Real-Time ECG Monitoring System with ESP32 and AD8232 Sensor

A Research Project Submitted in Partial Fulfillment of the Requirement for the Degree of Bachelor of Science (Engineering) in Computer Science and Engineering.

Submitted By

Rokibul Hassan Remon

ID: CE-19015

Session:2018-19

&

Md. Fattahul Hasan Fahim

ID: CE-19043

Session: 2018-19

Supervised By

Mohammed Hadifur Rahman

Associate Professor

Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY SANTOSH,

TANGAIL-1902, BANGLADESH.

Approval

The Research Project Report about "**ECG with Esp32 and AD8232**" Submitted by

Rokibul Hassan Remon (ID: CE19015) & Md. Fattahul Hasan Fahim (ID: CE19043) to the Department of Computer Science and Engineering, MBSTU, Santosh, Tangail-1902, Bangladesh, has been accepted as satisfactory for the partial fulfilment of the requirements for the degree of Bachelor of Science (Engineering) in Computer Science and Engineering and approved as to its style and contents.

Board of Examiner:

1.(Supervisor) ……………………………………………

2.(Examiner) ……………………………………………

3.(Examiner) …………………………………………….

Declaration

We, here by, declare that the Project development work which is presented by the outcome of the investigation performed by us under the supervision of Mohammed Hadifur Rahman, Associate Professor, Department of Computer Science and Engineering, (MBSTU), Santosh, Tangail-1902, Bangladesh.

We also declare that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

....

Mohammed Hadifur Rahman

Supervisor

Signature

……………………..

Rokibul Hassan Remon

………………………

Md. Fattahul Hasan Fahim

Candidate

Abstract

This study introduces an Analog-to-Analog Electrocardiogram (ECG) Monitoring System utilizing ESP32 microcontroller and AD8232 ECG sensor. The system enables continuous cardiac health monitoring by wirelessly transmitting analog ECG signals. The ESP32 microcontroller manages signal processing and communication, while the AD8232 sensor captures ECG signals accurately. Signal conditioning ensures signal integrity, and Wi-Fi connectivity facilitates remote data transmission. The system's performance was validated for real-time signal acquisition and transmission, demonstrating reliability and stability. With applications in remote patient monitoring and healthcare diagnostics, this system contributes to the development of wearable health monitoring devices, addressing the need for accessible and affordable cardiac health solutions.

Acknowledgement

First, we would like to express our deepest in debt ness and gratitude to the most powerful, gracious, Almighty Allah for giving us knowledge, energy, and patience for completing the project work successfully. We are very much grateful to our honorable project supervisor Mohammed Hadifur Rahman, Associate Professor, Department of Computer Science and Engineering, (MBSTU), for his continuous guidance, valuable suggestions, constructive comments, and endless encouragement throughout the preparation of this project.

We are also grateful to the rest of the Computer Science and Engineering teachers of Mawlana Bhashani Science and Technology University who have helped us by giving valuable suggestions at different times.

Finally, we are offering our regards and blessings to all of those who have supported us in any respect during the completion of the research project.

Preface

The project is outlined based on the working procedure and details of the system. This is carried out in the department of Computer Science & Engineering, Faculty of Engineering, at Mawlana Bhashani Science and Technology University in Santosh, Tangail -1902, Bangladesh.

This project includes 6 chapters which are briefed as follows:

Chapter-1:

Chapter 1 provides the introduction of the system we designed. It describes what the system intends to do.

Chapter-2:

Chapter 2 provides technical tools and language.

Chapter-3:

Chapter 3 discusses the Device architecture of the entire system.

Chapter-4:

Chapter 4 discusses the implementation.

Chapter-5:

Chapter 5 System Pros and Cons

Chapter-6:

Chapter 6 Future thoughts and Conclusions.

Table of Contents

List of Figure

Chapter 1

Introduction

1.1 Project Definition

The goal of this project is to develop a portable and user-friendly electrocardiogram (ECG) monitoring system. This system accurately captures and displays the heart's electrical activity in real-time, providing users with valuable insights into their heart rate and rhythm. By giving users an easy way to check their heart health and promoting proactive management of cardiac well-being.

Also, the project wants to help users feel more confident and in control. The system is easy to use and helps users keep an eye on their health. It encourages users to make smart choices about their well-being. This way of thinking not only helps users stay healthy but also joins a bigger effort to prevent health problems before they happen.

1.2 Project Purpose

The purpose of this project is to develop a user-friendly and accessible electrocardiogram (ECG) monitoring system that prioritizes ease of use and convenience. By ensuring that the system is free of advertisements and offers free access to ECG signals, it aims to make vital health information easily accessible to everyone. Additionally, the project focuses on scalability and accommodating an unlimited number of users, with the system being available for all individuals. With a commitment to low power consumption and easy integration with Wi-Fi routers, the goal is to create a robust and versatile platform that empowers individuals to monitor their heart health effortlessly.

1.3 Requirements

The requirement is a condition or capability that is required to be present in a product, service, or result to satisfy a business need. For creating our application some requirements are necessary. We have divided these requirements into two categories. These requirements are:

- Software Requirements
- Hardware Requirement

1.3.1 Software Requirement

Software requirements are the software tools we need to build a project. The software requirements of our project are given below:

- IDE: **VS** Code, Arduino.
- Web design: HTML, CSS.
- Frontend: JavaScript.

1.3.2 Hardware Requirement

Hardware requirements are the hardware needed to build a project. The hardware requirements of our project are given below:

- **Microcontroller Board:** Esp-32.
- **Display:** 16X2 LCD with I2C module
- **PCB:** (14.5 cm x 6.5 cm Dot Zero Board)
- **Sensor**: AD8232 (ECG sensor)
- **Connector**: Female Header Connector Single Row,

- **Soldering**: Lodestar Soldering Iron - 60W
- **Lead Mini Roll**: Solder Lead Mini Rang
- **Wire**: Jumper wire
- **Sensor pad**: 3.5 mm Jack for Biomedical Pad Connection.
- **Sensor Cable** - Electrode Pads (3 connector)

1.4 Overview

From the beginning to end the project, can describe into some steps:

- Connect to the Wi-Fi router.
- Publish the ECG signal what AD8232 sensor sensed on the website.
- Anybody can see user ECG signal.

Chapter 2

Technical Background and Tools

2.1 Related Work

It allows users to get instant ECG anywhere without going to hospital or expertise. It is harmless, portable, and easy to plug in and out, that's why it saves time and effort. Without any delay in one minute the system can show the ECG signal. However, it is crucial to work with such type of hardware component and connect each of item through Wi-Fi routers.

2.2 ESP32

The ESP32, developed by Espressif Systems, is a versatile microcontroller with a dual-core Tensilica Xtensa LX6 processor running at up to 240 MHz It offers a rich set of peripherals including GPIO, SPI, I2C, UART, ADC, DAC, and PWM for interfacing with sensors and devices. Its integrated Wi-Fi module supports 2.4 GHz and 5 GHz networks, enabling seamless internet connectivity and cloud communication. Bluetooth capabilities, including Bluetooth Low Energy (BLE), extend connectivity to smartphones and wearables. With ample GPIO pins and support for Arduino IDE and ESP-IDF, the ESP32 is widely used in IoT applications for its powerful processing, wireless connectivity, and extensive peripheral support.

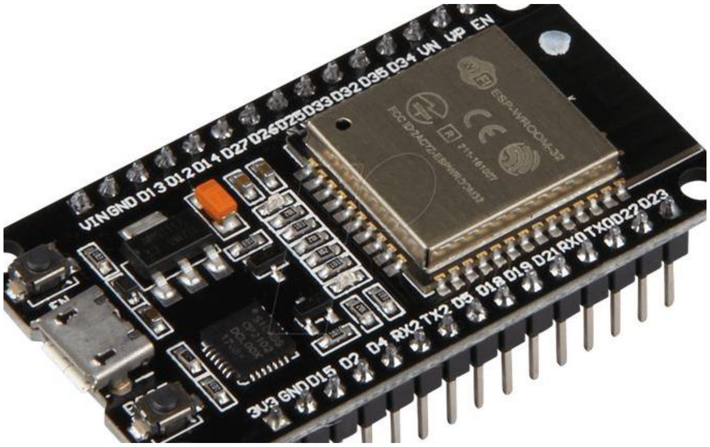

VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
ESP-WROOM-32
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23

Figure 2.2.1: ESP32 WROOM

2.3 16x2 LCD display

The 16x2 LCD display features 16 characters per line across 2 lines. It's compatible with Arduino and other microcontrollers, offering a simple interface for displaying text, numbers, and symbols. Ideal for various projects requiring basic alphanumeric display functionality.

