Single lead mobile ECG machine

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DIPLOMA

IN

Electical Engineering

Submitted by:

SUDIPTA SEN(D222322587)

TANIMA GHOSH(D222322596)

RAJU ADHIKARY(D222322568)

GOUTAM AMBALI(D202116553)

RUMA MANDI(D222322574)

Under the guidance of

Lec.(Mr.) SOUVIK BAG

DEPT. OF ELECTRICAL ENGINEERING

RANAGHAT GOVERNMENT POLYTECHNIC

Ranaghat, Nadia-741201

DEC 2024

DEPT. OF ELECTRICAL ENGINEERING

RANAGHAT GOVERNMENT POLYTECHNIC

Ranaghat, Nadia-741201

CANDIDATE'S DECLARATION

We, SUDIPTA SEN (D222322587), TANIMA GHOSH (D222322596), RAJU ADHIKARY (D222322568), GOUTAM AMBALI (D202116553) & RUMA MANDI (D222322574) students of Diploma (ELECTRICAL ENGINEERING), hereby declare that the Project Dissertation titled – "Single lead mobile ECG machine" which is submitted by us to the Department of Electical Engineering, RANAGHAT GOVERNMENT POLYTECHNIC, West Bengal in fulfillment of the requirement for awarding of Diploma, is not copied from any source without proper citation. This work has not previously formed the basis for the award of any Diploma or other similar title or recognition in our institute.

Place: West Bengal

Date: 2024-12-11

SUDIPTA SEN (D222322587)

RAJU ADHIKARY (D222322568)

RUMA MANDI (D222322574)

TANIMA GHOSH (D222322596)

GOUTAM AMBALI (D202116553)

DEPT. OF ELECTRICAL ENGINEERING

RANAGHAT GOVERNMENT POLYTECHNIC

Ranaghat, Nadia-741201

CERTIFICATE

I hereby certify that the Project titled "Single lead mobile ECG machine" which is submitted by SUDIPTA SEN (D222322587), TANIMA GHOSH (D222322596), RAJU ADHIKARY (D222322568), GOUTAM AMBALI (D202116553) & RUMA MANDI (D222322574) for fulfillment of the requirements for awarding of the Diploma is a record of the project work carried out by the students under my guidance & supervision. To the best of my knowledge, this work has not been submitted in any part or fulfillment for any Degree or Diploma to this Institute or elsewhere.

Place : West Bengal

Date : 2024-12-11

Lec.(Mr.) SOUVIK BAG

(SUPERVISOR)

Lecturer

Department of ELECTRICAL ENGINEERING

RANAGHAT GOVERNMENT POLYTECHNIC

ABSTRACT

Keywords - Portable ECG, Low-Cost Healthcare Technology, Rural Health Diagnostics

This project presents a cost-effective, portable, single-lead ECG (Electrocardiogram) machine designed to meet essential cardiac monitoring needs, particularly in underserved rural or remote areas. Multi-lead ECG machines are often costly and complex, limiting accessibility and usability in such settings. Our single-lead ECG device focuses on affordability, ease of use, and mobility without compromising essential diagnostic functionality.

The proposed system leverages bioinstrumentation amplifiers, precision filters, and a DIY digital storage oscilloscope (DSO) to capture, amplify, and visualize ECG signals. Using Wi-Fi-enabled technology, this prototype also supports data transfer and remote monitoring capabilities. Through systematic design and practical implementation, this project successfully demonstrates real-time ECG signal processing with a user-friendly interface.

In future iterations, we aim to integrate AI-driven health analytics, enhanced wireless capabilities, and expanded outreach for rural healthcare initiatives. This project underscores the potential of affordable healthcare technology to improve access to vital diagnostic tools, addressing a critical gap in global health equity.

ACKNOWLEDGEMENT

The successful completion of any task is incomplete and meaningless without giving any due credit to the people who made it possible without which the project would not have been successful and would have existed in theory.

First and foremost, we are grateful to **Mr. Achanchal Kundu**, HOD, Department of Electical Engineering, RANAGHAT GOVERNMENT POLYTECHNIC, and all other faculty members of our department for their constant guidance and support, constant motivation and sincere support and gratitude for this project work. We owe a lot of thanks to our supervisor, **Mr. Souvik Bag**, Lecturer, Department of Electical Engineering, RANAGHAT GOVERNMENT POLYTECHNIC for igniting and constantly motivating us and guiding us in the idea of a creatively and amazingly performed Major Project in undertaking this endeavor and challenge and also for being there whenever we needed his guidance or assistance.

We would also like to take this moment to show our thanks and gratitude to one and all, who indirectly or directly have given us their hand in this challenging task. We feel happy and joyful and content in expressing our vote of thanks to all those who have helped us and guided us in presenting this project work for our Major project. Last, but never least, we thank our well-wishers and parents for always being with us, in every sense and constantly supporting us in every possible sense whenever possible.

