

Heritage Institute of Technology

(An Autonomous Institute)

Department of Electronics and Communication Engineering



-----DESIGN AND DEVELOPMENT OF WIRELESS ECG DEVICE -----

Affiliated to

Maulana Abul Kalam Azad University of Technology

(Formerly WBUT), 2023

Name	College Roll no.	Autonomy Roll no.	Registration no.
Aniket Sarkar	2152022	12621003015	211260100310082
Ritwik Saha	2152209	12622003213	221260121062
Swastika Banerjee	2152110	12621003146	211260100310055
Mousumi Sinha	2152254	12622003204	221260121053

Under the Supervision of

(Name of the Project Guide) -----Prof. Pratima Shaw
(Designation of Guide) -----Assistant Professor

Heritage Institute of Technology
(An Autonomous Institute)
Department of
Electronics and Communication Engineering



This is to certify that the project report
-----**DESIGN AND DEVELOPMENT OF WIRELESS ECG DEVICE** -----
Has been successfully completed by

Name	College Roll no.	Autonomy Roll no.	Registration no.
Aniket Sarkar	2152022	12621003015	211260100310082
Ritwik Saha	2152209	12622003213	221260121062
Swastika Banerjee	2152110	12621003146	211260100310055
Mousumi Sinha	2152254	12622003204	221260121053

In partial fulfillment for the award of the degree in
Bachelor of Technology
In
Electronics and Communication Engineering
Maulana Abul Kalam Azad University of Technology
(Formerly WBUT), 2025

Under the Supervision of
(Name of the Project Guide) -----Prof. Pratima Shaw
(Designation of Guide) -----Assistant Professor

Dept. of Electronic and Communication Engineering
Heritage Institute of Technology, Kolkata-700107.



Certificate of Recommendation

This is to certify that the Thesis entitled ***“DESIGN AND DEVELOPMENT OF WIRELESS ECG DEVICE”*** submitted by **Aniket Sarkar, Ritwik Saha, Swastika Banerjee, Mousumi Sinha** under the supervision of ***Prof. Pratima Shaw*** (Assistant Professor, Dept. of ECE, HITK), has been prepared according to the regulations of **B.Tech. Degree in Electronics and Communication Engineering Department**, awarded by **Maulana Abul Kalam Azad University of Technology (Formerly WBUT)** and he/she has fulfilled the requirements for submission of thesis report and that neither his/her thesis report has been submitted for any degree/diploma or any other academic award anywhere before.

.....
-----Prof. Pratima Shaw-----
(Assistant Professor, Dept of ECE, HITK)
Project Supervisor

.....
Prof. Prabir Banerjee
(HOD, Dept of ECE, HITK)

Heritage Institute of Technology
(An Autonomous Institute)

Affiliated to

Maulana Abul Kalam Azad University of
Technology
(Formerly WBUT)



Certificate of Approval*

The foregoing thesis report is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned don't necessarily endorse or approve any statement made opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it is submitted.

Signature of the Examiners:

1.....

2.....

3.....

**Only in the case the thesis report is approved*

Heritage Institute of Technology
(An Autonomous Institute)

Affiliated to

Maulana Abul Kalam Azad University of
Technology
(Formerly WBUT)

DECLARATION FOR NON-COMMITMENT OF PLAGIARISM

We, Aniket Sarkar, Ritwik Saha, Swastika Banerjee, Mousumi Sinha, Student of B.Tech in the Department of Electronics & Communication Engineering, Heritage Institute of Technology Kolkata have submitted the project report in fulfilment of the requirements to obtain the above-noted degree.

We declare that we have not committed plagiarism in any form or violated copyright while writing the report and have acknowledged the sources and /or the credit of other authors wherever applicable.

Signature of the Students:

1

2

3

4

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who have supported and guided us throughout the successful completion of our final year project titled “Design & Development of Wireless ECG Device”.

First and foremost, we would like to extend our heartfelt thanks to Prof. Pratima Shaw, our project guide, for her constant support, valuable insights, and expert guidance throughout the course of this project. Her encouragement and constructive feedback have been instrumental in shaping our work.

We also express our gratitude to all the faculty members of the Electronics and Communication Engineering Department at Heritage Institute of Technology, Kolkata (HITK), for providing us with the foundational knowledge and support that enabled us to take on this project with confidence.

A special note of thanks to the IEEE Lab of HITK for granting us access to essential equipment and resources, which played a crucial role in the development and testing of our prototype.

We would also like to thank our peers, friends, and family members who motivated and supported us throughout this journey.

Finally, we acknowledge the efforts of all those who contributed directly or indirectly to the success of this project.

Team Members:

(Aniket Sarkar, Ritwik Saha, Swastika Banerjee, Mousumi Sinha)

Signature of the Students:

1.....

2.....

3.....

4.....

CONTENTS :

Sl.no	Topic	Page no.
C.1	ABSTRACT	2
C.1.1	INTRODUCTION	3
C.1.2	ECG WAVEFORM	4
C.1.2.1	ARDUINO UNO	5
C.1.2.2	ARDUINO UNO FEATURES	5-6
C.1.2.3	ARDUINO UNO PIN DESCRIPTION	6-7
C.2	CIRCUIT DIAGRAM/CONNECTION BETWEEN ARDUINO AND ECG SENSOR AD8232	8
C.2.1	ARDUINO CODE	9
C.2.2	ARDUINO IMPLEMENTATION RESULT	9
C.2.2.1	IMPLEMENTATION USING ESP32	10
C.2.2.2	AD8232 ECG SENSOR	10-11
C.2.2.3	ESP32	11-12
C.2.2.4	CIRCUIT CONNECTIONS & LEAD PLACEMENTS	12
C.2.3	SOURCE CODE & UBIDOTS PLATFORM	13-14
C.2.3.1	SOURCE CODE	14-15
C.2.3.2	ESP32 IMPLEMENTATION RESULT	16
C.3	FUTURE SCOPE OF THE PROJECT	17
C.4	REFERENCE	18

LIST OF FIGURES

Diagram no.	Description
1.) 4.)	Cross Section of Heart
3.) 5.) 6.)	ECG Waveform
7.) 8.)	Ad8232 ECG Sensor
9.) 10.)	Arduino Uno
11.) 12.)	Arduino and AD8232 Circuit, Output
13.) 14.)	Arduino Code and ECG Output
15.) 16.)	ESP 32 External & Internal View
17.) 20.) 21.)	Esp32 and AD8232 Connections. Diagramatic & Actual Images
18.) 19.)	Lead Placements on patient's body- Chest & Hand Types
22.)	Ubidots Cloud Platform
23.) 24.) 25.) 26.)	Source Code part 1, 2, 3, 4 describing libraies & void reconnect, callback, setup, loop
27.) 29.)	Outputs taken in Lab
28.)	ECG Data in excel sheet

Abstract

The ECG(electrocardiography) is a very crucial medical technique that helps in detecting and preventing cardiovascular disease at onset.

Our purpose in this project is to build a simple and cost effective wireless ECG system that records and publishes data on the cloud. This relatively cheap IoT implementation can allow doctors to remotely monitor their patients data from anywhere on the planet.

We capture ECG signals from the heart in real time using the AD8232 ECG sensor. Our preliminary implementation involved using Arduino Uno as a microcontroller, so that we could visualize the signals directly on an oscilloscope or a connecting laptop. This was to ensure reliability and precision during our initial testing phases.

The plan is to build an affordable IoT device for ECG monitoring. This was done by making a wireless version of the project, (initially done with Arduino Uno) using esp32. This is a vital upgrade as the device now has remote monitoring capabilities, with data being published on cloud, reducing the reliance on physical connections. Thus, the device is ideal for applications where mobility and convenience are critical, such as home healthcare and telemedicine

The integration of essential hardware components, including the breadboard and jumper wires, provided a robust yet flexible setup for prototyping and testing.

INTRODUCTION

An electrocardiogram (ECG) is a simple, non-invasive test that measures and evaluates the heart's electrical activity and rhythm:

How it works

Electrodes (often called lead, available in 3,6,12 configurations) are attached to the chest, arms, and legs, and connected to a device that records and displays the heart's electrical signals ECG waveform. This machine can be attached to a patient's body for an extended period of time to detect any abnormalities.

What it's used for

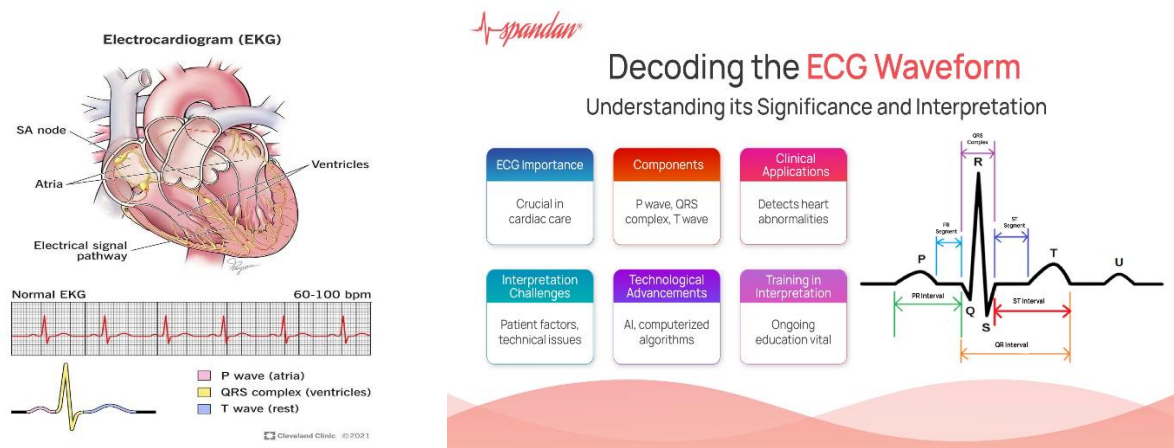
An ECG can help diagnose heart conditions like abnormal rhythms (arrhythmia, tachycardia), coronary heart disease, and heart attacks. It can also be used to monitor how well heart treatments are working, or to rule out heart disease before or after surgery.