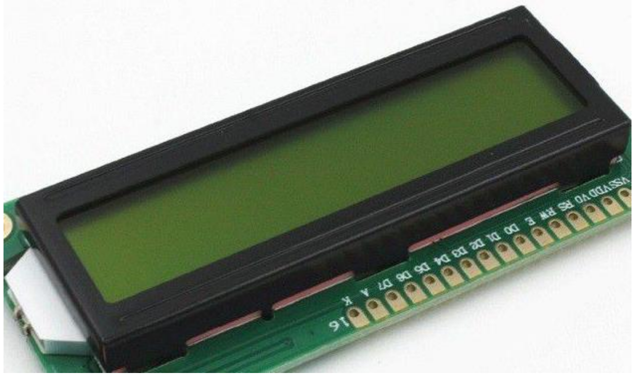


Figure 2.3.1: 16x2 LCD display

2.4 Dot Zero / Vero Board

The "Dot Zero" board is a compact PCB (Printed Circuit Board) designed for building electronic prototypes or projects. It typically features a grid of pre-drilled holes arranged in a pattern that allows for the easy mounting of electronic components and soldering connections. The Dot Zero board provides a convenient platform for creating custom circuits, experimenting with different designs, and assembling small-scale electronic projects without the need for custom PCB fabrication.

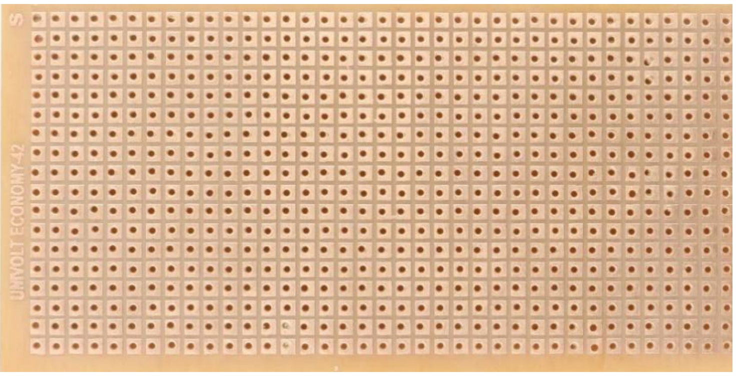


Figure 2.4.1: Dot Zero Board

2.5 AD8232 (ECG sensor)

The AD8232 is an integrated sensor for measuring bioelectric signals, particularly electrocardiogram (ECG) signals. Developed by Analog Devices, it integrates instrumentation amplifiers, filters, and lead-off detection circuitry. Compact and low power, it's ideal for portable ECG monitors, fitness trackers, and health monitoring systems. Its high-quality signal acquisition makes it popular among hobbyists, researchers, and medical professionals for a variety of wearable and medical applications.

Technical Data:

- Power voltage: DC 3.3V
- Output: analog output
- Interface (connect RA, LA, RL): 3PIN, 2.54PIN or earphone jack.
- Working Temperature: -40 °C to +85 °C

Pin Out:

3.3v – Input Power Supply

GND – Supply ground

LO+ - D16

LO- - D17

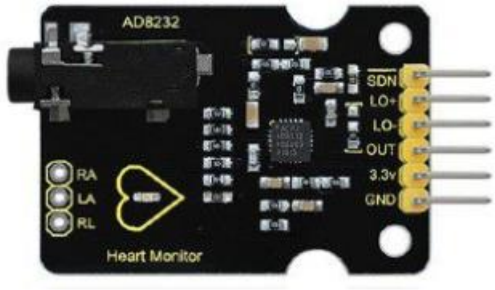


Figure 2.5.1: AD8232(ECG sensor)

2.6 Female Header Connectors

Single-row female header connectors make it easy to connect things like sensors and displays to circuit boards. They have slots in a single line where i can easily plug in and unplug male pins. These connectors are great for trying out ideas and making temporary connections when building electronic projects.

Figure 2.6.1: Female Header Connectors

2.7 Soldering Iron(60W)

A solder joint formed with a 60W soldering iron melts the solder onto the connection point, creating a secure bond between components. The heat from the iron melts the solder, allowing it to flow and adhere to the joint.

Figure 2.7.1: 16x2 LCD display

2.8 Lead Mini Roll - Rang

The higher the tin content, the higher the melting point. Lead reduces the melting point making lead-based solders easier to work with as they flow better and are quicker to bring to a working temperature. These are the reason of using lead for soldering: Quality of Connection, Component Protection, Efficiency, Ease of Use

Figure 2.8.1: Lead Mini Roll

2.9 Jumper wires

Jumper wires are flexible conductive wires used to make temporary electrical connections between components on a breadboard or PCB. They allow for easy prototyping and experimentation in electronic projects.

Figure 2.9.1: Jumper wires

2.10 USB Type-B

A USB Type-B cable is commonly used to connect devices like printers, scanners, and external hard drives to computers. It features a square-shaped Type-B connector on one end, designed to fit into the corresponding port on the device.

Figure 2.10.1: USB Type-B

2.11 Arduino Code Editor

The Arduino IDE (Integrated Development Environment) is a software platform used for programming Arduino microcontroller boards. It provides a user-friendly interface for writing, compiling, and uploading code to Arduino devices, making it accessible for hobbyists, students, and professionals to develop a wide range of projects.

Figure 2.11.1: Arduino IDE

2.12 Electrode Pad

An electrode pad is a component used in medical devices for attaching electrodes to the body. These pads typically have adhesive backing to securely attach to the skin and are used in procedures like ECG (Electrocardiogram), EMG (Electromyography), TENS (Transcutaneous Electrical Nerve Stimulation), and more.

Figure 2.12.1: Electrode pad

2.13 Sensor Cable

A sensor cable is a flexible wire with connectors on each end used to connect sensors to electronic devices. It facilitates the transmission of data or signals between the sensor and the device, enabling monitoring, control, or data acquisition in various applications.

Figure 2.13.1: Sensor Cable

2.14 I2C module

The use of a hardware I2C module significantly reduces the number of pins required to interface a 16x2 LCD display with a microcontroller. Instead of needing multiple pins for data and control signals, the I2C module only requires two pins: SDA (Serial Data) and SCL (Serial Clock). This simplifies the wiring process and conserves valuable GPIO pins on the microcontroller, making it easier to integrate the LCD display into projects with limited pin availability. Additionally, the I2C protocol allows for serial communication between multiple devices using just two wires, enabling efficient communication in various embedded systems applications.

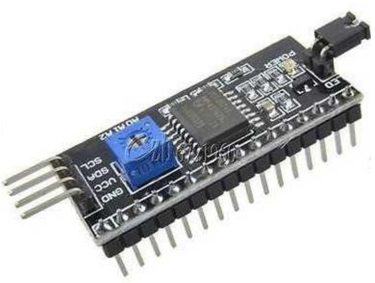


Figure 2.14.1: I2C module

Chapter 3

The System Architecture

3.1 System Architecture

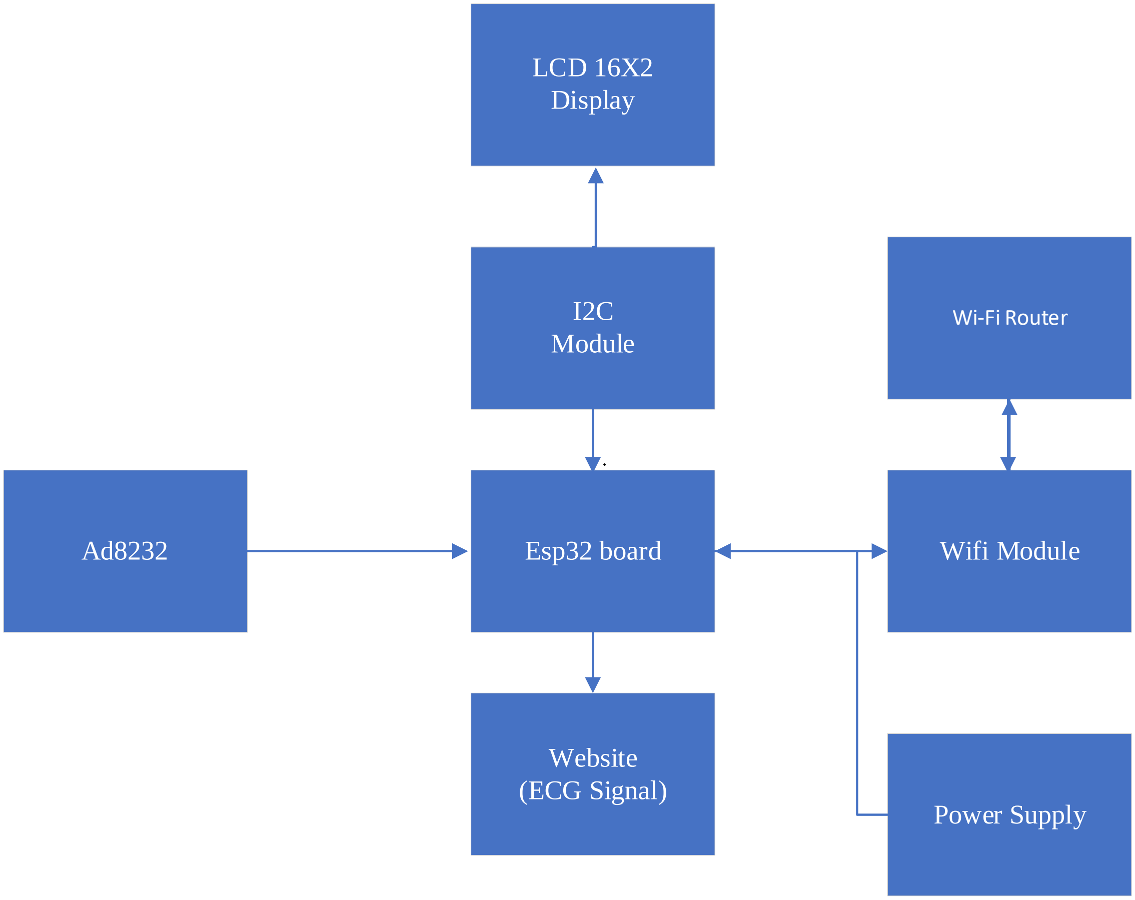


Figure 3.1.1: The System Architecture

3.2 Block diagram

A System block diagram portrays a series of actions that define how module work. A Block diagram represents how the task of system flow between resource find the machine or user uses the application software development, a block diagram is a delineation of series of steps a process must execute consistently.