SUDIPTA SEN
(D222322587)

RAJU ADHIKARY
(D222322568)

RUMA MANDI
(D222322574)

TANIMA GHOSH
(D222322596)

GOUTAM AMBALI
(D202116553)

Contents

Candidate's Declaration **i**

Certificate **ii**

Abstract **iii**

Acknowledgement **iv**

List of Figures **vi**

List of Symbols, Abbreviations **vii**

1 INTRODUCTION **1**

2 BACKGROUND **2**

3 PROPOSED DESIGN **4**

3.1 ELECTRODES 4

3.2 AMPLIFIER 5

3.3 NOISE & FILTER 5

3.4 DSO 6

3.5 WIFI 6

4 Software Development **7**

Microcontroller programming Algorithm 7

Web Development Algorithm 8

5 Implementation and Results **9**

6 Future Scope **10**

6.1 Artificial Intelligence (AI) Features 10

6.2 Multi-Lead ECG Option 10
6.3 Improving Signal Quality 10
6.4 Data Logging 11
6.5 Remote Monitoring 11
6.6 Customizable Features 11
6.7 Support for Rural Healthcare Programs 11

7 CONCLUSION 12

Appendices 13

References 24

List of Figures

2.1 The electrical conduction system of the heart 2
2.2 Normal ECG signal pattern 3
2.3 Traditional Hospital ECG Machine 3

3.1 ECG block diagram 4
3.2 DIY reusable electrodes 5
3.3 AD8232 block diagram [2] 5
3.4 DSO block diagram 6

5.1 Project Prototype, implemented the entire circuit inside the box 9
5.2 User interface with Project result or output 9

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

ECG : Electrocardiogram

EEG : Electroencephalogram

CVDs : Cardiovascular Diseases

DIY : Do It Yourself

AD8232 : Bioinstrumentation Amplifier

DSO : Digital Storage Oscilloscope

MCU : Microcontroller Unit

Wi-Fi : Wireless Fidelity

HTTP : Hypertext Transfer Protocol

UI : User Interface

API : Application Programming Interface

ESP8266 : Wi-Fi Module

RC Filter : Resistor-Capacitor Filter

DNS : Domain Name System

IoT : Internet of Things

Technical Terms and Concepts

Bioinstrumentation : Application of electronics and measurement techniques for physiological data.

Signal Amplification : Increasing the amplitude of ECG signals.

Baseline Wander : Low-frequency noise caused by body movements or respiration.

Electromagnetic Interference (EMI) : Disturbances from external electromagnetic sources.

WebSocket Protocol : Real-time full-duplex communication channel.

Telemedicine : Technology-based remote medical diagnostics and monitoring.

Graphical User Interface (GUI) : Visual interface for user interaction.

Sinoatrial Node : The natural pacemaker of the heart.

Real-Time Visualization : Instant display of ECG signals for analysis.

Motion Artifacts : Signal distortions caused by physical movements.

Design Elements

Spiral Copper Electrodes : Custom reusable electrodes for ECG acquisition.

Low-Cost Healthcare Technology : Affordable solutions for underserved areas.

Noise Filtering : Removing unwanted signals from ECG data.

Single-Lead ECG : Measuring electrical activity from one point on the body.

Precision Filters : High-pass and low-pass filters for signal processing.

Wireless Data Transmission : Sending ECG data via Wi-Fi.

Voltage Per Division (Volt/Div) : Scaling unit for voltage on an oscilloscope.

Time Per Division (Time/Div) : Scaling unit for time on an oscilloscope.

Advanced and Future Concepts

AI-Driven Analytics : Artificial Intelligence for automated diagnostics.

Health Equity : Ensuring equal access to medical resources.

Signal Processing : Techniques for improving signal quality.

Remote Monitoring : Analyzing health data from a distance.

Data Visualization Algorithms : Computational methods for graphical data display.

Chapter 1

INTRODUCTION

In today's world, cardiovascular diseases (CVDs) are one of the leading causes of mortality, claiming millions of lives every year. Early detection and timely monitoring of heart health are crucial for managing these conditions effectively. However, in many rural and remote areas, access to advanced healthcare facilities is limited, leaving a significant portion of the population at risk of undiagnosed heart conditions. Traditional multi-lead ECG machines, while effective, are expensive, bulky, and require trained professionals to operate [3]. These challenges create a pressing need for an affordable, portable, and easy-to-use solution that can bring basic cardiac monitoring to the doorstep of every individual.

Imagine a scenario: A middle-aged farmer working in a rural area experiences occasional chest discomfort but dismisses it due to lack of access to medical facilities. One day, he suffers a major heart event because his condition went undetected for years. This is not just an isolated case but a recurring reality in underserved regions. What if there was a device that could have detected his heart condition earlier, allowing timely intervention?

This project aims to address this critical gap by developing a single-lead, portable ECG machine. Our solution is compact, cost-effective, and user-friendly, making it ideal for rural and low-resource settings. The device is built using innovative DIY techniques, such as custom spiral copper wire electrodes, and powered by the AD8232 bioinstrumentation amplifier for reliable signal processing [1]. The ESP8266 module enables wireless data transmission to any Wi-Fi-enabled device, offering a seamless user experience.

Unlike traditional ECG systems, this project focuses on simplicity without compromising on diagnostic capabilities. The captured ECG data is displayed on a web interface in real-time, making it accessible to both healthcare professionals and individuals for personal monitoring. With its focus on affordability, mobility, and functionality, this device has the potential to revolutionize cardiac care in rural areas, promoting better health outcomes.

Chapter 2

BACKGROUND

The heart is mainly a muscle that requires a stimulus in order to beat and pump blood into the body. This stimulus comes in the form of electric pulses emitted by the sinoatrialnode, a grouping of specialised cells located in the top right hand corner of the heart. Pulses are produced periodically thus maintaining the beating of the heart. They can, - as it will be explained later in the report - if captured and plotted,help physicians diagnose the heart they originated from. The Electrocardiogram fulfils this purpose and has been used to do so for more than a century.