When it's recommended

A doctor may recommend an ECG if you have symptoms like chest pain, dizziness, breathlessness, or a feeling of your heart racing. It can help visualize any irregularity of heart rate or rhythm and potential damages to heart muscle fibres or blood vessels.

Risks and results

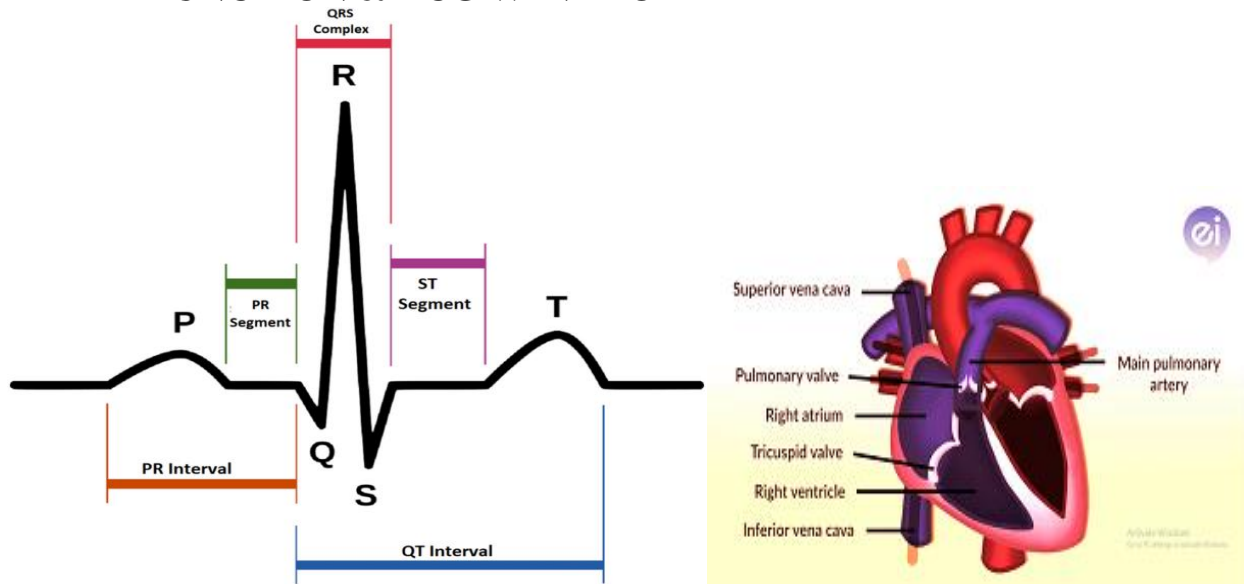
- An ECG procedure is generally considered safe and the patient doesn't feel anything during the process. The results are usually released on the same day but sometimes it could take a few weeks depending on the type of ECG.



1.) C.S of Heart

2.) ECG Waveform with Stages

HEART FUNCTION & ECG WAVEFORM

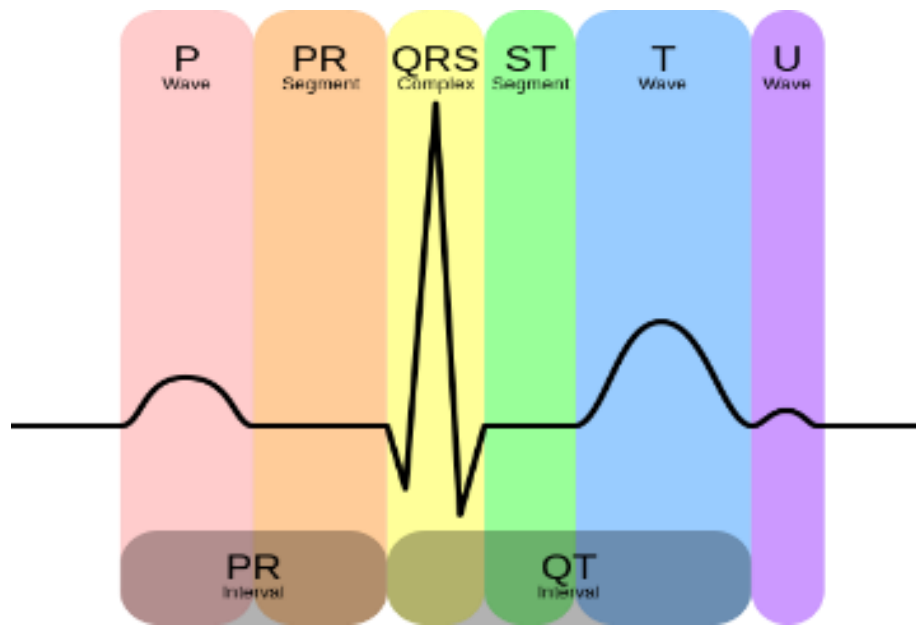


3.) ECG Waveform XL

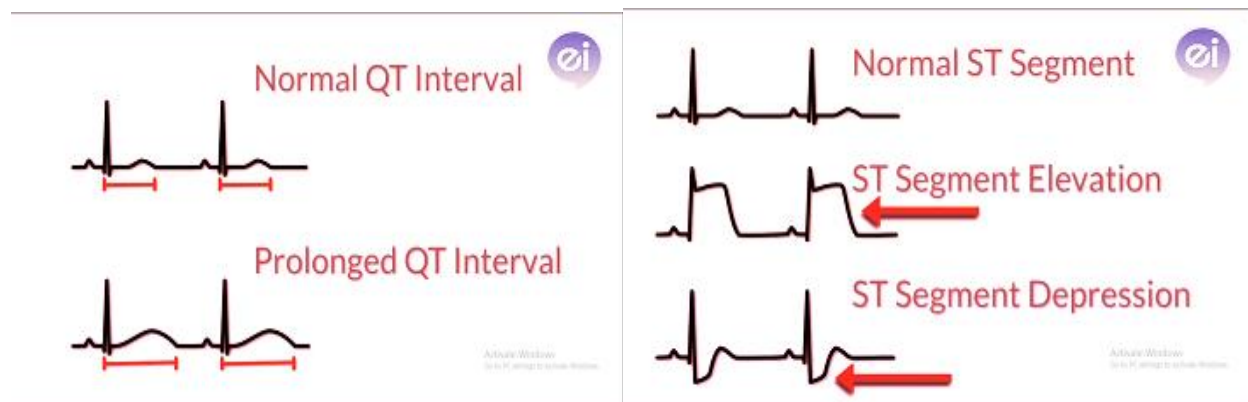
4.) C.S of Heart showing parts

The heart's right side(right atrium) collects venous(deoxygenated blood) via the superior and inferior vena cava. The blood then enters the right ventricle via the tricuspid valve, and gets pumped to the lungs through the pulmonary arteries. The pulmonary arteries divide into right and left pulmonary arteries, which enter the right and left lungs. The arteries divide and redivide to form a capillary bed where the exchange of carbon dioxide and oxygen occurs.

The two pulmonary arteries carry oxygenated blood from the right and left lungs, to the left side of the heart. The blood enters the left atrium and then the left ventricles, which then pump it to the rest of the body through the aorta.



5.) ECG Waveform showing Stages



6.) ECG waveform in patient with heart disease. Note: Deformation of waveform due to disease

The three main components to an ECG:

- The P wave- represents depolarization of the atria.
- The QRS complex- represents depolarization of the ventricles.
- The T wave- represents repolarization of the ventricles.

During each heartbeat, a healthy heart always has an orderly progression of depolarization, starting with pacemaker cells in the sinoatrial node, that then spreads throughout the atrium, and passes through the atrioventricular node down into the bundle of His and into the Purkinje fibres, spreading down and to the left throughout the ventricles. This orderly pattern gives rise to the

characteristic ECG tracing. A trained clinician can deduce a large amount of information about the heart and its electrical activity from the ECG.

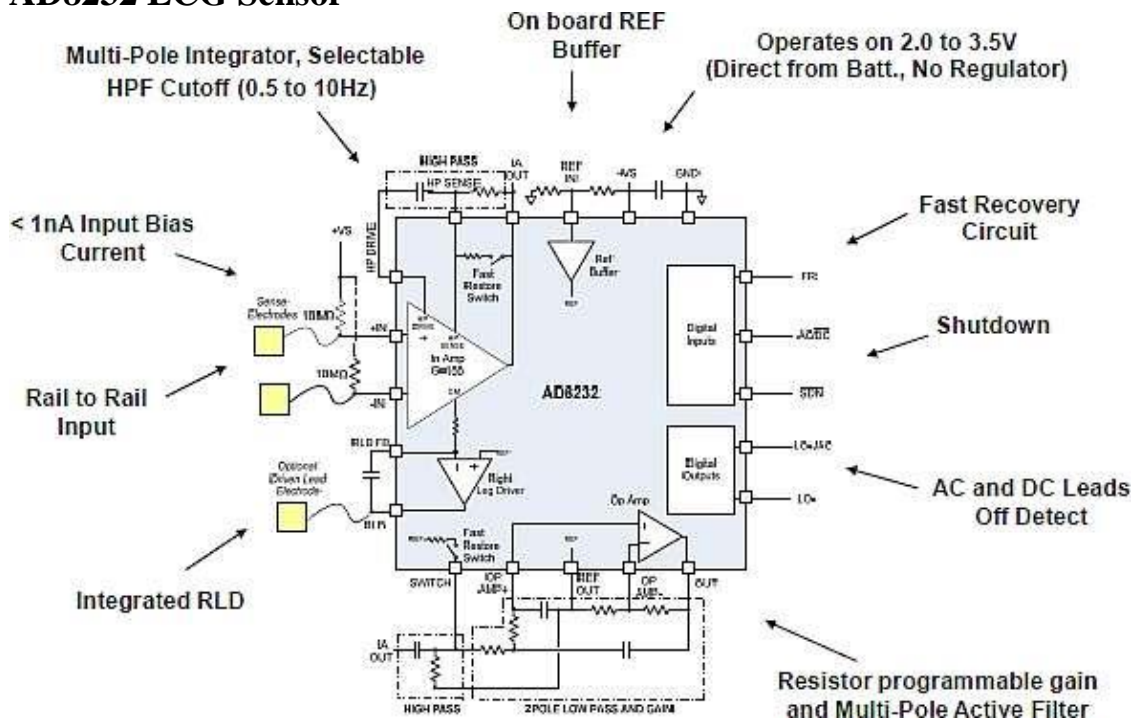
IMPLEMENTATION USING ARDUINO UNO (WIRED VERSION)