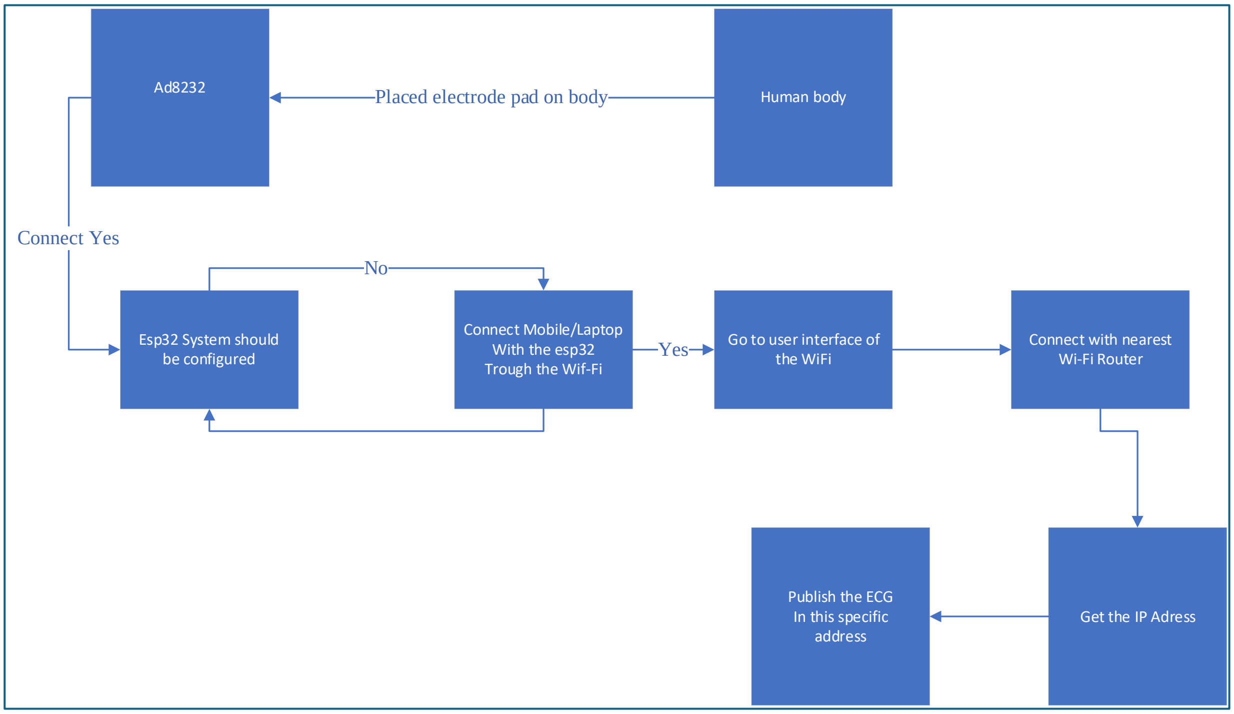

Ad8232
Placed electrode pad on body
Human body
Connect Yes
No
Esp32 System should be configured
Connect Mobile/Laptop With the esp32 Trough the Wif-Fi
Yes
Go to user interface of the WiFi
Connect with nearest Wi-Fi Router
Publish the ECG In this specific address
Get the IP Adress

Figure 3.2.1: Block Diagram

3.3 Use Case Diagram:

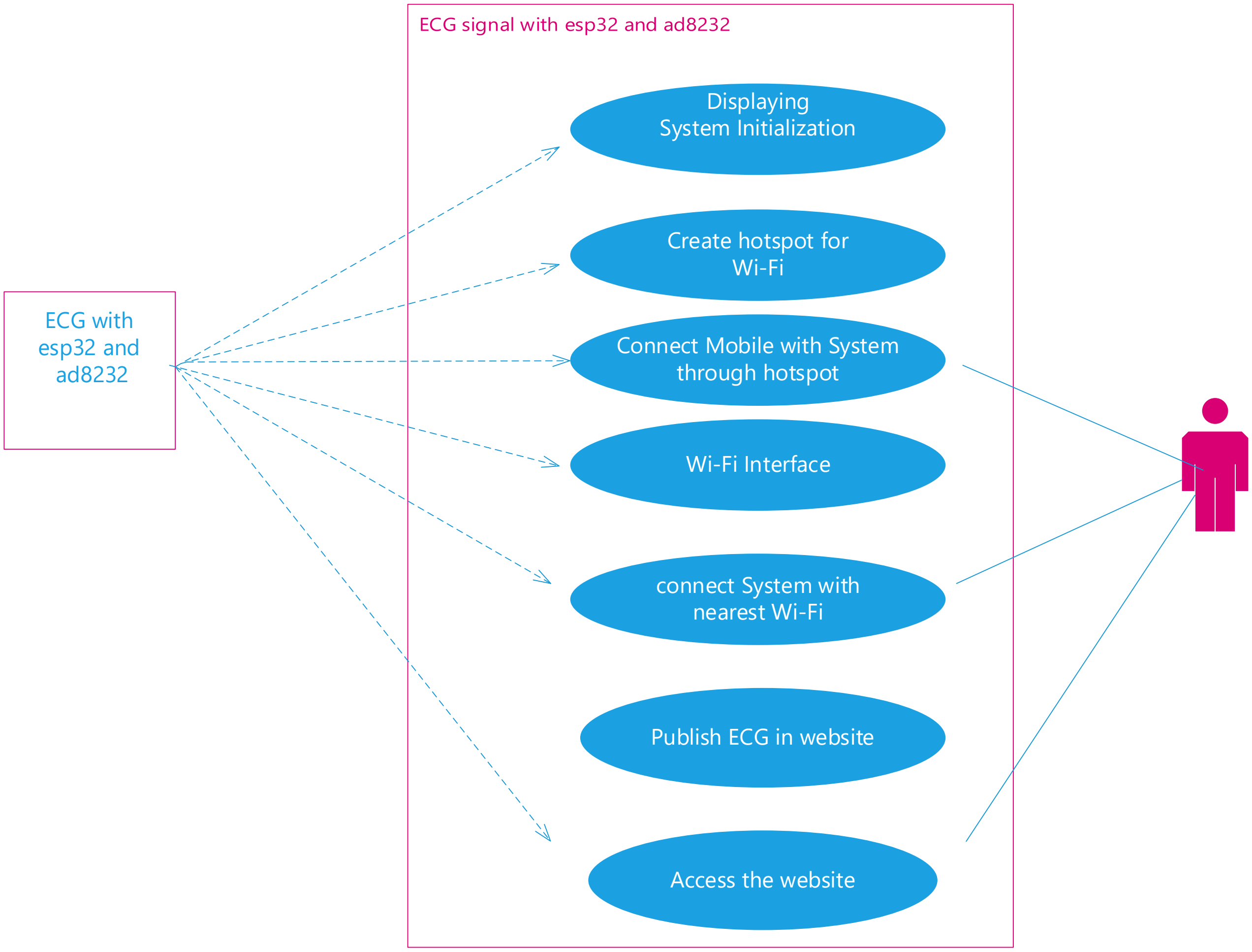

ECG signal with esp32 and ad8232
ECG with esp32 and ad8232
Displaying System Initialization
Create hotspot for Wi-Fi
Connect Mobile with System through hotspot
Wi-Fi Interface
connect System with nearest Wi-Fi
Publish ECG in website
Access the website

Figure 3.3.1: Use case diagram.