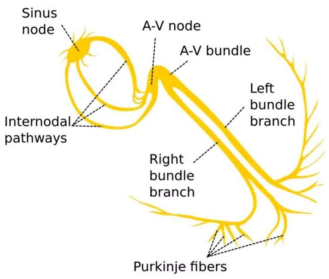

Figure 2.1: The electrical conduction system of the heart

The electrical pulse generated has a unique pattern (seen in Figure 2.2), it includes many significant waves that concern different parts of the heart [5].

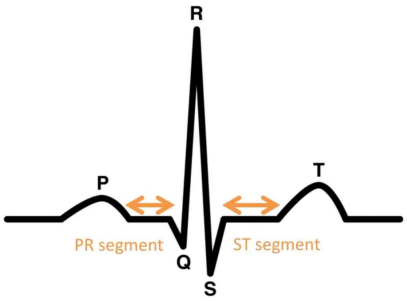

Figure 2.2: Normal ECG signal pattern

Multi-lead ECG machines are commonly used in medical settings but have notable limitations for rural healthcare. They are not only costly but also complex to operate, maintain, and interpret. This complexity, combined with high expenses, restricts accessibility in rural or remote areas where resources and trained personnel are scarce. Recognizing these barriers, our project focuses on creating a simplified, single-lead ECG device that can make essential cardiac monitoring accessible to everyone.

Figure 2.3: Traditional Hospital ECG Machine

Chapter 3

PROPOSED DESIGN

After having completed the background research on the type of signal that will have to be picked up by the ECG monitor to be built, it was decided that the project's ECG monitor would be a two or three electrode ECG powered by one 3.7 V battery. A few design specifications were set in order to ensure proper ECG signal sampling.

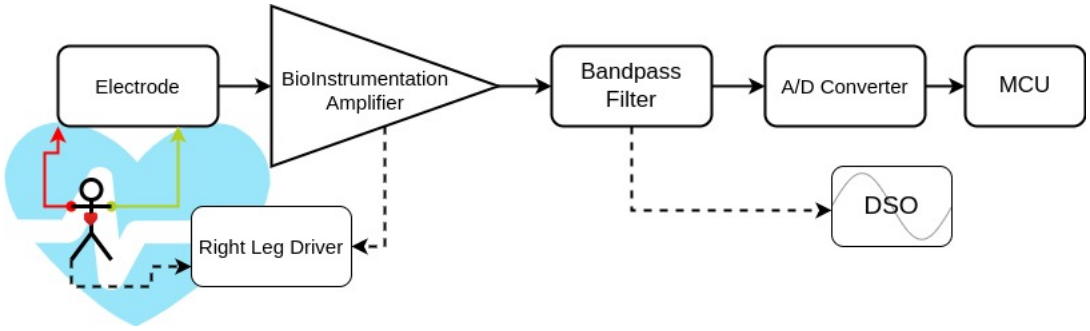

Figure 3.1: ECG block diagram

3.1 ELECTRODES

To reduce costs and improve accessibility, we designed custom electrodes using copper wires bent into a spiral shape (see in Figure 3.2) approximately the size of a finger. This spiral structure ensures adequate skin contact, allowing for stable electrical signal acquisition. These DIY electrodes are reusable, simple to construct, and effective for single-lead ECG applications.

Figure 3.2: DIY reusable electrodes

3.2 AMPLIFIER

We use the AD8232 [2] bioinstrumentation amplifier, a reliable choice for portable ECG devices. This amplifier is designed to capture the heart's weak electrical signals and amplify them for accurate processing. It offers high precision, low power consumption, and effective signal enhancement, making it suitable for mobile applications where battery efficiency and signal clarity are crucial.

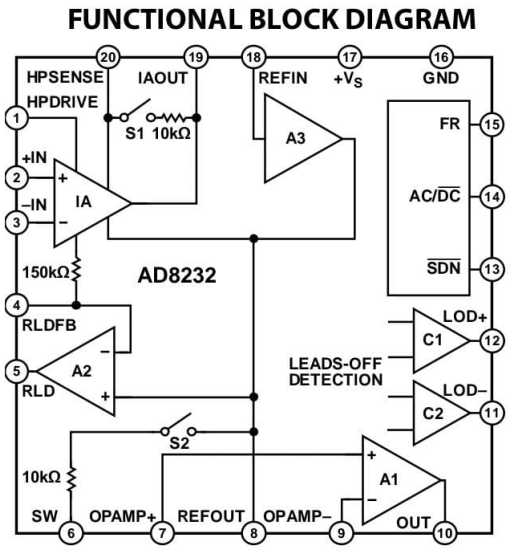

Figure 3.3: AD8232 block diagram [2]

3.3 NOISE & FILTER

Noise in the ECG system can originate from multiple sources, such as electromagnetic interference from power lines, motion artifacts due to user movement, and baseline wander

caused by respiration. The AD8232 module integrates built-in filters to address high-pass and low-pass noise, reducing complexity. For additional filtering, an RC filter can be added at critical stages to further attenuate power line noise or other interference [8].