We perform a preliminary wired implementation of our project using Arduino Uno and AD8232 ECG Sensor. Arduino Uno is widely used in beginner level electronics projects as it is affordable and open source. It also can be modified depending on the need of the project

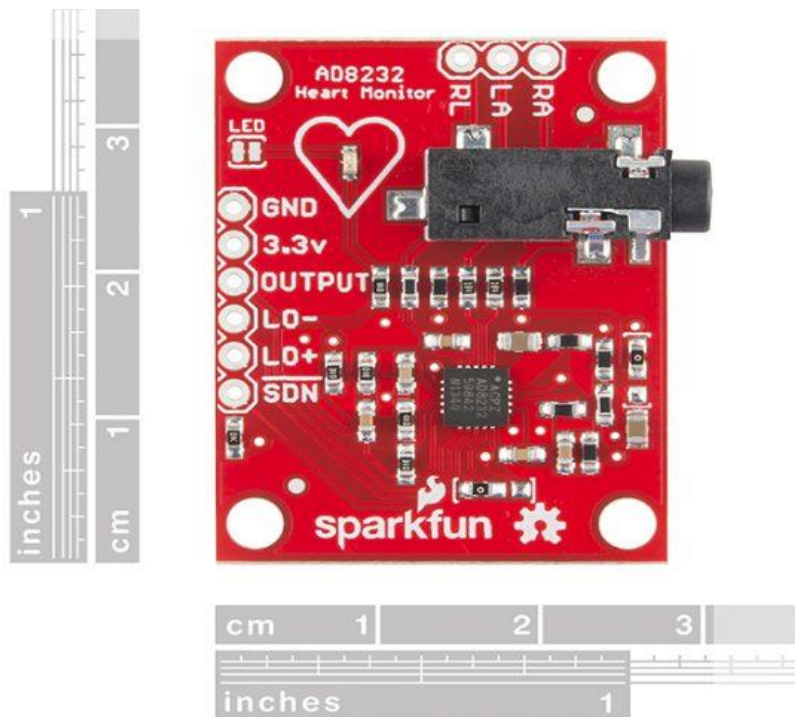
Components:

- Arduino Uno
- AD8232 ECG sensor
- Leads
- Breadboard
- Jumper Wires
- Arduino IDE

AD8232 ECG Sensor



7.) AD8232 ECG Sensor Internal Circuitry



8.) AD8232 ECG Sensor—Magnified view

The AD8232 is an analog frontend chip made by Analog Devices. It is designed to extract, filter and amplify small biopotential signals (ECG or EMG) under noisy conditions.

Feature	Description
Core IC	Analog Devices AD8232 (Analog front-end for biopotential measurement)
Inputs	3-lead ECG electrodes: RA , LA , and RL (Right Arm, Left Arm, Right Leg)
Power Supply	3.3V to 5V (works with Arduino power pins)
Output	Analog ECG signal (can be read via microcontroller's analog input)
Size	Compact — about the size of a coin
Leads Off Detection	Built-in pins to detect when electrodes are not connected

Feature	Description
Filters	High-pass and low-pass filters to clean signal and reduce noise
Applications	ECG, EMG (muscle signals), fitness wearables, bio-signal logging

HOW IT WORKS

The electrodes are attached to the body in a 3-lead configuration. The chip then amplifies and filters the small voltages generated by the heart (which generally are in the mV range). The analog output pin then sends the ECG signal to a microcontroller like Arduino, ESP 32 etc. The waveform can be plotted in real time to detect any abnormalities in heart rate or rhythm.

PINS

Pins	Description
GND	Ground
3.3V	Power Supply
OUTPUT	Analog ECG signal output
LO+/LO-	Leads off detection
SDN	Shutdown-pin(optional)

3 LEAD CONFIGURATION

- RA- Right arm
- LA- Left Arm
- RL- Right Leg or lower body

ARDUINO UNO

Uno is a microcontroller board that is based on 8-bit ATmega328P microcontroller. Along with an ATmega328P, it has other components like a crystal oscillator, serial communication, voltage regulators, etc. to complement the microcontroller.

The board comes with an USB interface, 6 analog input pins, 14 I/O digital ports that are used to connect to external electronic devices. Of the 14 I/O ports, 6 can be used for PWM output. This allows the designers to control and sense external electronic devices in the real world.

The Uno has been a huge hit in the electronics industry and is used by both amateur enthusiasts and professionals alike for a wide variety of projects. It is completely open source, i.e. anyone can purchase the board and download the software(which is free) and can freely modify or update the components for better performance or efficiency. The software is called an IDE(we use Arduino IDE) and uses C/C++ to program devices.



9.) Arduino Uno

Arduino Uno Features:

This board comes with all the features required to run the controller and can be directly connected to the computer through USB cable that is used to transfer the code to the controller using IDE (Integrated Development Environment) software, mainly developed to program Arduino. So, let's dive into the features of Arduino Uno.

1. **More frequency and number of instructions per cycle:** Atmega328 microcontroller is placed on the board that comes with a number of features like timers, counters, interrupts, PWM, CPU, I/O pins and based on a 16MHz clock that helps in producing more frequency and number of instructions/cycle.
2. **Built-in regulation:** This board comes with a built-in regulation feature which keeps the voltage under control when the device is connected to the external device.
3. **Flexibility & Ease of use:** There are 14 I/O digital and 6 analog pins incorporated in the board that allows the external connection with any circuit with the board. These pins provide the flexibility and ease of use to the external devices that can be connected through these pins.

Arduino Uno Pin Description:

The Uno comes with several pins, all with varying functionalities like for ground, reset, voltage supply, voltage reference/regulation, I/O and communication ports for protocols like SPI, I2C etc.

Several of the pins, like the digital, analog and I/O pins

operate at 5V. There are internal pullup resistors in the board to limit the current flowing in the board so that it does not exceed the threshold and damage the device

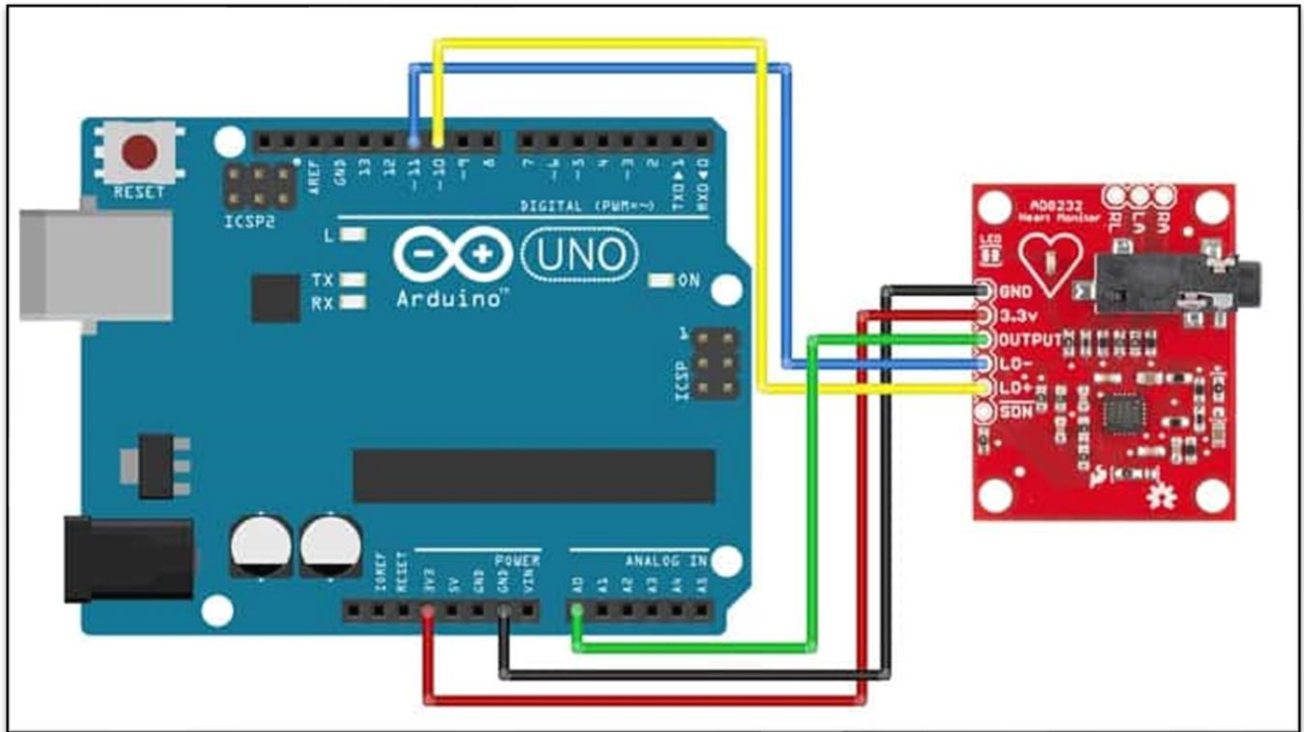
- **Vin** - This is the input voltage supplied to the Arduino Uno. It is different from 5 V supplied through a USB port. The voltage provided through power jack can be accessed through this pin.
- **5V** - This board has the ability to provide voltage regulation and the output regulated voltage is provided through the 5V pin. There are three ways to power the board: the USB, the Vin pin and the DC power jack. The USB supports voltage around 5V while the Vin and Power Jack can support a voltage ranging from 7V to 20V.
- **GND** - The board is equipped with ground pins to ground the circuit. There are many ground pins on the board. They can be used as per requirement.
- **LED** – The Uno comes with a built-in LED which is connected at pin 13. Providing LOW will turn it OFF. When it goes High the LED turns on, while a Low value turns it off.
- **Reset** -This pin resets the program running on the board. Along with a physical reset on the board, the IDE also comes with a feature of resetting the board through programming.
- **IOREF** - The abbreviation for Input Output Voltage Reference. It is used for providing voltage reference to the board.
- **Serial Communication** - It occurs via two pins: Pin 0 being the receiver and Pin 1 being the transmitter
- **Rx. & Tx.** – The Rx (Receiver) pin is used to receive data. The Tx (Transmitter) pin is used to transmit data.
- **External Interrupts** – They are provided through pin 2 and pin 3. An interrupt is initiated through a LOW or changing signal.
- **PWM** –It stands for Pulse Width Modulation. It is provided by 3, 5, 6, 9, 10, 11 pins. These pins have been configured to provide 8-bit output PWM.
- **SPI** - It stands for Serial Peripheral Interface. There are four pins: 10(SS), 11(MOSI), 12(MISO), 13(SCK). They provide SPI communication with the help of SPI library.

