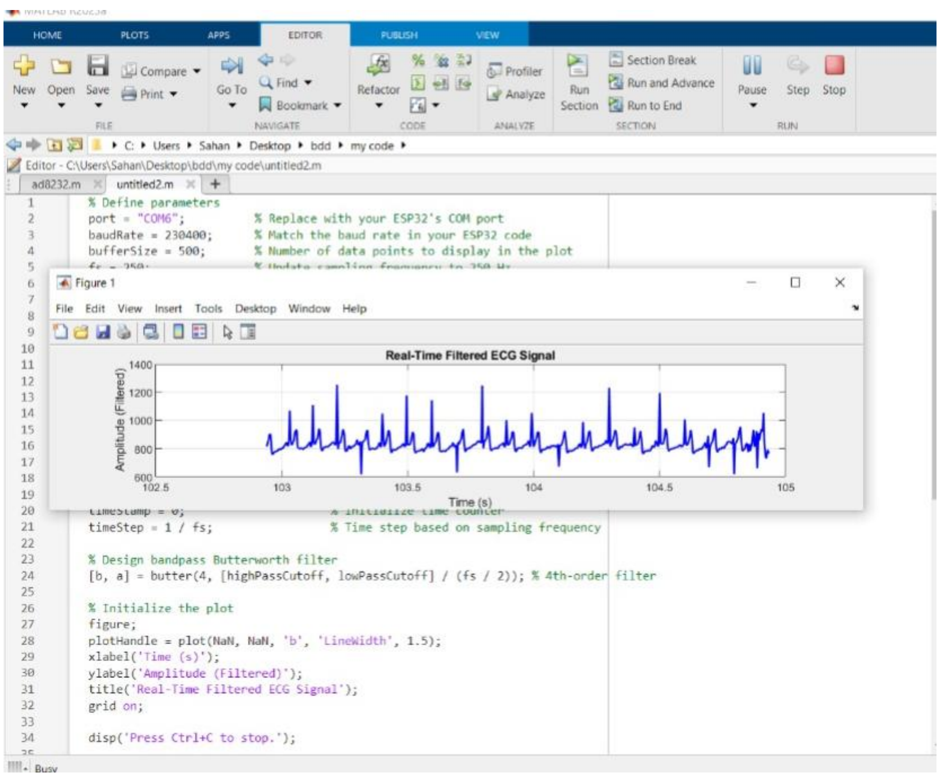
5. Results

The primary objective of the project was to drastically reduce the cost of Holter monitors, as the current market price in Sri Lanka ranges between LKR 250,000 and LKR 600,000, making them inaccessible for many. The developed low-cost Holter monitor, priced at just LKR 10,000, successfully performs real-time single-lead ECG monitoring with data storage and Wi-Fi connectivity for remote transmission. Following is the breakdown of our devices' costs:

Item	**Unit Cost (LKR)**	**Quantity**	**Total Cost (LKR)**
AD8232 Module	1590	1	1590
ESP32-WROOM	1250	1	1250
Micro SD Card	160	1	160
USB Charging Circuit	60	1	60
Buckbooster	1050	1	1050
LiPo Battery	910	2	1820
Enclosure	1500	1	1500
Other Miscellaneous Costs	2570	-	2570
Total Cost			**10,000**

Table 1: Budget for the Holter Monitor Project

Through testing, we were able to achieve acceptable accuracy in ECG signal acquisition and recording, meeting the fundamental requirements of a Holter monitor. The system successfully captured single-lead ECG data using the AD8232 module and reliably stored it on the SD card via the ESP32 microcontroller, making it a viable and affordable solution for addressing the significant healthcare accessibility challenges in rural and underserved areas. This innovation has the potential to alleviate the financial and operational burdens of cardiac monitoring in Sri Lanka.



```
% Define parameters
port = "COM6";          % Replace with your ESP32's COM port
baudRate = 230400;      % Match the baud rate in your ESP32 code
bufferSize = 500;       % Number of data points to display in the plot
fs = 250;               % Update sampling frequency to 250 Hz
```

```
timeStamp = 0;                      % Initialize time counter
timeStep = 1 / fs;                  % Time step based on sampling frequency

% Design bandpass Butterworth filter
[b, a] = butter(4, [highPassCutoff, lowPassCutoff] / (fs / 2)); % 4th-order filter

% Initialize the plot
figure;
plotHandle = plot(NaN, NaN, 'b', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Amplitude (Filtered)');
title('Real-Time Filtered ECG Signal');
grid on;

disp('Press Ctrl+C to stop.');
```

Figure 9: Testing results

6. Conclusion

The project began with a structured approach, starting with needs finding and needs screening to identify the key challenges in Sri Lanka's healthcare system. Based on these insights, idea generation followed, leading to the development of a low-cost Holter monitor design. Concept selection was then carried out to choose the most suitable solution, balancing cost-effectiveness, functionality, and accessibility.