3.4 DSO

We have implemented a DIY Digital Storage Oscilloscope using the ESP8266 [4] microcontroller with WiFi capabilities. This module captures the processed ECG signal and allows for real-time visualization in any of your wifi enabled device. Unlike traditional DSOs, this DIY approach is cost-effective, compact, and customizable. The ESP8266's WiFi functionality also supports remote monitoring, enabling healthcare providers to view ECG data without physical contact—a significant advantage for rural and telemedicine applications.

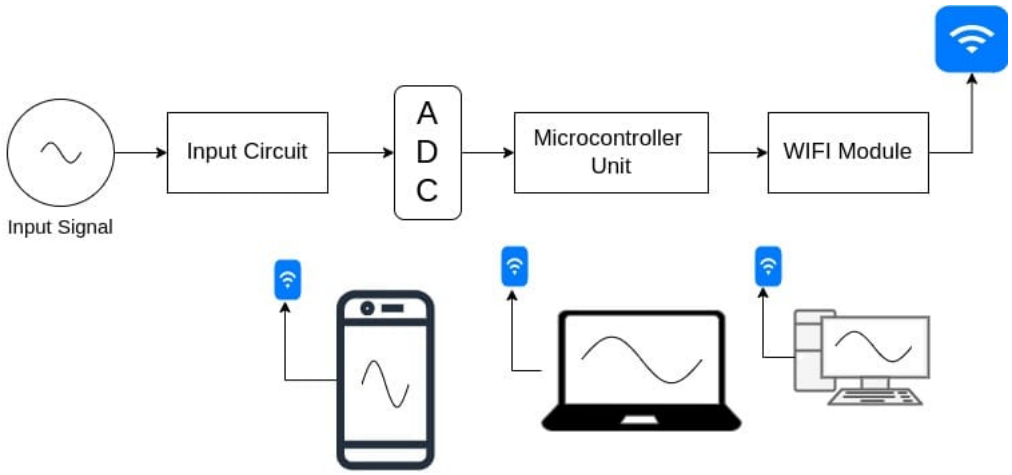

Figure 3.4: DSO block diagram

3.5 WIFI

The ESP8266 [4] MCU provides built-in Wi-Fi connectivity for transmitting ECG data to nearby devices. The system take advantage of HTTP or WebSocket protocols for efficient real-time data streaming. This ensures users can view ECG signals wirelessly on a smartphone, PC or any other wifi enabled device with working browser, enabling mobile diagnostics.

Chapter 4

Software Development

Capture analog ECG signals in MCU, process them, and transmit data wirelessly to a user interface maybe a webpage for real-time visualization [6].

Algorithm 1 Microcontroller programming Algorithm

Initialize:
Initialize Serial Communication for debugging
Initialize Wi-Fi as Access Point
Configure DNS Server
Configure WebSocket Server
Configure Web Server Routes
while True **do**
  Handle WebSocket Events:
  if Message Type = `WStype_TEXT` **then**
    Extract message data
    Parse the message
    Collect sensor readings
    Prepare and send response to client
  end if
  if Message Type = `WStype_CONNECTED` **then**
    Log connection details
  end if
  if Message Type = `WStype_DISCONNECTED` **then**
    Log disconnection details
  end if
end while

Algorithm 2 Web Development Algorithm (User Interface)

Initialize:
  Initialize an empty buffer `data_buffer` with a predefined size (e.g., 2000)
  Set up WebSocket connection to receive incoming data from the MCU [7]
On WebSocket Connection Open:
  Display success notification "Connection established!"
  Send a signal (1) to the WebSocket server to request data
On Receiving Data:
  Parse the incoming data
  Append the received data to `data_buffer`
  Trim the `data_buffer` to maintain a fixed length (e.g., the number of elements corresponding to the canvas width)
  Send a signal (200) to the server indicating successful data reception
Auto Scale Function:
  Calculate the scale factor for voltage (`volt_per_div`) using the difference between the maximum and minimum values in `data_buffer`
  Adjust the vertical position (`positionY`) to center the data values in the middle of the canvas
  Update the UI with the new scale factor and position values
Graph Drawing Loop:
  Clear the canvas area to prepare for a new drawing
  Call the function to draw graph axes
  Begin plotting the graph:
    For each point in `data_buffer`:
      Calculate the corresponding x and y coordinates on the canvas based on the time per division (`time_per_div`) and voltage per division (`volt_per_div`)
      Plot the calculated point on the canvas
    End the line when the graph reaches the end of the canvas
Draw Graph Axes:
  Draw horizontal and vertical grid lines based on the time and voltage divisions
  Label the axes with appropriate markers (e.g., time and voltage values)
Resize Event Handling:
  Adjust the canvas width and reset the data buffer size based on the screen width
Touch Event Handling:
  Track touch input on the canvas and update the pointer position based on user interaction (for example, tracking voltage or time)
Notification System:
  Display WebSocket status notifications (success, error, warning) with appropriate colors and auto-hide functionality