AREF - It is the short form of Analog Reference. It is used for providing a reference voltage for the analog signals.

- **TWI** - It stands for Two-Wire Interface. It is a communication protocol based on I2C. It is accessed and implemented through Wire Library. A4 and A5 pins are used for this purpose.

CIRCUIT DIAGRAM/CONNECTION BETWEEN ARDUINO AND ECG SENSOR AD8232

The connections are made according to the diagram given below. In this project the SDN pin is not used. The pin is generally used in case of low powered applications.

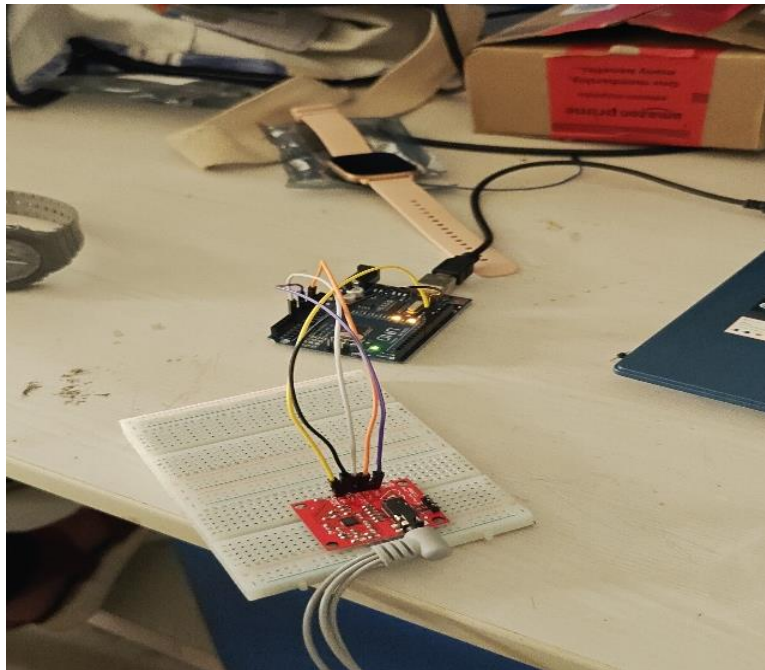


11.) Arduino Uno and AD8232 Connections

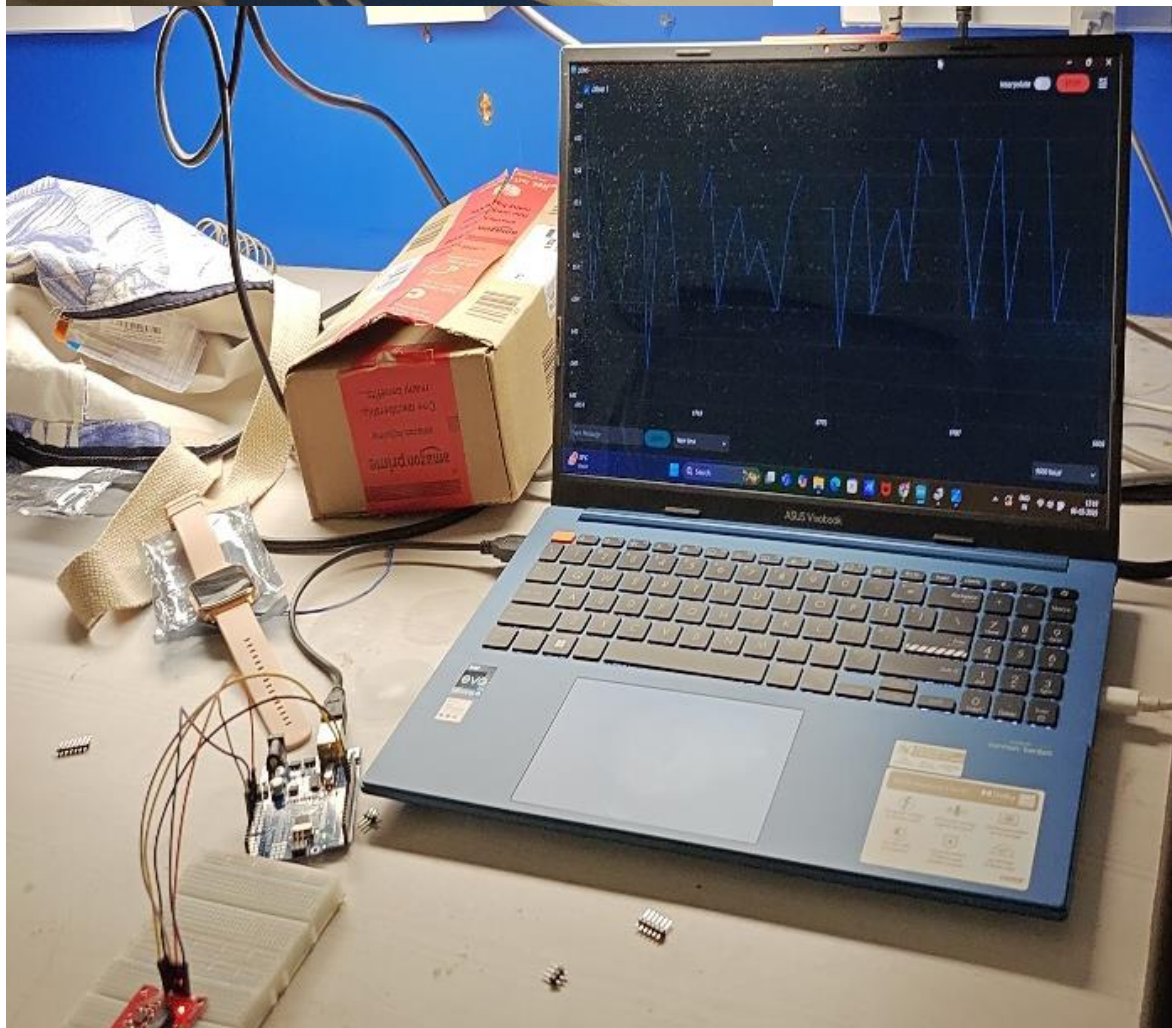
Pin Connections:

There are nine 'pins' present in the AD8232 ECG sensor, of which only five are needed in our device. The five are GND, OUTPUT, LO-,LO+ and 3.3v.

AD8232 Label	Arduino Connection	Colour
GND	GND	Black
3.3v	3.3V	Red
OUTPUT	A0	Green
LO-	11	Blue
LO+	10	Yellow
SDN	Not Used	Null



12.) Lab Implementation-
-- Arduino and Ad8232
Circuit+Output



Arduino Code:

```
sketch_may1b.ino
1  void setup() {
2    // initialize the serial communication:
3    Serial.begin(9600);
4    pinMode(10, INPUT); // Setup for leads off detection LO +
5    pinMode(11, INPUT); // Setup for leads off detection LO -
6
7  }
8
9  void loop() {
10
11    if((digitalRead(10) == 1)|| (digitalRead(11) == 1)){
12      Serial.println('!');
13    }
14    else{
15      // send the value of analog input 0:
16      Serial.println(analogRead(A0));
17    }
18    //Wait for a bit to keep serial data from saturating
19    delay(1);
20  }
```

13.) Lab Implementation--- Code

After that you will see an output like this on Arduino serial plotter.

Result:



14.) Lab Implementation--- Output on Plotter

IMPLEMENTATION USING ESP32: (WIRELESS VERSION)

The wireless version of our project involves using ESP32 due to its wireless capabilities. It has inbuilt WiFi and Bluetooth. It is widely used in IoT applications. We continue to use the 3 lead AD8232 ECG sensor to receive the analog input from the patient.