3.4 The Circuit Diagram

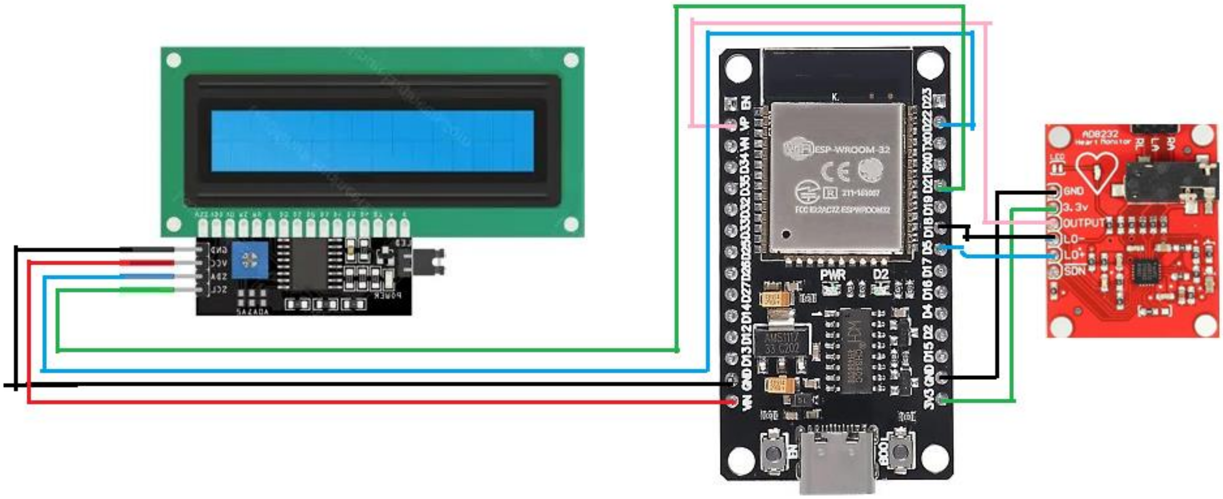


Figure 3.4 .1: Circuit Diagram

Chapter 4

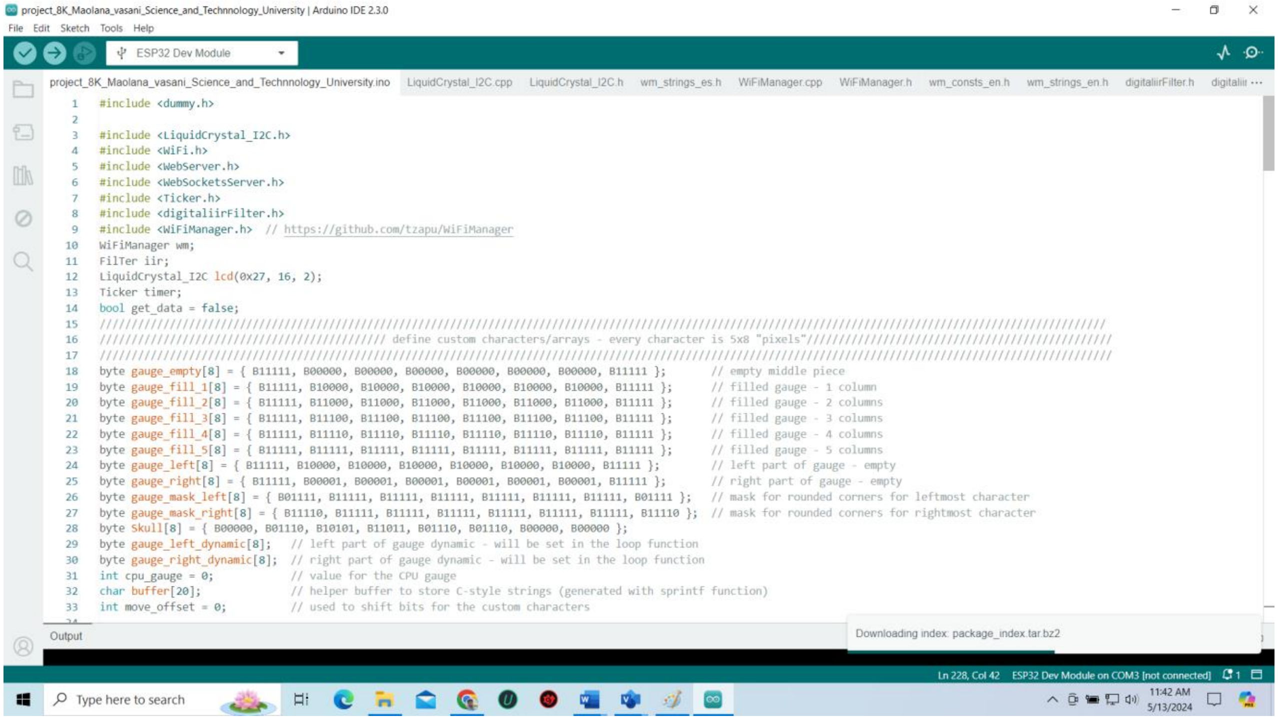
The System Implementation

The objective of this project is to develop an eCommerce website integrated with a voice application, enabling users to interact with the platform using voice commands. The website will provide a seamless shopping experience, allowing customers to browse products, make purchases, track orders, and perform other common eCommerce functions, all through voice interactions.

4.1 The Arduino Code

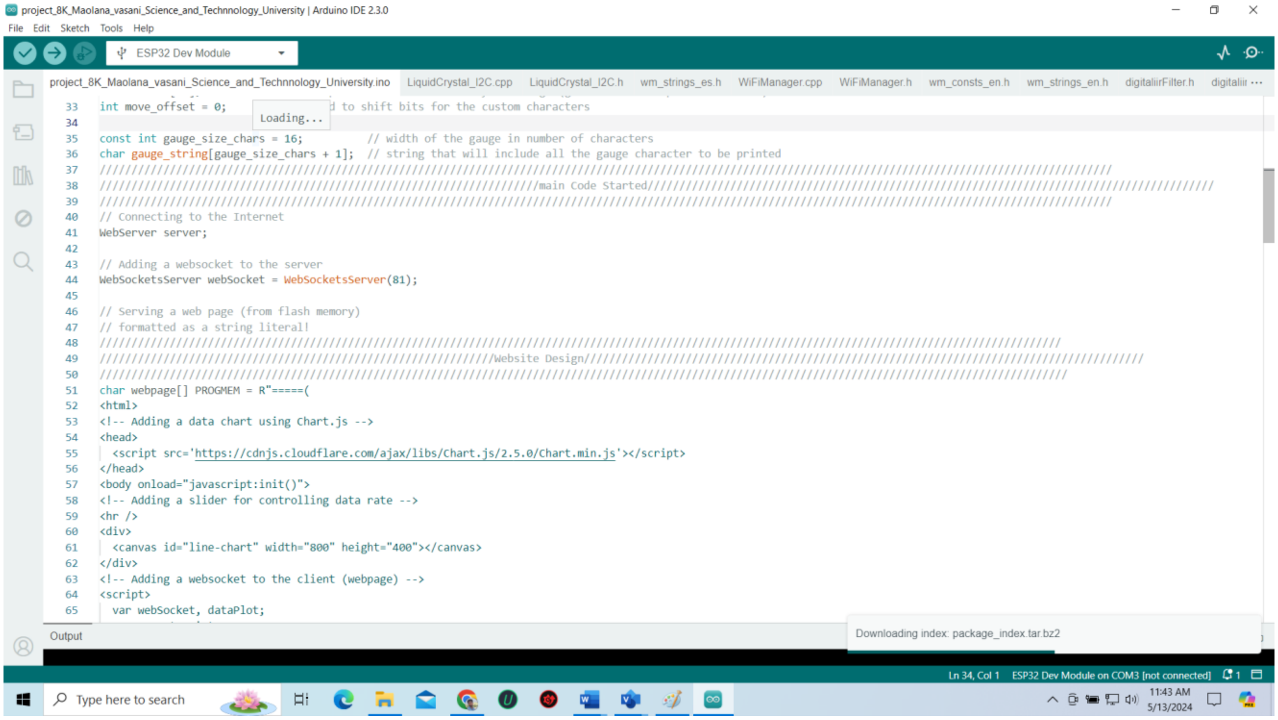
The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Here the code of the project is implemented in Arduino platform. The code is shown in the following figures:


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitaliir ...
Output
Downloading index: package_index.tar.bz2
Ln 228, Col 42 ESP32 Dev Module on COM3 [not connected]
Type here to search
11:42 AM
5/13/2024