Chapter 5

Implementation and Results

Figure 5.1: Project Prototype, implemented the entire circuit inside the box

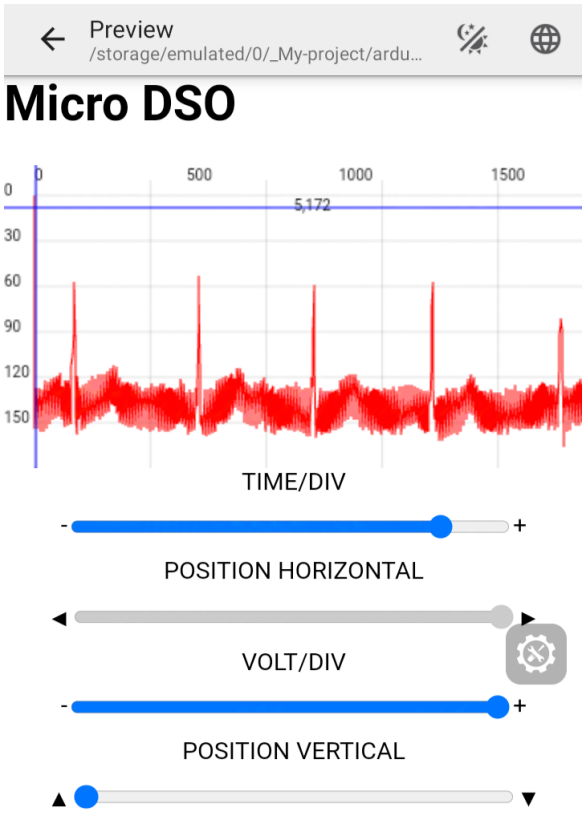

Figure 5.2: User interface with Project result or output

Chapter 6

Future Scope

The portable single-lead ECG machine is a big step toward making heart health check-ups available to everyone, especially in areas where healthcare is hard to access. While the current design works well for basic monitoring, there is a lot of room to improve and add new features in the future. Below are some ideas for what could be done next:

6.1 Artificial Intelligence (AI) Features

Adding AI to the device can help it automatically detect heart problems like irregular heartbeats. Using machine learning models as discussed in previous research [5], this feature would make the device useful even for people who don't have medical training.

6.2 Multi-Lead ECG Option

Right now, the machine uses a single lead, which works for basic monitoring. In the future, it can be upgraded to include more leads, which would make it better for detailed heart check-ups and more complex diagnoses [3].

6.3 Improving Signal Quality

The ECG machine can be improved by using advanced filtering techniques to reduce noise, such as motion artifacts or interference from other electrical devices. Techniques like precision filters and real-time noise reduction [8] would help provide clearer and more accurate heart signals for analysis.

6.4 Data Logging

Adding a data logging feature would allow users to save their ECG recordings for future reference. This could help doctors track a patient's heart health over time and identify any long-term changes. Data storage capabilities using IoT modules like ESP8266 have been explored in similar devices [4].

6.5 Remote Monitoring

This machine could be connected to telemedicine apps so doctors can monitor patients from far away in real time. This feature aligns with global trends in telemedicine [3], enabling quicker responses to emergencies and better care for rural populations.

6.6 Customizable Features

The device could include options for users to adjust the settings based on their needs, such as monitoring for children or older adults. A user-friendly interface, as implemented in this project, could be further refined to suit a wider range of users [**?**].

6.7 Support for Rural Healthcare Programs

Working with organizations and governments, this device could be made available to more people in remote areas. This would help ensure that everyone, no matter where they live, can check their heart health easily. Affordable healthcare technologies, like this ECG machine, have a proven impact on improving health equity [1].

These improvements would make the single-lead ECG machine even more useful and impactful, helping more people take care of their heart health in a simple and affordable way.

Chapter 7

CONCLUSION

The development of a portable, single-lead ECG machine demonstrates the potential for affordable and accessible cardiac monitoring solutions, particularly for underserved rural and remote areas. By utilizing custom-built spiral copper electrodes, the AD8232 bioinstrumentation amplifier, and the ESP8266 Wi-Fi module, the project successfully achieves real-time ECG signal acquisition, processing, and wireless data transmission.

The integration of a user-friendly web interface for ECG visualization ensures that the system is practical and convenient for non-specialist users. Field testing has shown that the device performs reliably, with comparable accuracy to standard ECG machines, while significantly reducing costs. This prototype bridges a critical gap in rural healthcare by providing an essential diagnostic tool for early detection and management of cardiac conditions.

Future enhancements could include the incorporation of multi-lead functionality, AI-driven data analysis, and extended battery life. This project highlights the significant impact that low-cost healthcare technologies can have on improving global health equity.