Components:

- ESP32
- AD8232 ECG sensor
- Breadboard
- Jumper Wires
- Arduino IDE
- Ubidots Platform

ESP32

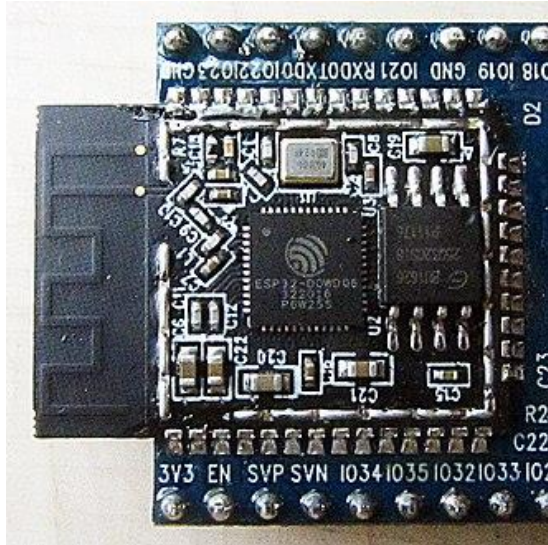
Esp32 is a low cost, low power system on chip (SoC) developed by Espressif Systems. It is widely used in fields like IoT(Internet of Things), embedded systems and smart devices due to its high processing power, integrated wireless capabilities and affordability.

FEATURES

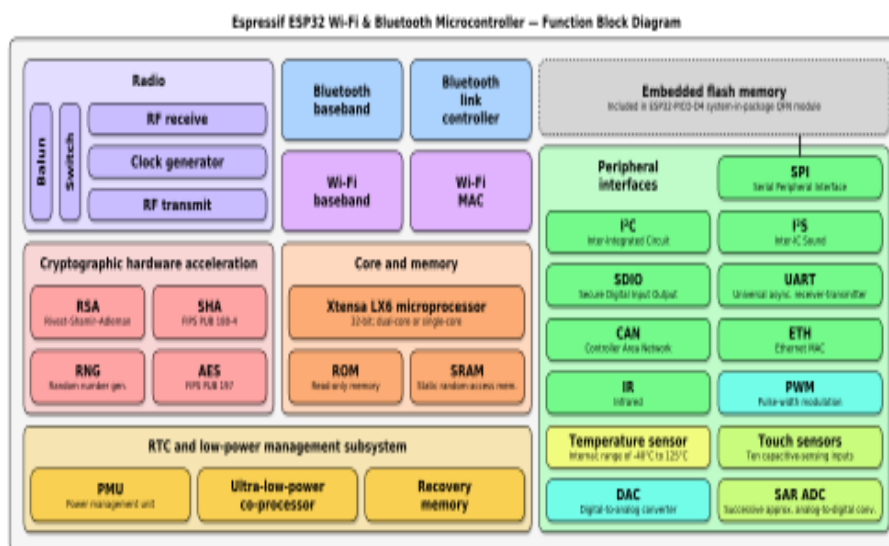
The esp32 possesses a dual core or single core Tensilica Xtensa LX6 32-bit processor that can operate upto frequencies of 240Mhz. It has 520 KB of internal SRAM and supports external storage ranging from 4-16MB. It has both WiFi(IEEE 802.11 b/g/n) and Bluetooth 4.2 integrated and present on-chip, making it highly suitable for IoT applications or applications requiring high connectivity. The chip supports up to 34 GPIO pins, which can be used for digital input/output or repurposed for specialized functions like ADC, DAC, PWM, UART, SPI, I2C, and others. It operates at a range of around 3- 3.3 V and has robust security features like hardware encryption, secure boot and flash encryption. It is widely known for its power efficiency with multiple power modes like active, light sleep, deep sleep, hibernation etc.

PERIPHERALS

The ESP32 is rich in peripherals. It has a 12-bit ADC that supports up to 18 channels, and two 8-bit DAC outputs. It has capacitive touch sensors, which allows for the development of touch-based interfaces. It also supports standard communication protocols like UART, SPI, and I2C, which are necessary for interfacing with sensors, actuators, and displays.

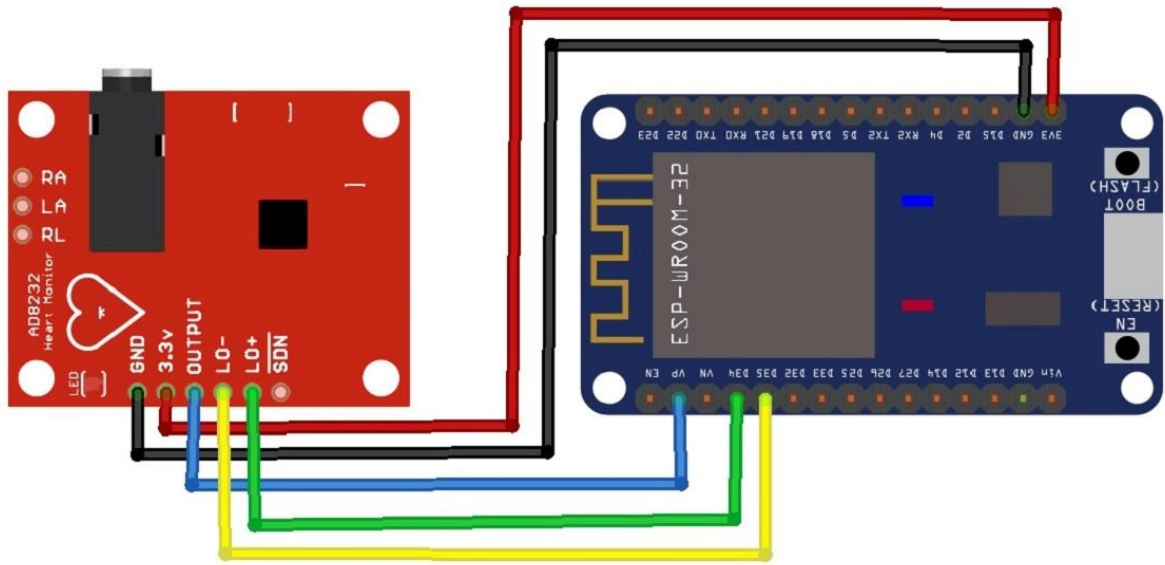


15.) ESP 32



16.) ESP 32: Internal Architecture

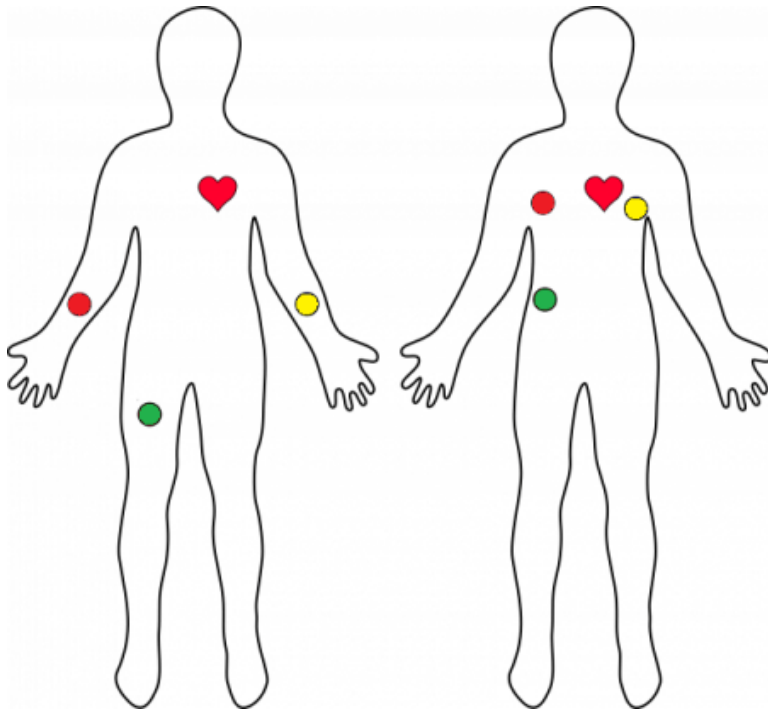
CIRCUIT CONNECTIONS & LEAD PLACEMENTS



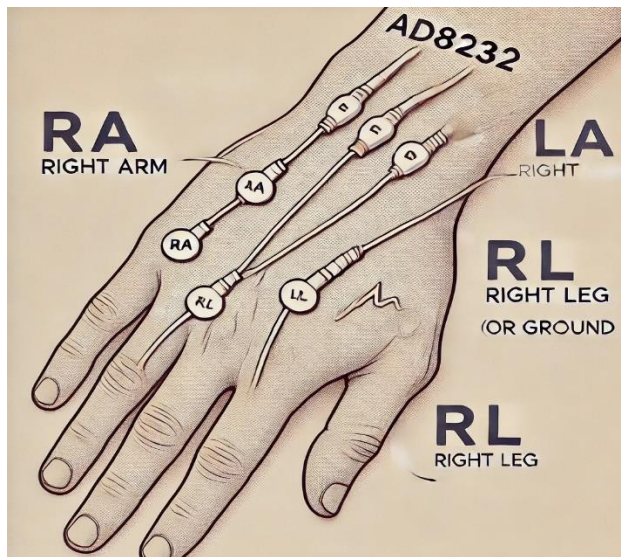
17.) Connection of ESP32 with AD8232 ECG Sensor

Esp32	AD8232	Colour
3.3V	3V3	Red
GND	GND	Black
OUTPUT	VP	Blue
LO+	DE4	Green
LO-	DE5	Yellow
SDN	Not Used	Null