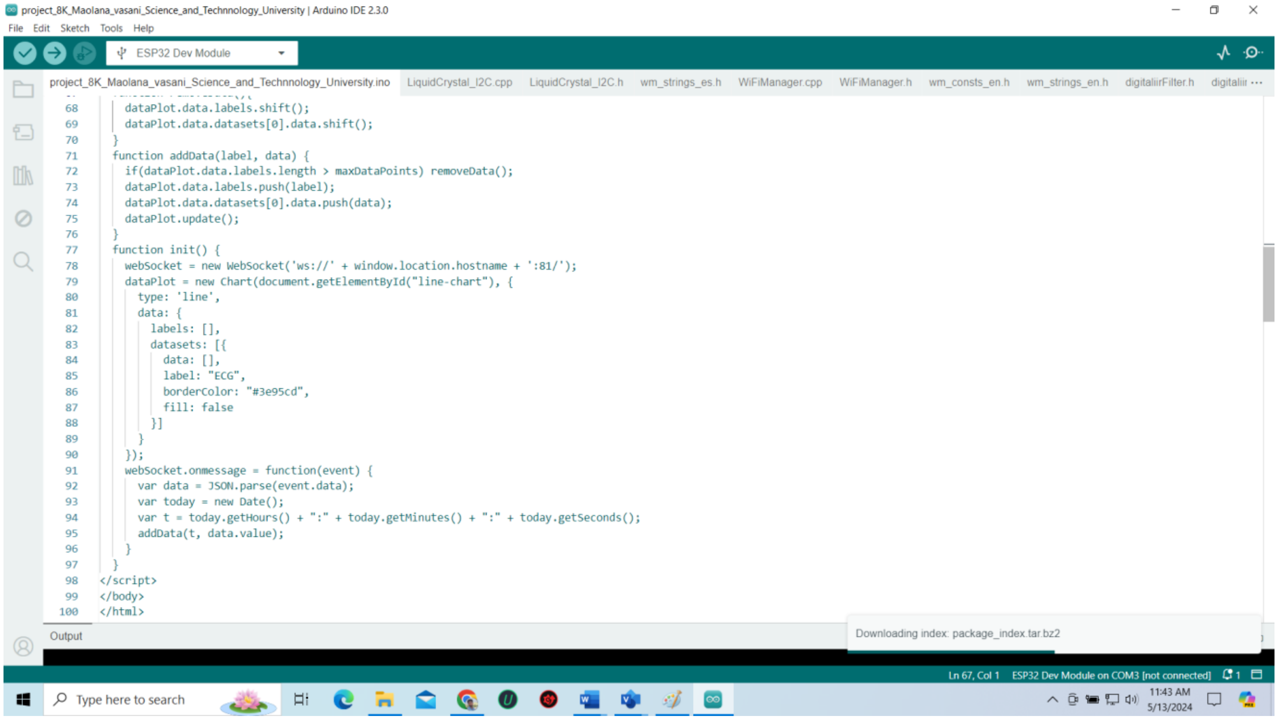
```
#include <dummy.h>

#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <WebServer.h>
#include <WebSocketsServer.h>
#include <Ticker.h>
#include <digitaliirFilter.h>
#include <WiFiManager.h>  // https://github.com/tzapu/WiFiManager
WiFiManager wm;
FilTer iir;
LiquidCrystal_I2C lcd(0x27, 16, 2);
Ticker timer;
bool get_data = false;
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////// define custom characters/arrays - every character is 5x8 "pixels"//////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
byte gauge_empty[8] = { B11111, B00000, B00000, B00000, B00000, B00000, B00000, B11111 };       // empty middle piece
byte gauge_fill_1[8] = { B11111, B10000, B10000, B10000, B10000, B10000, B10000, B11111 };      // filled gauge - 1 column
byte gauge_fill_2[8] = { B11111, B11000, B11000, B11000, B11000, B11000, B11000, B11111 };      // filled gauge - 2 columns
byte gauge_fill_3[8] = { B11111, B11100, B11100, B11100, B11100, B11100, B11100, B11111 };      // filled gauge - 3 columns
byte gauge_fill_4[8] = { B11111, B11110, B11110, B11110, B11110, B11110, B11110, B11111 };      // filled gauge - 4 columns
byte gauge_fill_5[8] = { B11111, B11111, B11111, B11111, B11111, B11111, B11111, B11111 };      // filled gauge - 5 columns
byte gauge_left[8] = { B11111, B10000, B10000, B10000, B10000, B10000, B10000, B11111 };        // left part of gauge - empty
byte gauge_right[8] = { B11111, B00001, B00001, B00001, B00001, B00001, B00001, B11111 };       // right part of gauge - empty
byte gauge_mask_left[8] = { B01111, B11111, B11111, B11111, B11111, B11111, B11111, B01111 };   // mask for rounded corners for leftmost character
byte gauge_mask_right[8] = { B11110, B11111, B11111, B11111, B11111, B11111, B11111, B11110 };  // mask for rounded corners for rightmost character
byte Skull[8] = { B00000, B01110, B10101, B11011, B01110, B01110, B00000, B00000 };
byte gauge_left_dynamic[8];   // left part of gauge dynamic - will be set in the loop function
byte gauge_right_dynamic[8];  // right part of gauge dynamic - will be set in the loop function
int cpu_gauge = 0;            // value for the CPU gauge
char buffer[20];              // helper buffer to store C-style strings (generated with sprintf function)
int move_offset = 0;          // used to shift bits for the custom characters
```

Figure 4.4.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitaliir ...
Loading...
Output
Downloading index: package_index.tar.bz2
Ln 34, Col 1 ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

```
int move_offset = 0;                d to shift bits for the custom characters

const int gauge_size_chars = 16;          // width of the gauge in number of characters
char gauge_string[gauge_size_chars + 1];  // string that will include all the gauge character to be printed
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////main Code Started////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Connecting to the Internet
WebServer server;

// Adding a websocket to the server
WebSocketsServer webSocket = WebSocketsServer(81);

// Serving a web page (from flash memory)
// formatted as a string literal!
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////Website Design/////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
char webpage[] PROGMEM = R"=====(
<html>
<!-- Adding a data chart using Chart.js -->
<head>
  <script src='https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.5.0/Chart.min.js'></script>
</head>
<body onload="javascript:init()">
<!-- Adding a slider for controlling data rate -->
<hr />
<div>
  <canvas id="line-chart" width="800" height="400"></canvas>
</div>
<!-- Adding a websocket to the client (webpage) -->
<script>
  var webSocket, dataPlot;
```

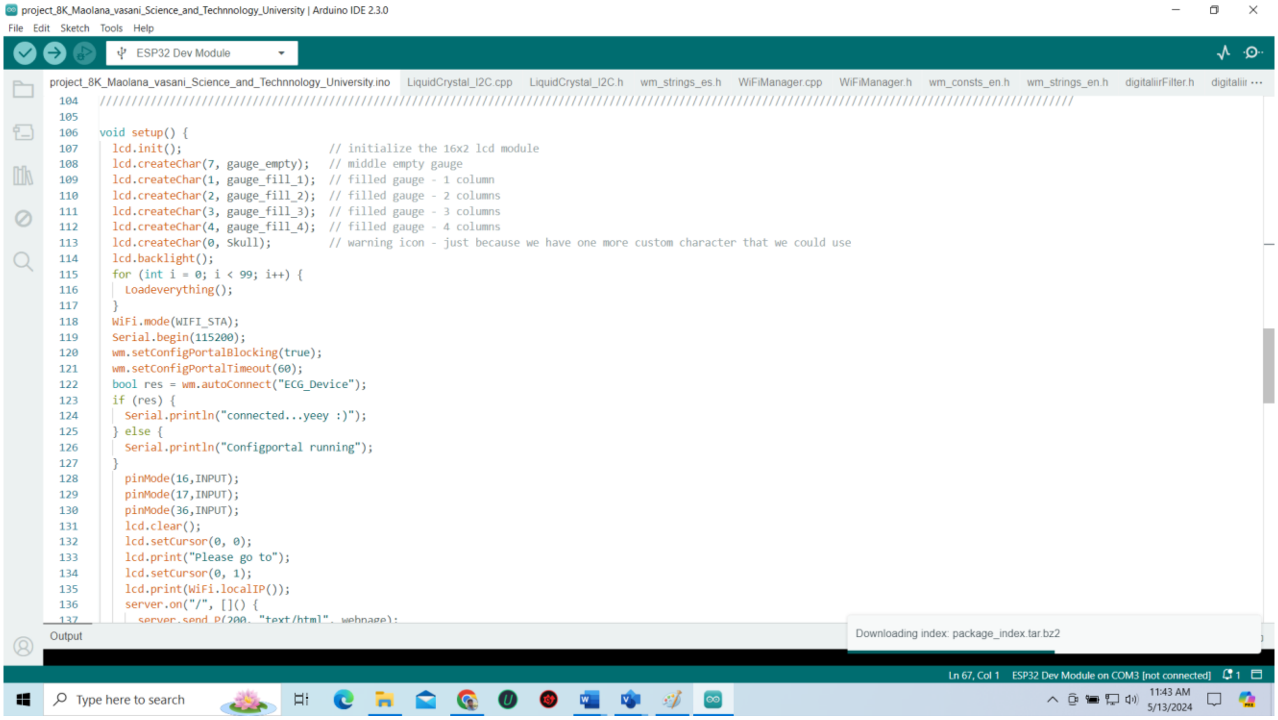
Figure 3.3.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitalii
Output
Ln 67, Col 1 ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

```
    dataPlot.data.labels.shift();
    dataPlot.data.datasets[0].data.shift();
  }
  function addData(label, data) {
    if(dataPlot.data.labels.length > maxDataPoints) removeData();
    dataPlot.data.labels.push(label);
    dataPlot.data.datasets[0].data.push(data);
    dataPlot.update();
  }
  function init() {
    webSocket = new WebSocket('ws://' + window.location.hostname + ':81/');
    dataPlot = new Chart(document.getElementById("line-chart"), {
      type: 'line',
      data: {
        labels: [],
        datasets: [{
          data: [],
          label: "ECG",
          borderColor: "#3e95cd",
          fill: false
        }]
      }
    });
    webSocket.onmessage = function(event) {
      var data = JSON.parse(event.data);
      var today = new Date();
      var t = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
      addData(t, data.value);
    }
  }
</script>
</body>
</html>
```