Appendix

Microcontroller program

```
#include <string>

#include <DNSServer.h>
#include <WebSocketsServer.h>
#include <ESP8266WebServer.h>

#define USE_SERIAL Serial

const byte DNS_PORT = 53;
IPAddress apIP(172, 217, 28, 1);
DNSServer dnsServer;
ESP8266WebServer server(80);
WebSocketsServer webSocket = WebSocketsServer(81);


// ADC buffer and indexing variables
int *adcBuffer = nullptr;    // Dynamic buffer pointer
int sampleIndex = 0;
int sps = 10;                // Default samples per second
unsigned long previousMillis = 0;
const long sendInterval = 333;   // Time interval to send data (333ms)

unsigned long lastSampleTime = 0;  // Time of last sample
long sampleInterval = 1000 / sps;  // Interval between each sample based on
    SPS


String webServerIndexPage = "<WEB UI CODE>";

void webSocketEvent(uint8_t num, WStype_t type, uint8_t* payload, size_t
   length) {

    switch (type) {
        case WStype_DISCONNECTED:
        USE_SERIAL.printf("[%u] Disconnected!\n", num);
```

```
34          break;
35          case WStype_CONNECTED:
36          {
37              IPAddress ip = webSocket.remoteIP(num);
38              USE_SERIAL.printf("[%u] Connected from %d.%d.%d.%d url: %s\n",
                    num, ip[0], ip[1], ip[2], ip[3], payload);
39          }
40          break;
41          case WStype_TEXT:
42          USE_SERIAL.printf("[%u] get Text: %s\n", num, payload);
43
44          //int arrayLen = payload - 0;
45          int newSps = atoi((const char*)payload);
46
47          // Try to parse the samples per second (SPS) from the message
48          if (newSps > 0 && newSps != sps) {
49              // Adjust the number of samples per second if needed
50              sps = newSps;
51
52              // Reallocate the ADC buffer with the new size
53              if (adcBuffer != nullptr) {
54                  free(adcBuffer); // Free old buffer
55              }
56              adcBuffer = (int *)malloc(sps * sizeof(int)); // Allocate new
                    buffer
57
58              // Update sampling interval based on SPS
59              sampleInterval = 1000 / sps;
60
61              Serial.print("New SPS: ");
62              Serial.println(sps);
63          }
64
65
66          break;
67      }
68  }
69
70  void handleNotFound() {
71      server.sendHeader("Location", "/", true); //Redirect to our html web
            page
72      server.send(301, "text/plane", "");
73  }
74
75
76  void setup() {
```

```
77      USE_SERIAL.begin(115200);
78
79      //USE_SERIAL.setDebugOutput(true);
80
81      USE_SERIAL.println();
82      USE_SERIAL.println();
83      USE_SERIAL.println();
84
85      for (uint8_t t = 4; t > 0; t--) {
86          USE_SERIAL.printf("[SETUP] BOOT WAIT %d...\n", t);
87          USE_SERIAL.flush();
88          delay(1000);
89      }
90      WiFi.mode(WIFI_AP);
91      WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
92      WiFi.softAP("RANMPGroupF");
93
94      // start webSocket server
95      webSocket.begin();
96      webSocket.onEvent(webSocketEvent);
97
98      dnsServer.start(DNS_PORT, "*", apIP);
99
100     // handle index
101     server.on("/", []() {
102         server.send(200, "text/html", webServerIndexPage);
103     });
104     server.onNotFound(handleNotFound);
105
106     server.begin();
107 }
108
109 unsigned long last_10sec = 0;
110 unsigned int counter = 0;
111
112 void loop() {
113     unsigned long t = millis();
114     webSocket.loop();
115     dnsServer.processNextRequest();
116     server.handleClient();
117
118     if ((t - last_10sec) > 10 * 1000) {
119         counter++;
120         bool ping = (counter % 2);
121         int i = webSocket.connectedClients(ping);
```

```
122         USE_SERIAL.printf("%d Connected websocket clients ping: %d\n", i,
                ping);
123         last_10sec = millis();
124     }
125
126
127     // Check if it's time to sample ADC data based on SPS
128     unsigned long currentMillis = millis();
129
130     if (currentMillis - lastSampleTime >= sampleInterval) {
131         // Read ADC and store it in buffer
132         adcBuffer[sampleIndex] = analogRead(A0);
133         sampleIndex++;
134
135         // Reset buffer and send data every 333ms
136         if (sampleIndex >= sps) {
137             sendSamplesThroughWebSocket(adcBuffer, sps);
138             sampleIndex = 0; // Reset buffer index
139         }
140
141         lastSampleTime = currentMillis;
142     }
143
144     // Send data every 333ms
145     if (currentMillis - previousMillis >= sendInterval) {
146         if (sampleIndex > 0) {
147             sendSamplesThroughWebSocket(adcBuffer, sampleIndex);
148             sampleIndex = 0; // Reset buffer index
149         }
150         previousMillis = currentMillis;
151     }
152 }
153
154 // Function to send ADC samples via WebSocket
155 void sendSamplesThroughWebSocket(int *samples, int count) {
156   // Create a buffer to hold raw data (binary format)
157   uint8_t buffer[count * sizeof(int)];
158   memcpy(buffer, samples, count * sizeof(int));  // Copy ADC samples to
          buffer
159
160   // Send the buffer as a binary message over WebSocket
161   webSocket.broadcastBIN(buffer, count * sizeof(int));
162 }
```

Web Development for UI

HTML

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>ECG Machine Mejor Project</title>
</head>
<body>
    <h1> Micro DSO</h1>
    <canvas height="200" id="graph"></canvas>
    <div class="position-control">
        <button onclick="autoScale()">Auto Scale</button><br />
        TIME/DIV
        <div>-<input type="range" oninput="time\_per\_div = (this.value
            /100)*6" min="1" max="99" value="99" />+</div>
        POSITION HORIZONTAL
        <div ><input disabled type="range" oninput="time\_per\_div = 31 -
            this.value" min="1" max="30" value="30" /></div>
        VOLT/DIV
        <div>-<input type="range" oninput="volt\_per\_div = 0.003 * this.
            value" min="15" max="45" value="1" />+</div>
        POSITION VERTICAL
        <div>\&\#9650;<input type="range" oninput="positionY =  -this.value
            " min="-200" max="200" value="30" />\&\#9660;</div>
    </div>
</body>
</html>
%
```

CSS