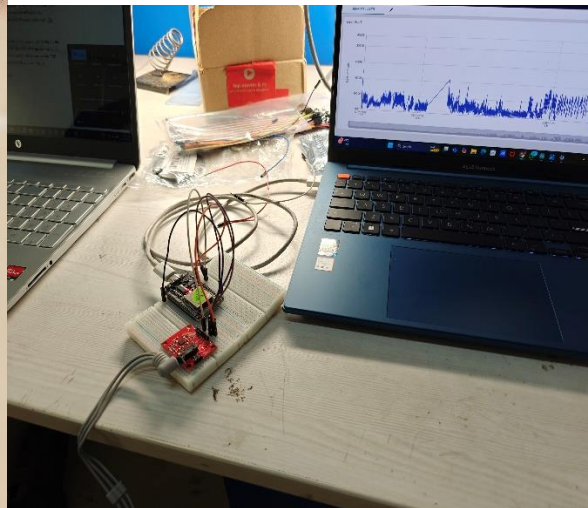
Connection Table of ESP32 with AD8232



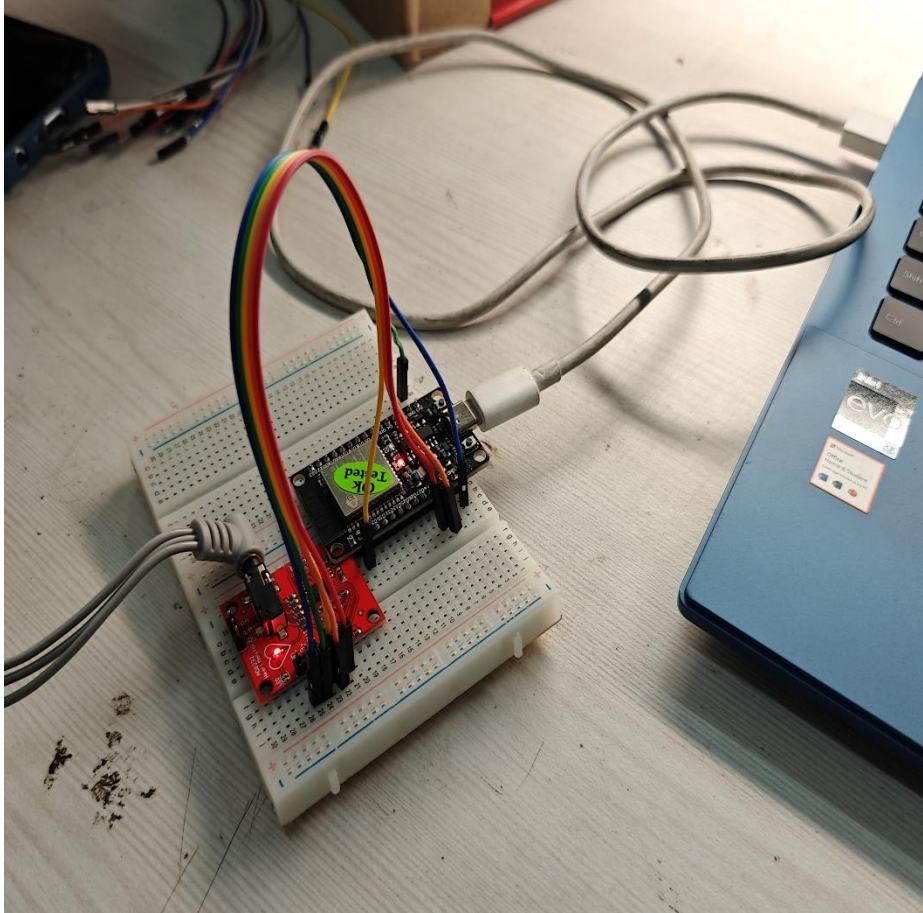
18.) Placement of leads on the body of the patient



19.) Placement of leads on hand



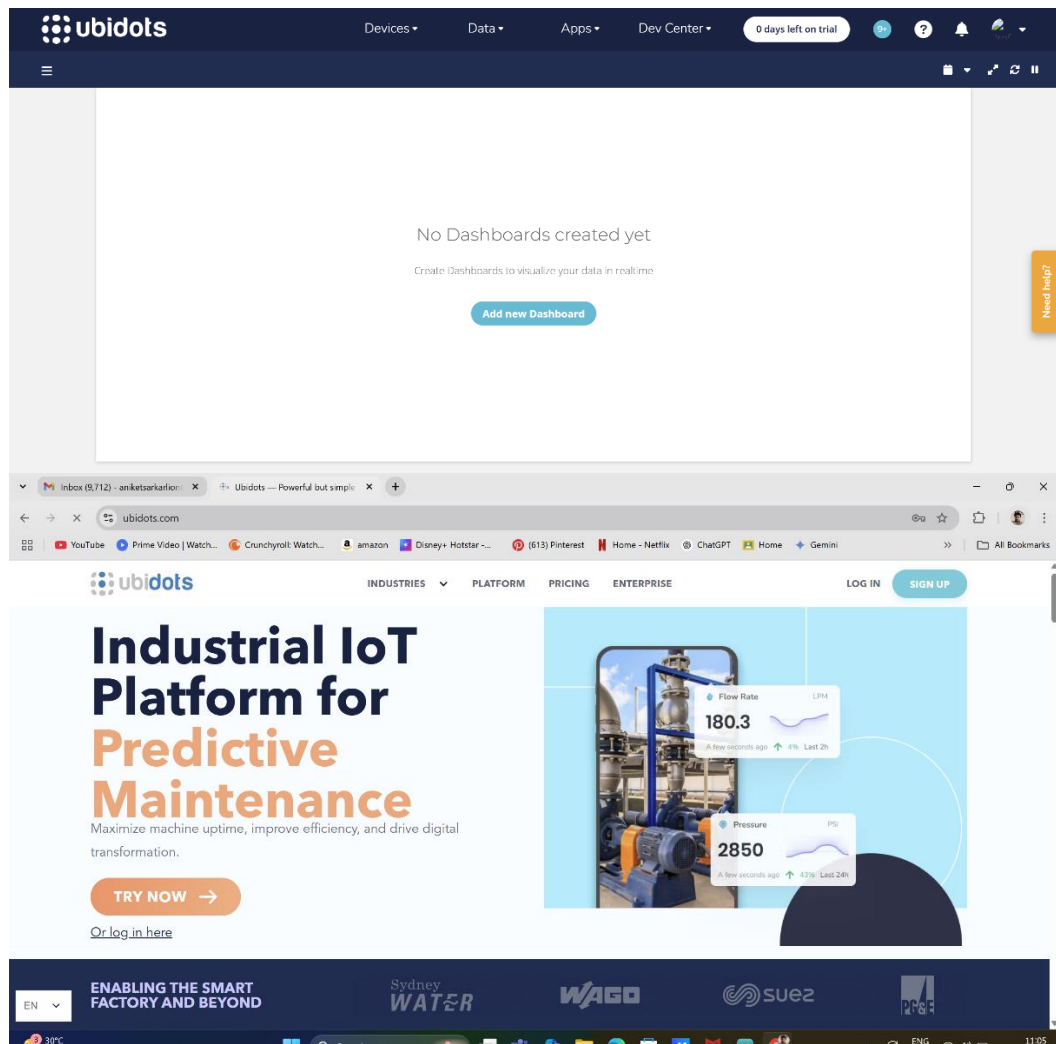
20.) Actual connection between Esp32 and AD8232 performed in lab



21.) Actual Circuitry Image (Enlarged. Wires coalesced into bundle)

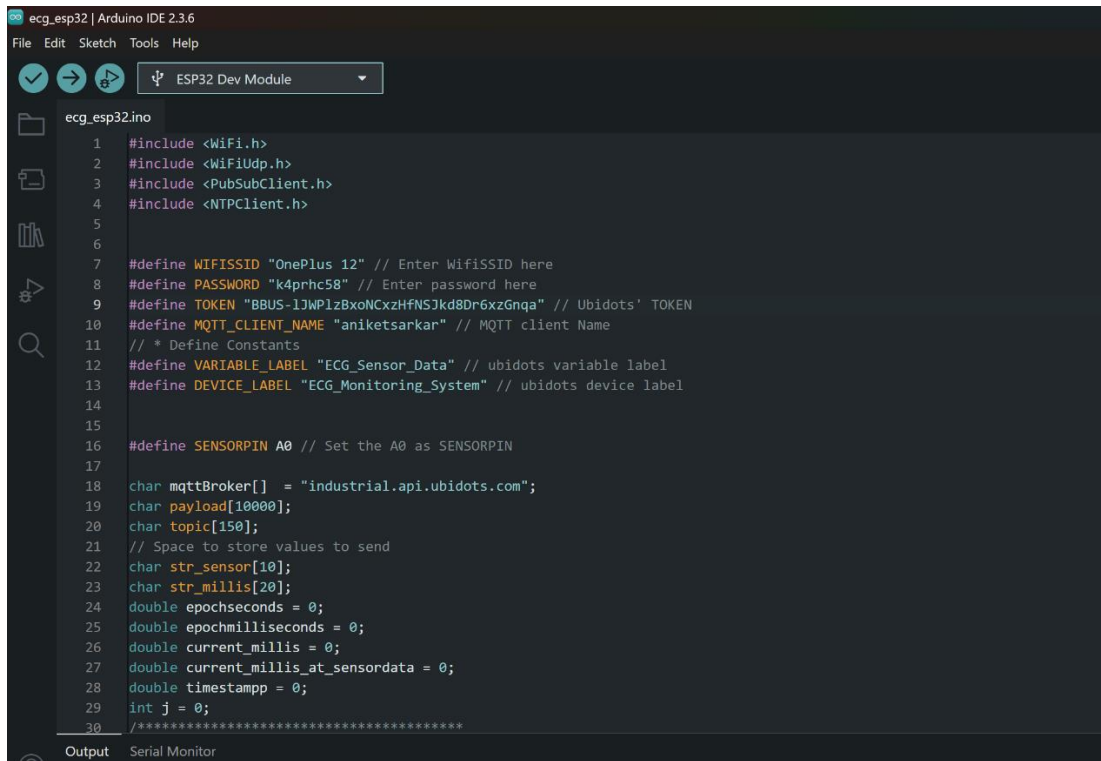
SOURCE CODE & UBIDOTS PLATFORM

Ubidots is a cloud-based Internet of Things (IoT) platform that enables users to connect devices, collect sensor data, visualize it in real time, and automate responses. It is widely used for prototyping, industrial monitoring, and educational IoT projects. It serves as a bridge between physical devices and cloud applications, allowing developers and organizations to build and deploy IoT systems without managing backend infrastructure. It supports real-time analytics, alerting, and dashboard creation.