Downloading index: package_index.tar.bz2

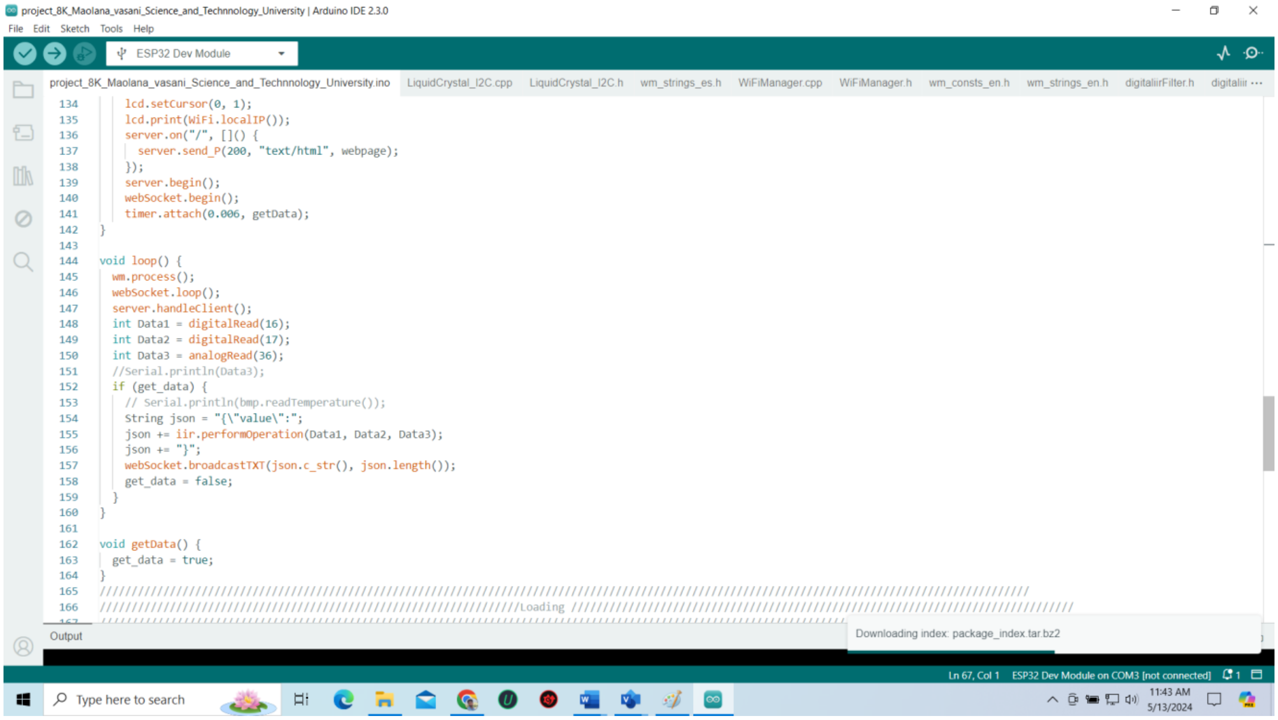
Figure 3.3.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitalii
Output
Ln 67, Col 1 ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

```
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void setup() {
  lcd.init();                    // initialize the 16x2 lcd module
  lcd.createChar(7, gauge_empty);   // middle empty gauge
  lcd.createChar(1, gauge_fill_1);  // filled gauge - 1 column
  lcd.createChar(2, gauge_fill_2);  // filled gauge - 2 columns
  lcd.createChar(3, gauge_fill_3);  // filled gauge - 3 columns
  lcd.createChar(4, gauge_fill_4);  // filled gauge - 4 columns
  lcd.createChar(0, Skull);         // warning icon - just because we have one more custom character that we could use
  lcd.backlight();
  for (int i = 0; i < 99; i++) {
    Loadeverything();
  }
  WiFi.mode(WIFI_STA);
  Serial.begin(115200);
  wm.setConfigPortalBlocking(true);
  wm.setConfigPortalTimeout(60);
  bool res = wm.autoConnect("ECG_Device");
  if (res) {
    Serial.println("connected...yeey :)");
  } else {
    Serial.println("Configportal running");
  }
    pinMode(16,INPUT);
    pinMode(17,INPUT);
    pinMode(36,INPUT);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Please go to");
    lcd.setCursor(0, 1);
    lcd.print(WiFi.localIP());
    server.on("/", []() {
      server.send_P(200, "text/html", webpage);
```

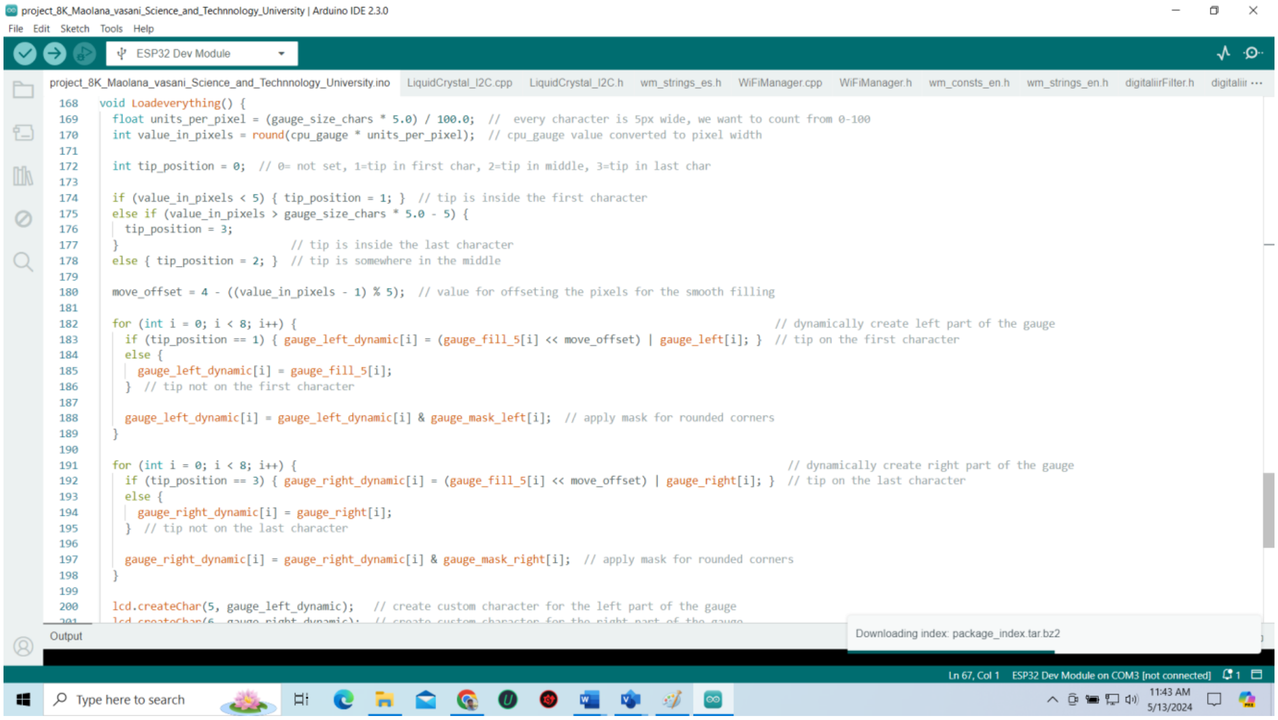
Downloading index: package_index.tar.bz2

Figure 3.3.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitaliir
Output
Downloading index: package_index.tar.bz2
Ln 67, Col 1
ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