```
\#canvas {
    box-sizing: border-box;
}
body {
    margin: 0;
}
.position-control {
    text-align: center;
}
.position-control div {
    display: flex;
```

```
    justify-content: center;
    margin: 10px;
}
.position-control input[type="range"] {
    width: 80%;
}

.notify-container {
    width: 100%;
    height: auto;
    max-height: 230px;
    overflow: scroll;
    box-sizing: border-box;
    display: flex;
    flex-direction: column;
    align-items: center;
    position: fixed;
    top: 10px;
    left: 0;
    pointer-events: none;
}
.notify-container>div {
    max-width: 400px;
    margin: 10px;
    border-radius: 5px;
    text-align: center;
    padding: 10px 20px;
    background-color: \#eef;
    color: \#00f;
    position: relative;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    transition: transform .5s ease;
    pointer-events: auto;
}
.notify-container>div:hover {
    transform: translateY(10%);
}
.notify-container>div>.notifyClose {
    position: absolute;
    right: 5px;
    bottom: 25%;
    font-weight: 1000;
}
```

JavaScript

```
//Self designed lightweight notification
(()=>{
    var container = document.createElement('div');
    container.className = "notify-container";
    document.body.appendChild(container);
    document.notify = (r,a,j,u) => {
        var notifyDiv = document.createElement('div'),
        notifyClose = document.createElement('div');
        notifyDiv.innerText = r;
        notifyClose.className = "notifyClose";
        notifyClose.innerText = "×";
        notifyDiv.appendChild(notifyClose);
        if(a == "error")  notifyDiv.style = 'background-color:#fdd;color:
            f00;';
        if(a == "warning")  notifyDiv.style = 'background-color:#ffc;color:
            f00;';
        if(a == "success")  notifyDiv.style = 'background-color:#cfc;color
            :#080;';
        var stayTime = (j||(container.innerText.length + t.toString().
            length)*2e2);
        setTimeout(()=>{notifyDiv.remove()},stayTime)
        notifyClose.onclick = function(){notifyDiv.remove()};
        container.appendChild(notifyDiv);
    }
})();

const wSocket = new WebSocket('ws://172.217.28.1:81');
//const wSocket = new WebSocket('ws://localhost:8765');
wSocket.onopen = function() {
    document.notify("WebSocket: Connection established!",'success');
    wSocket.send(1)
};
wSocket.onmessage = function(event) {
    var incomingData = eval(event.data);
    var arr = String.fromCharCode.apply(null, new Uint8Array(incomingData))
        ;
    console.log(arr);
    data_plot.push(...incomingData);
    data_plot.splice(0, incomingData.length);
    wSocket.send(200);
};
wSocket.onerror = function(error) {
    document.notify("WebSocket: Error " + error,"error");
};
```

```
wSocket.onclose = function() {
    document.notify("WebSocket: Connection closed",'warning');
};

const canvas = _("#graph");
const ctx = canvas.getContext("2d");
var time_per_div = 6;
var _AnimationFrame;
var volt_per_div = 0.03
var axis_lebel_spaceX = 20;
var axis_lebel_spaceY = 20;
var positionX = 0;
var positionY = 0;
var pointerX = axis_lebel_spaceX;
var pointerY = axis_lebel_spaceY;
canvas.width = screen.width;

//Auto scale (in-progress)
function autoScale(){
    var scaleFactor = (canvas.height/2)/(Math.max(...data_plot) - Math.min
        (...data_plot));
    var scalingPosition  = (Math.min(...data_plot)+((Math.max(...data_plot)
         - Math.min(...data_plot))/2))*scaleFactor;
    //console.log(scalingPosition)
    positionY = scalingPosition-100 // sub 100 hardcoded for testing
        purpose ... must be a flexible solution
    volt_per_div = scaleFactor
    return scaleFactor;
}
//autoScale()

var sqr_sum = 0;
for(var i in data_plot){
    sqr_sum += Math.pow(data_plot[i],2)
}
var data_Vrms = Math.sqrt(sqr_sum/data_plot.length);
var data_Vavg = data_plot.reduce((acc, curr) => acc + curr, 0) / data_plot.
    length;
var data_Vmax = Math.max(...data_plot);
var data_Vmin = Math.min(...data_plot);
var data_Vpp = data_Vmax - data_Vmin;
console.log("Vrms :::",data_Vrms)
console.log("Vavg :::",data_Vavg);
console.log("Vmax :::",data_Vmax)
console.log("Vmin :::",data_Vmin)
console.log("Vp-p :::",data_Vpp)
```

```
82
83 addEventListener('resize', () => {
84     canvas.width = screen.width;
85     data_plot.length = canvas.width*time_per_div;
86 })
87 function draw() {
88     ctx.clearRect(0, 0, canvas.width, canvas.height);
89     drawGraphAxis();
90
91     ctx.beginPath();
92     ctx.moveTo(axis_lebel_spaceX, axis_lebel_spaceY);
93
94     for (var I in data_plot) {
95         i = Number(I);
96         //if (i % 5) continue;  // Skip every other index for performance
97         var plot_x = Math.round(i / time_per_div) + axis_lebel_spaceX;
98         var plot_y = canvas.height - Math.round(data_plot[i] * volt_per_div
               ) + axis_lebel_spaceY + positionY;
99
100        ctx.lineTo(plot_x, plot_y);
101
102        // Only for Testing perpose
103        /*if (graphPicks.indexOf(i) !== -1) {  // Mark every pick
104            ctx.fillText(i, plot_x, plot_y);
105        }*/
106
107
108        if (!(i % (100 * time_per_div))) {  // Mark every 100th data point
109            ctx.fillText(i, plot_x, axis_lebel_spaceY / 2);
110        }
111
112        if (plot_x > canvas.width) break;  // Stop if exceeding canvas
               width
113    }
114
115    ctx.lineWidth = 1;
116    ctx.strokeStyle = 'red';
117    ctx.stroke();
118
119    _AnimationFrame = requestAnimationFrame(draw);
120 }
121
122 draw();
123
124 // Function to create grid on canvas
125 function drawGraphAxis() {
```