22.) Ubidots Cloud Platform--Home Page and User dashboard

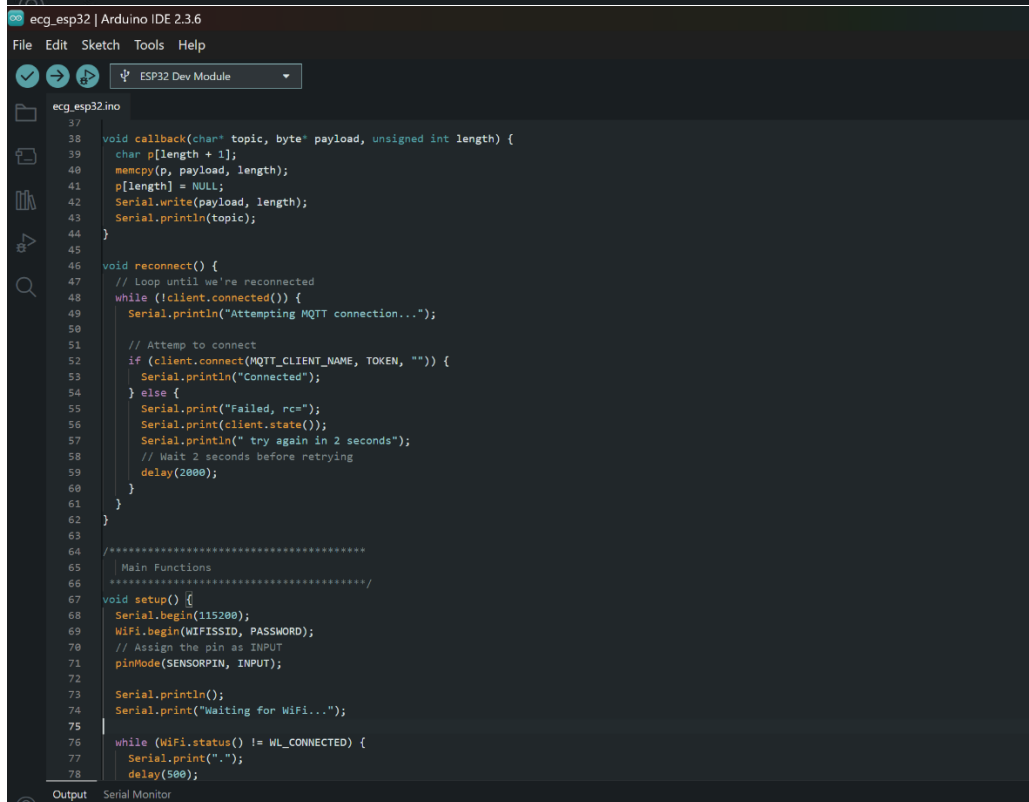
Source Code



```
ecg_esp32 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module

ecg_esp32.ino
1  #include <WiFi.h>
2  #include <WiFiUdp.h>
3  #include <PubSubClient.h>
4  #include <NTPTClient.h>
5
6
7  #define WIFISSID "OnePlus 12" // Enter WifiSSID here
8  #define PASSWORD "k4prhc58" // Enter password here
9  #define TOKEN "BBUS-1JWPlzBxoNCxzHfNSJkd8Dr6xzGnqa" // Ubidots' TOKEN
10 #define MQTT_CLIENT_NAME "aniketsarkar" // MQTT client Name
11 // * Define Constants
12 #define VARIABLE_LABEL "ECG_Sensor_Data" // ubidots variable label
13 #define DEVICE_LABEL "ECG_Monitoring_System" // ubidots device label
14
15
16 #define SENSORPIN A0 // Set the A0 as SENSORPIN
17
18 char mqttBroker[] = "industrial.api.ubidots.com";
19 char payload[1000];
20 char topic[150];
21 // Space to store values to send
22 char str_sensor[10];
23 char str_millis[20];
24 double epochseconds = 0;
25 double epochmilliseconds = 0;
26 double current_millis = 0;
27 double current_millis_at_sensordata = 0;
28 double timestamp = 0;
29 int j = 0;
30 //*****
```

23.) Source Code Part 1--- Library initializations



```
ecg_esp32 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module

ecg_esp32.ino
37
38 void callback(char* topic, byte* payload, unsigned int length) {
39   char p[length + 1];
40   memcpy(p, payload, length);
41   p[length] = NULL;
42   Serial.write(payload, length);
43   Serial.println(topic);
44 }
45
46 void reconnect() {
47   // Loop until we're reconnected
48   while (!client.connected()) {
49     Serial.println("Attempting MQTT connection...");
50
51     // Attempt to connect
52     if (client.connect(MQTT_CLIENT_NAME, TOKEN, "")) {
53       Serial.println("Connected");
54     } else {
55       Serial.print("Failed, rc=");
56       Serial.print(client.state());
57       Serial.println(" try again in 2 seconds");
58       // Wait 2 seconds before retrying
59       delay(2000);
60     }
61   }
62 }
63
64 //*****
65 | Main Functions
66 |*****
67 void setup() {
68   Serial.begin(115200);
69   WiFi.begin(WIFISSID, PASSWORD);
70   // Assign the pin as INPUT
71   pinMode(SENSORPIN, INPUT);
72
73   Serial.println();
74   Serial.print("Waiting for WiFi...");
75
76   while (WiFi.status() != WL_CONNECTED) {
77     Serial.print(".");
78     delay(500);
79   }
```

24.) Source Code Part 2--- void reconnect, callback and setup

```
ecg_esp32 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module

ecg_esp32.ino
61 }
62 }
63
64 /*****
65 | Main Functions
66 | *****/
67 void setup() {
68   Serial.begin(115200);
69   WiFi.begin(WIFISSID, PASSWORD);
70   // Assign the pin as INPUT
71   pinMode(SENSORPIN, INPUT);
72
73   Serial.println();
74   Serial.print("Waiting for WiFi...");
75
76   while (WiFi.status() != WL_CONNECTED) {
77     Serial.print(".");
78     delay(500);
79   }
80
81   Serial.println("");
82   Serial.println("WiFi Connected");
83   Serial.println("IP address: ");
84   Serial.println(WiFi.localIP());
85   timeClient.begin();
86   client.setServer(mqttBroker, 1883);
87   client.setCallback(callback);
88   timeClient.update();
89   epochseconds = timeClient.getEpochTime();
90   epochmilliseconds = epochseconds * 1000;
91   Serial.print("epochmilliseconds=");
92   Serial.println(epochmilliseconds);
93   current_millis = millis();
94   Serial.print("current_millis=");
95   Serial.println(current_millis);
96
97
98 void loop() {
99   if (!client.connected()) {
100     reconnect();
101     j = 0;
102   }
103 }
```

25.) Source Code
part 3-- void setup

26.) Source Code
part 3-- void loop

```
ecg_esp32 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module

ecg_esp32.ino
94 Serial.print("current_millis= ");
95 Serial.println(current_millis);
96 }
97
98 void loop() {
99   if (!client.connected()) {
100     reconnect();
101     j = 0;
102   }
103   //sprintf(payload, "%s", "{\"ECG_Sensor_data\": [{\"value\":1234, \"timestamp\": 1595972075},{\"value\":1111, \"timestamp\": 1595971075},{\"value\":2222, \"timestamp\": 1595970075}]}");
104   j = j + 1;
105   Serial.print("j=");
106   Serial.println(j);
107   sprintf(topic, "%s", "/v1.6/devices/", DEVICE_LABEL);
108   sprintf(payload, "%s", ""); // Cleans the payload
109   sprintf(payload, "{\"%s\": [%s, VARIABLE_LABEL]}", VARIABLE_LABEL); // Adds the variable label
110   for (int i = 1; i <= 3; i++)
111   {
112     float sensor = analogRead(SENSORPIN);
113     dtostrf(sensor, 4, 2, str_sensor);
114     sprintf(payload, "%s(\"value\":", payload); // Adds the value
115     sprintf(payload, "%s %s,", payload, str_sensor); // Adds the value
116     current_millis_at_sensordata = millis();
117     timestamp = epochmilliseconds + (current_millis_at_sensordata - current_millis);
118     dtostrf(timestamp, 10, 0, str_millis);
119     sprintf(payload, "%s \"timestamp\": %s\",", payload, str_millis); // Adds the value
120     delay(150);
121   }
122
123   float sensor = analogRead(SENSORPIN);
124   dtostrf(sensor, 4, 2, str_sensor);
125   current_millis_at_sensordata = millis();
126   timestamp = epochmilliseconds + (current_millis_at_sensordata - current_millis);
127   dtostrf(timestamp, 10, 0, str_millis);
128   sprintf(payload, "%s(\"value\":%s, \"timestamp\": %s)}", payload, str_sensor, str_millis);
129   Serial.println("Publishing data to Ubidots Cloud");
130   client.publish(topic, payload);
131   Serial.println(payload);
132   // client.loop();
133
134 }
135 }
```

CODE EXPLANATIONS

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include <PubSubClient.h>
#include <NTPClient.h>
```

WiFi.h: Enables the ESP32 to connect to a Wi-Fi network.

WiFiUdp.h: Allows sending/receiving UDP packets. Needed by NTPClient to talk to NTP servers.

PubSubClient.h: Handles MQTT communication to send sensor data to Ubidots.

NTPClient.h: Synchronizes the time from online NTP servers to provide accurate timestamps for data logging.