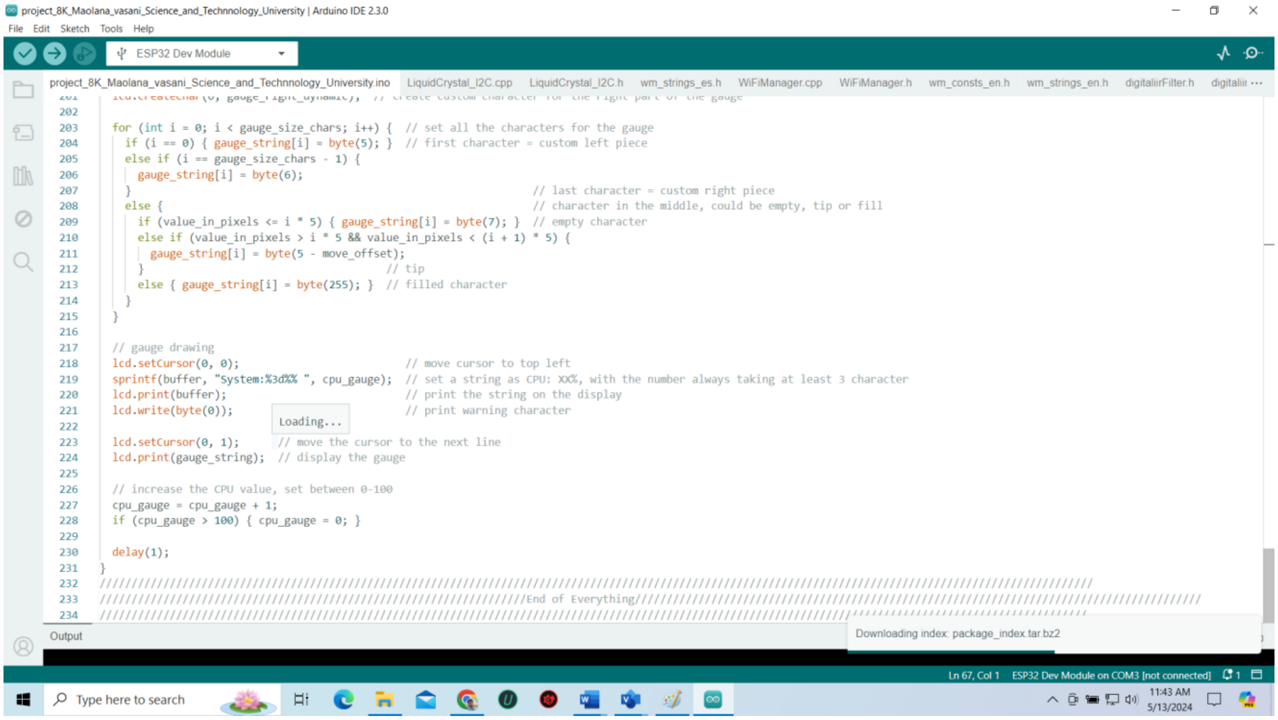
```
    lcd.setCursor(0, 1);
    lcd.print(WiFi.localIP());
    server.on("/", []() {
      server.send_P(200, "text/html", webpage);
    });
    server.begin();
    webSocket.begin();
    timer.attach(0.006, getData);
}

void loop() {
  wm.process();
  webSocket.loop();
  server.handleClient();
  int Data1 = digitalRead(16);
  int Data2 = digitalRead(17);
  int Data3 = analogRead(36);
  //Serial.println(Data3);
  if (get_data) {
    // Serial.println(bmp.readTemperature());
    String json = "{\"value\":";
    json += iir.performOperation(Data1, Data2, Data3);
    json += "}";
    webSocket.broadcastTXT(json.c_str(), json.length());
    get_data = false;
  }
}

void getData() {
  get_data = true;
}
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////Loading ///////////////////////////////////////////////////////////////
```

Figure 3.3.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitaliirFilter.h
digitaliir
Output
Downloading index: package_index.tar.bz2
Ln 67, Col 1
ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

```
void Loadeverything() {
  float units_per_pixel = (gauge_size_chars * 5.0) / 100.0;  //  every character is 5px wide, we want to count from 0-100
  int value_in_pixels = round(cpu_gauge * units_per_pixel);  // cpu_gauge value converted to pixel width

  int tip_position = 0;  // 0= not set, 1=tip in first char, 2=tip in middle, 3=tip in last char

  if (value_in_pixels < 5) { tip_position = 1; }  // tip is inside the first character
  else if (value_in_pixels > gauge_size_chars * 5.0 - 5) {
    tip_position = 3;
  }                            // tip is inside the last character
  else { tip_position = 2; }  // tip is somewhere in the middle

  move_offset = 4 - ((value_in_pixels - 1) % 5);  // value for offseting the pixels for the smooth filling

  for (int i = 0; i < 8; i++) {                                                                  // dynamically create left part of the gauge
    if (tip_position == 1) { gauge_left_dynamic[i] = (gauge_fill_5[i] << move_offset) | gauge_left[i]; }  // tip on the first character
    else {
      gauge_left_dynamic[i] = gauge_fill_5[i];
    }  // tip not on the first character

    gauge_left_dynamic[i] = gauge_left_dynamic[i] & gauge_mask_left[i];  // apply mask for rounded corners
  }

  for (int i = 0; i < 8; i++) {                                                                    // dynamically create right part of the gauge
    if (tip_position == 3) { gauge_right_dynamic[i] = (gauge_fill_5[i] << move_offset) | gauge_right[i]; }  // tip on the last character
    else {
      gauge_right_dynamic[i] = gauge_right[i];
    }  // tip not on the last character

    gauge_right_dynamic[i] = gauge_right_dynamic[i] & gauge_mask_right[i];  // apply mask for rounded corners
  }

  lcd.createChar(5, gauge_left_dynamic);   // create custom character for the left part of the gauge
  lcd.createChar(6, gauge_right_dynamic);  // create custom character for the right part of the gauge
```

Figure 3.3.1: Code.