Using affordable components like the AD8232 ECG module, ESP32 microcontroller, and Li-Po batteries, the device provides essential features such as real-time heart rate monitoring, data storage, and cloud integration. These features make it a valuable tool for preliminary cardiac screening and long-term ECG monitoring.

This project demonstrates the potential to bridge gaps in Sri Lanka's healthcare system by providing an accessible solution for preliminary cardiac screening. With further development, this low-cost Holter monitor could significantly reduce waiting times and improve cardiac care, especially in underserved regions of the country.

7. Possible Future Improvements

1. **Enhanced Battery Life**
   The current dual 1200 mAh Li-Po battery setup provides adequate power for short-term use. Future improvements could include higher-capacity batteries for extended runtime, a smart Battery Management System (BMS) to optimize performance and ensure safe operation, and energy-harvesting technologies such as solar or piezoelectric systems to enhance sustainability. Replacement of the dual battery setup with a single high capacity battery could also simplify the design, reduce weight, and improve portability, making the device more efficient and user-friendly.

2. **Improved Signal Quality**
   Advanced noise filtering techniques or adaptive signal processing algorithms can be implemented to improve the quality of ECG signals, particularly in environments with high motion or electrical interference.

3. **Compact and Lightweight Design**
   Due to time constraints, we designed a single-layer PCB that could be manufactured locally in Sri Lanka. However, future iterations can use multilayer PCBs to reduce the device's size and improve component layout. Additionally, instead of using preassembled modules, individual chips could be directly integrated onto the PCB, optimizing the design for compactness and cost-effectiveness. This would improve patient comfort, particularly during prolonged periods of use.

4. **Integration with AI-based Analytics**
   Adding machine learning-based algorithms for real-time arrhythmia detection, trend analysis, and automated diagnostic reports can significantly improve the device's utility for healthcare professionals, making it a valuable tool for advanced diagnostics.

8. References

For more details, refer to the following datasheets:

- **AD8232**: Analog Devices, "AD8232 Single-Lead Heart Rate Monitor IC," *Datasheet*, 2017. Available: `https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf`.
- **ESP32-WROOM-32**: Espressif Systems, "ESP32-WROOM-32: Integrated Wi-Fi and Bluetooth SoC," *Datasheet*, 2020. Available: `https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf`.

9. Appendix

9.1. MATLAB Code for Filtering

The following MATLAB code is used to design a real-time ECG signal filtering system:

Listing 1: MATLAB Code for ECG Signal Filtering

```
% Define parameters
port = "COM6";          % Update with your actual COM port
baudRate = 230400;      % Baud rate to match Arduino/ESP32
fs = 250;               % Sampling frequency (Hz)
lowPassCutoff = 100;     % Low-pass filter cutoff frequency in
    Hz
highPassCutoff = 0.5; % High-pass filter cutoff frequency in
    Hz
bufferSize = 700;       % Number of points for the real-time
    plot

% Sensor Gain
sensorGain = 1100;      % Gain factor of the AD8232 module

% Create serialport object
device = serialport(port, baudRate);

% Allocate storage for data
timeBuffer = zeros(bufferSize, 1);
ecgBuffer = zeros(bufferSize, 1);

% Design bandpass Butterworth filter (0.5 Hz - 70 Hz)
[b, a] = butter(4, [highPassCutoff, lowPassCutoff] / (fs /
    2));
```

```matlab

% Initialize the plot
figure;
plotHandle = plot(NaN, NaN, 'b', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('Real-Time ECG Signal');
grid on;

% Open a file to log raw data (optional)
fileID = fopen('raw_ecg_data.txt', 'a'); % Append mode

disp('Press Ctrl+C to stop.');

try
    % Initialize buffers for real-time plotting
    timeStamp = 0; % Initialize the time counter
    while true
        % Read data from the serial port
        dataLine = readline(device);

        % Parse the incoming data (format: time,raw_value)
        parsedData = sscanf(dataLine, '%f,%f');
        if numel(parsedData) == 2
            timeStamp = parsedData(1); % Extract time
            rawValue = parsedData(2); % Extract raw ADC value

            % Display raw data in Command Window
            disp(['Raw Value: ', num2str(rawValue), ' |
                Time: ', num2str(timeStamp)]);

            % Log raw data to file
            fprintf(fileID, 'Time: %.4f, Raw Value: %d\n',
                timeStamp, rawValue);

            % Normalize the raw ECG signal by the sensor gain
            normalizedSignal = rawValue / sensorGain;

            % Filter the normalized ECG signal
            filteredSignal = filter(b, a, normalizedSignal);

            % Update buffers
            timeBuffer = [timeBuffer(2:end); timeStamp];
            ecgBuffer = [ecgBuffer(2:end); filteredSignal];

            % Update the plot
            set(plotHandle, 'XData', timeBuffer, 'YData',
                ecgBuffer);
            drawnow;
        end
    end
catch ME
    disp('Terminating...');
    disp(ME.message);
    fclose(fileID);  % Close the file
    clear device;    % Clean up the serial port
end
```