```
126     ctx.lineWidth = 0.1;
127 
128     // Vertical grid lines
129     for (var i = 0; i < canvas.width; i += canvas.width / time_per_div) {
130         ctx.beginPath();
131         ctx.moveTo(i + axis_lebel_spaceX, axis_lebel_spaceY / 2);
132         ctx.lineTo(i + axis_lebel_spaceX, canvas.height + axis_lebel_spaceY
                );
133         ctx.strokeStyle = '#000';
134         ctx.stroke();
135     }
136 
137     // Horizontal grid lines
138     for (var i = 0; i < canvas.height - axis_lebel_spaceY; i += 30) {
139         ctx.beginPath();
140         ctx.moveTo(axis_lebel_spaceX, i + axis_lebel_spaceY);
141         ctx.lineTo(canvas.width + axis_lebel_spaceX, i + axis_lebel_spaceY)
                ;
142         ctx.fillText(i, 0, i + axis_lebel_spaceY);
143         ctx.strokeStyle = '#000';
144         ctx.stroke();
145     }
146 
147     // Draw axis pointers
148     ctx.lineWidth = 1;
149 
150     // Vertical pointer line
151     ctx.beginPath();
152     ctx.moveTo(pointerX, 0);
153     ctx.lineTo(pointerX, canvas.height);
154     ctx.strokeStyle = '#00f';
155     ctx.stroke();
156 
157     // Horizontal pointer line
158     ctx.beginPath();
159     ctx.moveTo(0, pointerY);
160     ctx.lineTo(canvas.width, pointerY);
161     ctx.strokeStyle = '#00f';
162     ctx.stroke();
163 
164     // Display pointer coordinates
165     ctx.fillText(
166         Math.round((pointerX - axis_lebel_spaceX) * time_per_div) + "," + (
                canvas.height - pointerY),
167         canvas.width / 2, axis_lebel_spaceX + 10
168     );
```

```
}

/*
/
/   TO BE Modified for mouse compatiblity
/
*/
// x & y axis on touch
//canvas.addEventListener("mousemove", pointerEvent)
canvas.addEventListener("touchmove", pointerEvent)
function pointerEvent() {
    event.preventDefault();
    const touch = event.type == "touchmove"?event.touches[0]:event;
    const x = Math.round(touch.clientX - canvas.offsetLeft);
    const y = Math.round(touch.clientY - canvas.offsetTop);

    x > 0 && x < canvas.width-axis_lebel_spaceX && (pointerX = (event.type
       == "touchmove"?canvas.width-x:x));
    y > 0 + axis_lebel_spaceY && y < canvas.height && (pointerY = y);
};

function _(o){
    return document.querySelector(o);
}

/*TO DO's
-include time, volt mul factor -- in ESP API
-fix y axis scale
- optimize code (make it flexible for wide range of application)
- auto scale (max - min && find volt_per_div)
- pack it in a Class

*/

// For Testing
// positionY = 200
// time_per_div = 5.9
```

References

[1] Parker, J. (2019). *Introduction to Bioinstrumentation: Fundamentals and Applications*. Pearson Education.

[2] Analog Devices. (2014). *AD8232 Data Sheet*. Retrieved from `https://www.analog.com/media/en/technical-documentation/data-sheets/AD8232.pdf`

[3] Arias, R. A., & Llamas, J. S. (2016). Portable Electrocardiogram (ECG) for Remote Health Monitoring. *Journal of Medical Systems*, 40(12), 271-285. `https://doi.org/10.1007/s10916-016-0620-9`

[4] Espressif Systems. (2015). *ESP8266EX Datasheet*. Retrieved from `https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf`

[5] Kiani, M. (2018). ECG Signal Processing, Classification, and Interpretation: A Comprehensive Review. *Computers in Biology and Medicine*, 59, 125-141. `https://doi.org/10.1016/j.compbiomed.2015.11.002`

[6] Wang, J., & Xu, X. (2017). Design and Implementation of Wireless ECG Monitoring System Based on Wi-Fi. *Journal of Medical Engineering & Technology*, 41(4), 239-244. `https://doi.org/10.1080/03091902.2017.1362821`

[7] Mozilla Developer Network. (2024). *WebSocket API*. Retrieved from `https://developer.mozilla.org/en-US/docs/Web/API/WebSocket`

[8] Tayal, A., & Soni, R. (2018). Signal Filtering and Noise Reduction in ECG Systems: A Comparative Study. *Bioengineering*, 5(4), 88-98. `https://doi.org/10.3390/bioengineering5040088`

[9] Arduino. (2023). *Arduino IDE: Integrated Development Environment*. Retrieved from `https://www.arduino.cc/en/software`