project_8K_Maolana_vasani_Science_and_Technnology_University | Arduino IDE 2.3.0
File Edit Sketch Tools Help
ESP32 Dev Module
project_8K_Maolana_vasani_Science_and_Technnology_University.ino
LiquidCrystal_I2C.cpp
LiquidCrystal_I2C.h
wm_strings_es.h
WiFiManager.cpp
WiFiManager.h
wm_consts_en.h
wm_strings_en.h
digitalIirFilter.h
digitalIir
Loading...
Output
Downloading index: package_index.tar.bz2
Ln 67, Col 1
ESP32 Dev Module on COM3 [not connected]
Type here to search
11:43 AM
5/13/2024

```
  lcd.createChar(0, gauge_right_dynamic);  // create custom character for the right part of the gauge

  for (int i = 0; i < gauge_size_chars; i++) {  // set all the characters for the gauge
    if (i == 0) { gauge_string[i] = byte(5); }  // first character = custom left piece
    else if (i == gauge_size_chars - 1) {
      gauge_string[i] = byte(6);
    }                                                         // last character = custom right piece
    else {                                                    // character in the middle, could be empty, tip or fill
      if (value_in_pixels <= i * 5) { gauge_string[i] = byte(7); }  // empty character
      else if (value_in_pixels > i * 5 && value_in_pixels < (i + 1) * 5) {
        gauge_string[i] = byte(5 - move_offset);
      }                                        // tip
      else { gauge_string[i] = byte(255); }  // filled character
    }
  }

  // gauge drawing
  lcd.setCursor(0, 0);                          // move cursor to top left
  sprintf(buffer, "System:%3d%% ", cpu_gauge);  // set a string as CPU: XX%, with the number always taking at least 3 character
  lcd.print(buffer);                            // print the string on the display
  lcd.write(byte(0));                           // print warning character

  lcd.setCursor(0, 1);        // move the cursor to the next line
  lcd.print(gauge_string);  // display the gauge

  // increase the CPU value, set between 0-100
  cpu_gauge = cpu_gauge + 1;
  if (cpu_gauge > 100) { cpu_gauge = 0; }

  delay(1);
}
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////End of Everything////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Figure 3.3.1: Code

4.2 User Interface

4.2.1 Beginning of the System

Here we showed the loading percentage into the LCD display. LCD provides excellent resolution, brightness, and contrast so the picture quality is crystal clear. LCDs can be suitable with CMOS integrated circuits so making an LCD is very easy. It gives perfect sharpness at the native resolution.

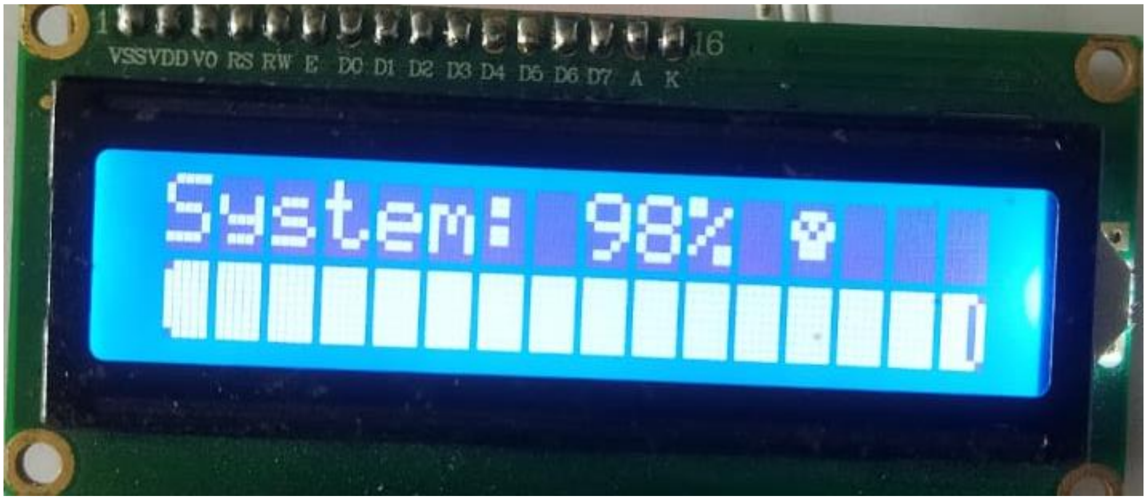
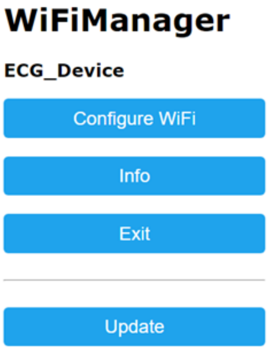


Figure 3.3.1: Beginning of the system.

4.2.2 While mobile connected to the system

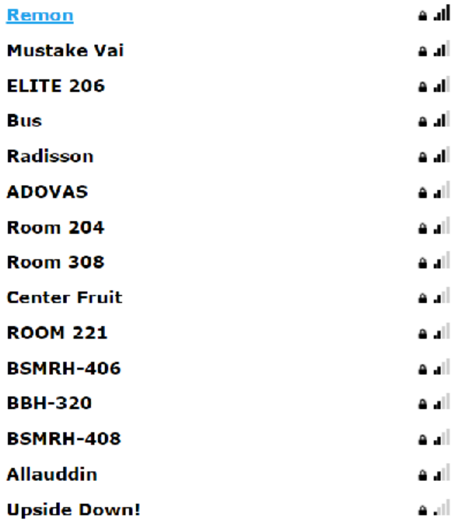


WiFiManager

ECG_Device

Figure 3.3.1: Wi-Fi Manager.

4.2.3 Showing Wi-Fi interface.



Remon

Mustake Vai

ELITE 206

Bus

Radisson

ADOVAS

Room 204

Room 308

Center Fruit

ROOM 221

BSMRH-406

BBH-320

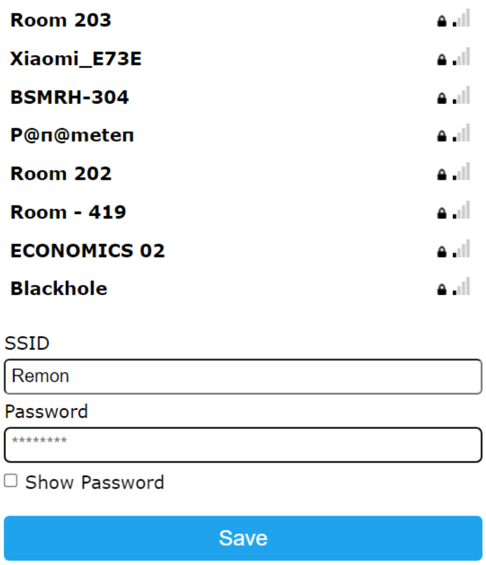
BSMRH-408

Allauddin

Upside Down!

Figure 3.3.1: Wi-Fi Interface.

4.2.4 Choose the nearest Wi-Fi Router for connecting system.



Room 203

Xiaomi_E73E

BSMRH-304

P@п@meteп

Room 202

Room - 419

ECONOMICS 02

Blackhole

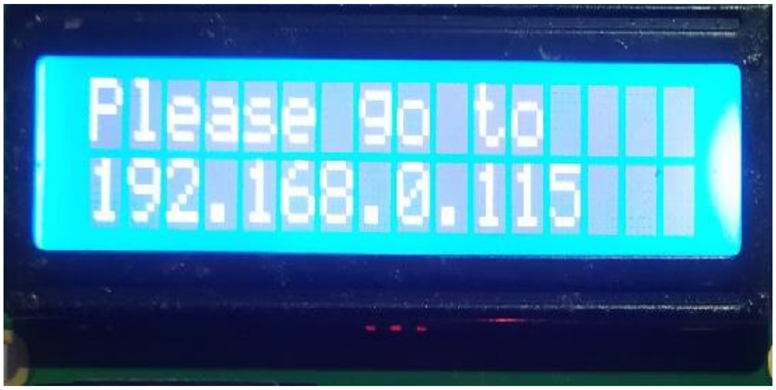
SSID: Remon

Password: ********

☐ Show Password

Figure 3.3.1: Nearest Wi-Fi.

4.2.5 Showing Website address.



Please go to
192.168.0.115

Figure 3.3.1: Website Address.

4.2.6 Publish the ECG data on the website.

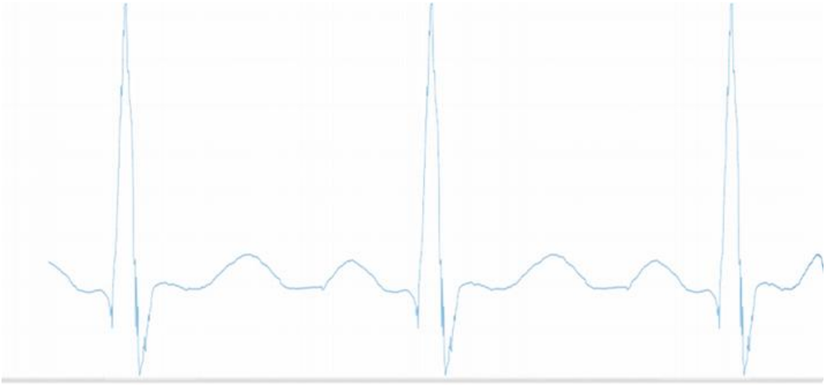


Figure 3.3.1: ECG signal.

Chapter 5

System Pros and Cons

5.1 The pros of the system

The system comes with a handful of benefit these are described below decision.

- **Responsive**

  The system we have developed responds to almost all devices while designing the website we consider the minimum target API level was selected. Nowadays almost 90% of Android and all the digital devices can use this website. So, it will be supported in most of the devices and will give the proper response via Wi-Fi module.

- **Robust**

  The system is robust for obvious reasons. The design of the app is kept simple so that the user interface does not affect the performance. Besides, the system is created to show the ECG signal of humans and there are no extra features. So that the entire system works smoothly, and the device is any type of advs. free, so the system gets efficiency issue here.

- **User Friendly**

The system is so simple that anybody can access it and use it for their own purpose.

- **Cost-Effective**

Utilizing an esp32 board and Ad8232 ECG sensor reduces the cost compared to traditional.

ECG, which can be expensive. Esp32 board is very cheap, and the requirement of the coding tool is open source and available on the internet.

- **No external power supply**

The esp32 board is powered solely via USB connection to the PC, eliminating the need for an external power supply. This simplifies the setup.

- **Real-Time Monitoring:**

The system allows for real-time monitoring of ECG signals, providing immediate feedback to users or healthcare professionals. This capability can be crucial in monitoring patient health or conducting real-time experiments in research settings.

- **Low Maintenance:**

Due to its simple design and reliable components, the system requires minimal maintenance. This lowers operational costs and ensures consistent performance over time, making it a practical long-term solution.

- **Educational Use:**

The simplicity and affordability of our system make it ideal for educational purposes. It can be used to teach students about ECG technology, signal processing, and programming, fostering learning and innovation in related fields.

5.2 The cons of the system

Also, some there are also a few cons of the system for which the system is repeat inconvenient my collector:

- **External Power Supply**

  The system requires external power supply which can be provided by a battery or mobile or power bank.

- **Wire complexity**

  Place the 3 electrode pads on the chest is not comfortable but the signal comes from chest through the sensor is less noiseless but if we place those pads on hand then not get better ECG signal.

- **Resolution and Accuracy**

  The resolution and accuracy of the measurements may be limited compared to higher -end ECG. Factors such as ADC resolution, noise and signal conditions can affect the overall performance of the system.

- **Signal Integrity**

  The quality of signal may be impacted by noise interference or signal dispersion particularly when using capturing the sensor data from human body.

- **Dependency on Internet Connectivity:**

  Our system relies on internet connectivity for data transmission or remote monitoring, it may face limitations in areas with poor or unreliable internet access.

- **Single User Limitation:**

  The system may be designed for single-user use at a time, which could be limiting in scenarios where simultaneous monitoring or data collection from multiple users is required.

- **Educational or Training Requirements:**

  Users may need initial education or training to properly set up and interpret the ECG signals, potentially increasing initial learning curve or deployment time.

To improve these issues over time, we can listen to users' feedback, make technological advancements, and focus on targeted improvements. This approach will help us reduce these problems and make our system better for everyone.

Chapter 6

Future Thoughts and Conclusions

In future, there are lots of things we can do to enhance our project even better:

Adding More Cool Stuff

We want to make our ECG system do even more, like analyzing data in real-time and giving personalized health tips.

Working with Hospitals

We could team up with hospitals to share our ECG data with doctors and make it part of people's medical records.

Making Wearable Tech

We might create small gadgets people can wear to keep an eye on their heart all the time, even when they are moving around.

Finding New Uses

We will keep exploring how our ECG tech can help, like spotting early signs of heart problems or tracking fitness levels.

Integration with Smart Devices:

Expand compatibility with smart devices and IoT (Internet of Things) platforms, enabling seamless integration with other health monitoring devices and smart home systems for comprehensive wellness management.

User Feedback and Customization: Implement features for users to provide feedback and customize their ECG monitoring experience, ensuring the system evolves according to user needs and preferences.

6.2 Conclusion:

"The introduction of our 'ECG with ESP32 and AD8232' project represents a significant step forward in heart health monitoring. By making it easier to use and accessible to everyone, we're helping people keep track of their heart health more effectively.

Looking ahead, we're focused on making our system even better. We plan to work closely with healthcare providers to integrate our technology into hospitals and clinics. We're also exploring wearable options, like devices you can wear all day, to give continuous updates on heart health. These efforts aim to improve how people take care of their hearts, using technology to empower both individuals and healthcare professionals."