9.2. React Code for Mobile Application

Below is the React code for creating a mobile application with real-time ECG display:

Listing 2: React Code for Mobile Application

```
import React, { useState, useEffect } from "react";
import { Line } from "react-chartjs-2";
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
} from "chart.js";

ChartJS.register(CategoryScale, LinearScale, PointElement,
    LineElement);

function App() {
  const [showWelcome, setShowWelcome] = useState(true);
  const [ecgData, setEcgData] = useState([]);
  const [timeLabels, setTimeLabels] = useState([]);

  // Simulate realistic PQRST ECG waveform
  const pqrstWaveform = [
    0, 0.1, 0.5, 1.0, 0.5, 0.1, 0, -0.2, -0.4, -0.2, 0, 0,
        0, 0.05, 0.1, 0.05,
    0,
  ];

  useEffect(() => {
    let index = 0;
    const ecgInterval = setInterval(() => {
      setEcgData((prev) => [...prev.slice(-50),
          pqrstWaveform[index]]);
      setTimeLabels((prev) => [
        ...prev.slice(-50),
        new Date().toLocaleTimeString(),
      ]);
      index = (index + 1) % pqrstWaveform.length;
    }, 100);
    return () => clearInterval(ecgInterval);
  }, []);

  const ecgChartData = {
    labels: timeLabels,
    datasets: [
      {
        label: "ECG Signal",
        data: ecgData,
        borderColor: "#00e5ff",
        backgroundColor: "rgba(0, 229, 255, 0.2)",
        tension: 0.4,
      },
    ],
  };

  const handleStart = () => {
    setShowWelcome(false);
  };

  return (
    <div style={{ fontFamily: "'Poppins', sans-serif" }}>
      {showWelcome ? (
        // Welcome Page
        <div style={styles.welcomeContainer}>
          <img
            src="https://i.imgur.com/ItUJ8cu.jpeg"
            alt="Meditrones Logo"
            style={{ width: "200px", marginBottom: "20px" }}
          />
          <h1 style={styles.logoText}>MEDITRONES</h1>
          <p style={styles.tagline}>Holter Monitor</p>
          <button onClick={handleStart}
             style={styles.button}>
            Get Started
          </button>
        </div>
      ) : (
        // Main App Page
        <div style={styles.mainContainer}>
          <h1 style={styles.mainTitle}>MEDITRONES</h1>
          <div style={styles.cardContainer}>
            <div style={styles.welcomeCard}>
              <h3>Welcome, Banuka</h3>
              <p style={{ color: "#fff" }}>
```

```
                {new Date().toLocaleDateString()} |{" "}
                {new Date().toLocaleTimeString()}
              </p>
            </div>
            <div style={styles.card}>
              <button onClick={handleStart}
                 style={styles.button}>
              <h3>  Heart   Rate Monitor</h3>
              <p>Real-time ECG Tracking</p>
              </button>
            </div>
            <div style={styles.card}>
              <h3>  Health   Analytics</h3>
              <p>Detailed Health Insights</p>
            </div>
            <div style={styles.card}>
              <h3>  Daily   Health Report</h3>
              <p>Summary of Your Health</p>
            </div>
            <div style={styles.card}>
              <h3>  Settings   </h3>
              <p>Configure App Preferences</p>
            </div>
          </div>

          <div style={styles.ecgContainer}>
            <h2 style={{ textAlign: "center", color:
               "#00e5ff" }}>
              Live ECG Signal
            </h2>
            <Line
              data={ecgChartData}
              options={{
                responsive: true,
                maintainAspectRatio: false,
                scales: {
                  x: { display: false },
                  y: { min: -1, max: 1.5 },
                },
                plugins: { legend: { display: false } },
              }}
            />
          </div>
        </div>
      )}
    </div>
  );
}

```

```
const styles = {
  welcomeContainer: {
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    alignItems: "center",
    height: "100vh",
    background: "linear-gradient(135deg, #121212, #1f1f1f)",
    color: "#fff",
    textAlign: "center",
  },
  logoText: {
    fontSize: "3rem",
    fontWeight: "700",
    marginBottom: "10px",
    color: "#00e5ff",
  },
  tagline: {
    fontSize: "2rem",
    marginBottom: "20px",
    color: "#aff",
  },
  button: {
    backgroundColor: "#00e5ff",
    color: "#121212",
    padding: "10px 20px",
    border: "none",
    borderRadius: "5px",
    fontSize: "1rem",
    cursor: "pointer",
    fontWeight: "bold",
    boxShadow: "0 4px 6px rgba(0, 229, 255, 0.3)",
  },
  mainContainer: {
    backgroundColor: "#121212",
    minHeight: "100vh",
    padding: "20px",
    color: "#fff",
  },
  mainTitle: {
    textAlign: "center",
    color: "#00e5ff",
    marginBottom: "20px",
  },
  cardContainer: {
    display: "flex",
    justifyContent: "center",
    flexWrap: "wrap",
    gap: "20px",
  },
  card: {
    backgroundColor: "#1e1e1e",
    padding: "15px",
    borderRadius: "10px",
    boxShadow: "0 4px 6px rgba(0, 0, 0, 0.5)",
    textAlign: "center",
    width: "200px",
    color: "#ddd",
  },
  welcomeCard: {
    backgroundColor: "#00e5ff",
    color: "#121212",
    padding: "15px",
    borderRadius: "10px",
    boxShadow: "0 4px 6px rgba(0, 229, 255, 0.3)",
    textAlign: "center",
    width: "300px",
  },
  ecgContainer: {
    marginTop: "30px",
    backgroundColor: "#1e1e1e",
    padding: "20px",
    borderRadius: "10px",
    boxShadow: "0 4px 6px rgba(0, 0, 0, 0.5)",
    height: "300px",
  },
};

export default App;
```

9.3. ESP32 Code for Real-Time ECG Monitoring

The ESP32 code facilitates Wi-Fi communication and reads ECG data from the AD8232 sensor:

Listing 3: ESP32 Code for ECG Monitoring

```
#include <WiFi.h>
#include <WebServer.h>
#include <WebSocketsServer.h>

// WiFi Credentials - REPLACE WITH YOUR ACTUAL WIFI DETAILS
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";

// AD8232 Pin Configurations
#define HEART_RATE_PIN 36  // Analog pin for AD8232 output
```

```
#define LO_PLUS 23         // Lead Off Detect Positive Pin
#define LO_MINUS 22         // Lead Off Detect Negative Pin

WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);

void setup() {
  Serial.begin(115200);

  // Configure AD8232 Pins
  pinMode(HEART_RATE_PIN, INPUT);
  pinMode(LO_PLUS, INPUT);
  pinMode(LO_MINUS, INPUT);

  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  // Serve HTML Page
  server.on("/", []() {
    server.send(200, "text/html", htmlPage);
  });
  server.begin();

  // Start WebSocket Server
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
}

void loop() {
  server.handleClient();
  webSocket.loop();

  // Read Heart Rate Sensor
  if ((digitalRead(LO_PLUS) == 1) || (digitalRead(LO_MINUS)
     == 1)) {
    Serial.println("Leads Off!");
  } else {
    int heartRateValue = analogRead(HEART_RATE_PIN);
    webSocket.broadcastTXT(String(heartRateValue));
  }

  delay(50);  // Small delay to prevent overwhelming the network
}

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
  // WebSocket event handler
}

// HTML Page stored in program memory
const char htmlPage[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
    <title>ESP32 Heart Rate Monitor</title>
    <style>
        body { font-family: Arial; text-align: center; }
        #heartRateDisplay {
            font-size: 48px;
            margin: 50px;
            color: red;
        }
        #chartContainer {
            width: 100%;
            height: 300px;
        }
    </style>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h1>Real-Time Heart Rate Monitor</h1>
    <div id="heartRateDisplay">--</div>
    <canvas id="chartContainer"></canvas>

    <script>
        const heartRateDisplay = document.getElementById('heartRateDisplay');
        const ctx = document.getElementById('chartContainer').getContext('2d');

        // Chart Configuration
        const chart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: [],
                datasets: [{
                    label: 'Heart Rate Signal',
                    data: [],
                    borderColor: 'red',
                    backgroundColor: 'rgba(255, 0, 0, 0.1)'
                }]
            },
            options: {
                responsive: true,
                scales: {
                    y: {
                        beginAtZero: false
                    }
                }
            }
        });

        // WebSocket Connection
        const socket = new WebSocket('ws://' +
           window.location.hostname + ':81');

        socket.onmessage = function(event) {
            const heartRateValue = event.data;
            heartRateDisplay.textContent = heartRateValue;

            // Update Chart
            if (chart.data.labels.length > 50) {
                chart.data.labels.shift();
                chart.data.datasets[0].data.shift();
            }

            chart.data.labels.push(new
               Date().toLocaleTimeString());
            chart.data.datasets[0].data.push(heartRateValue);
            chart.update();
        };
    </script>
</body>
</html>
)rawliteral";
```