```
#define WIFISSID "OnePlus 12"
#define PASSWORD "k4prhc58"
#define TOKEN "BBUS-IJWPlzBxoNCxzHfNSJkd8Dr6xzGnqa"
#define MQTT_CLIENT_NAME "aniketsarkar"
#define VARIABLE_LABEL "ECG_Sensor_Data"
#define DEVICE_LABEL "ECG_Monitoring_System"
#define SENSORPIN A0
```

WIFISSID: The SSID (name) of your Wi-Fi network.

PASSWORD: Your Wi-Fi password.

TOKEN: The Ubidots token used to authenticate your account.

MQTT_CLIENT_NAME: A unique client ID to identify your device to the MQTT broker.

VARIABLE_LABEL: Name of the variable under which ECG readings will be stored on Ubidots.

DEVICE_LABEL: Name of the virtual device on Ubidots that represents this ESP32 hardware.

SENSORPIN: The analog pin (A0) where the AD8232 ECG sensor is connected.

GLOBAL VARIABLES AND BUFFERS

```
char mqttBroker[] = "industrial.api.ubidots.com"; // MQTT server URL for Ubidots
char payload[10000]; // Buffer to hold the entire MQTT payload (JSON format)
char topic[150]; // Buffer to hold the topic string for MQTT

char str_sensor[10]; // Temporary buffer to store sensor reading as string
char str_millis[20]; // Buffer to store timestamp as string

double epochseconds = 0; // Holds time in seconds since 1970
double epochmilliseconds = 0; // epochseconds converted to milliseconds
double current_millis = 0; // millis() at the time NTP time was fetched
double current_millis_at_sensordata = 0; // millis() at the moment ECG value is captured
double timestamp = 0; // Final timestamp of sensor data in epoch ms
int j = 0; // Loop counter
```

At the beginning, the code performs several crucial tasks to prepare the ESP32 for Wi-Fi, MQTT, sensor reading, and time synchronization. First, it includes important libraries: `WiFi.h` for Wi-Fi connectivity, `WiFiUdp.h` for UDP protocol (used by NTP), `PubSubClient.h` for MQTT communication (to send data to Ubidots), and `NTPClient.h` for fetching accurate internet time. Then, several macros (`#define`) are declared to simplify and centralize key configuration values such as Wi-Fi credentials (`WIFISSID`, `PASSWORD`), Ubidots token (`TOKEN`), MQTT client name, sensor and device labels, and the analog pin connected to the ECG sensor (`SENSORPIN`). After that, global variables are initialized: character arrays for MQTT payload and topic construction, variables for storing sensor values and timestamps, and counters like `j` for keeping track of data points. Wi-Fi and MQTT clients are also instantiated, along with the NTP time client using `pool.ntp.org` as the time server. Additionally, a callback function is defined to handle incoming MQTT messages (though it's not heavily used here), and a `reconnect()` function is created to ensure the ESP32 automatically reconnects to Ubidots if the MQTT connection drops. All of this pre-setup configuration lays the foundation for the ESP32 to operate reliably and transmit ECG data accurately.

VOID SETUP(). (INITIALIZES EVERYTHING. RUNS ONCE)

```
void setup() {  
  Serial.begin(115200);           // Start serial communication for debugging  
  WiFi.begin(WIFISSID, PASSWORD); // Connect to Wi-Fi network  
  pinMode(SENSORPIN, INPUT);      // Set ECG sensor pin as input  
  
  while (WiFi.status() != WL_CONNECTED) { // Wait until Wi-Fi is connected  
    Serial.print(".");  
    delay(500);  
  }  
  
  Serial.println("");  
  Serial.println("WiFi Connected");  
  Serial.println("IP address: ");  
  Serial.println(WiFi.localIP());    // Print assigned IP  
  
  timeClient.begin();               // Start NTP time client  
  client.setServer(mqttBroker, 1883); // Set MQTT broker and port  
  client.setCallback(callback);      // Register callback  
  
  timeClient.update();              // Fetch time from NTP server  
  epochseconds = timeClient.getEpochTime(); // Get epoch time (seconds since 1970)  
  epochmilliseconds = epochseconds * 1000; // Convert to milliseconds  
  current_millis = millis();        // Save millis() at this moment  
}
```

The setup() function in this ECG monitoring project serves as the initialization block that prepares the ESP32 to begin operation. It starts by enabling serial communication at a baud rate of 115200 for debugging via the Serial Monitor. Next, it connects the ESP32 to the specified Wi-Fi network using the credentials defined earlier. While attempting to connect, it prints dots on the serial console to indicate progress and, once connected, displays the IP address assigned to the ESP32. The analog pin connected to the AD8232 ECG sensor is set as an input to allow ECG signal acquisition. Then, the Network Time Protocol (NTP) client is initialized to obtain the current time in UNIX epoch format. This time is converted into milliseconds and stored for later timestamping of the ECG data. Additionally, the MQTT client is configured to connect to the Ubidots cloud broker and is linked to a callback function (though it is not used actively in this code). The current internal system time (millis()) is also saved to align sensor readings with actual timestamps. Altogether, setup() ensures that Wi-Fi, time synchronization, MQTT communication, and sensor input are properly configured before any data is collected or sent.

THE VOID LOOP()(THIS PART OF THE CODE RUNS REPEATEDLY)

```
void loop() {
  if (!client.connected()) {
    reconnect(); // If MQTT is disconnected, reconnect
    j = 0;      // Reset sample counter
  }

  j = j + 1;    // Increment sample counter
  Serial.print("j=");
  Serial.println(j);

  sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL); // e.g., "/v1.6/devices/ECG_Monitoring_System"
  sprintf(payload, "%s", ""); // Clear payload
  sprintf(payload, "{\"%s\": [", VARIABLE_LABEL); // Start JSON array e.g., {"ECG_Sensor_Data": [

  for (int i = 1; i <= 3; i++) { // Collect 3 sensor readings
    float sensor = analogRead(SENSORPIN); // Read ECG signal (0-4095)
    dtostrf(sensor, 4, 2, str_sensor); // Convert float to string

    current_millis_at_sensordata = millis(); // Get time of reading
    timestamp = epochmillisecseconds + (current_millis_at_sensordata - current_millis); // Compute actual epoch
    timestamp

    dtostrf(timestamp, 10, 0, str_millis); // Convert timestamp to string

    // Add reading to JSON payload
    sprintf(payload, "%s{\"value\": %s, \"timestamp\": %s},", payload, str_sensor, str_millis);
    delay(150); // Short delay between readings
  }

  // Final reading (4th value)
  float sensor = analogRead(SENSORPIN);
  dtostrf(sensor, 4, 2, str_sensor);
  current_millis_at_sensordata = millis();
  timestamp = epochmillisecseconds + (current_millis_at_sensordata - current_millis);
  dtostrf(timestamp, 10, 0, str_millis);

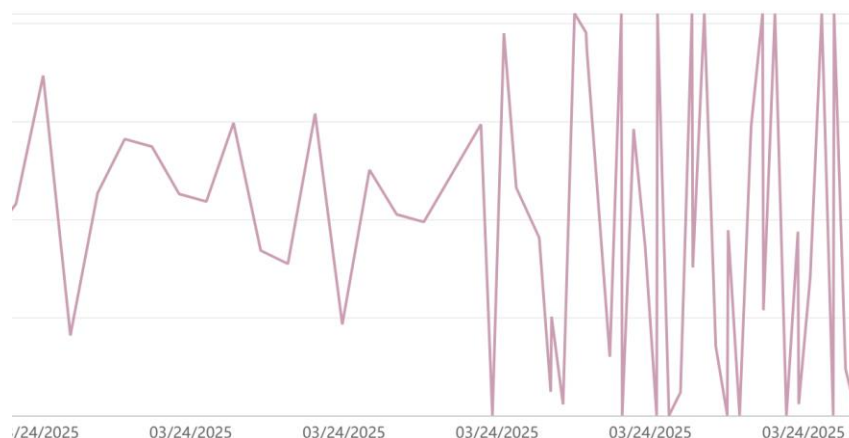
  // Close JSON array
  sprintf(payload, "%s{\"value\":%s, \"timestamp\": %s}]", payload, str_sensor, str_millis);

  Serial.println("Publishing data to Ubidots Cloud");
  client.publish(topic, payload); // Send data to Ubidots
  Serial.println(payload); // Print payload
}
```

The `loop()` function is the heart of the ECG monitoring system—it runs continuously after the `setup()` has completed. Its primary responsibility is to repeatedly collect ECG sensor data, timestamp it, format it correctly, and publish it to the Ubidots cloud. It first checks if the MQTT client is still connected; if not, it calls the `reconnect()` function to restore the connection and resets the data counter `j`. Then, it increments the counter `j` to keep track of how many times data has been published. The topic for MQTT publishing is prepared based on the device label, and the payload is initialized as an empty JSON array to store multiple sensor readings. Inside a loop that runs three times, the function reads ECG data using `analogRead(SENSORPIN)`, converts it to a string with `dtostrf`, and calculates a precise timestamp by combining the initial NTP-synced epoch time with the elapsed time since `setup()` using `millis()`. This ensures each ECG reading is accurately timestamped. These data points are formatted into JSON and appended to the payload. After a short delay (150 ms) between each reading, a final ECG value is read and added to close the JSON array correctly. Finally, the complete payload is published to Ubidots using `client.publish`, allowing remote visualization and analysis. The `loop()` function ensures continuous real-time monitoring and transmission of ECG signals.

Procedure and Output

To connect the wireless ECG system to Ubidots and display ECG data, the ESP32 first connects to a Wi-Fi network using the provided SSID and password via the `WiFi.begin()` function. After successful connection, it sets up MQTT communication with Ubidots by configuring the MQTT broker (`industrial.api.ubidots.com`) and authenticating using the device's unique token. Simultaneously, the ESP32 initializes an NTP client to obtain the current epoch time from an NTP server (`pool.ntp.org`). This time synchronization ensures that each ECG data point is accurately timestamped. The AD8232 ECG sensor is connected to the analog pin A0 of the ESP32, and sensor values are read periodically within the `loop()` function. For every cycle, the ESP32 takes multiple readings (spaced 150ms apart), creating a series of data points. Each value is converted into a JSON object with its corresponding timestamp and added to a JSON array. This payload is then published to the specific Ubidots device and variable topic using the MQTT protocol. On the Ubidots dashboard, users can visualize the ECG data in real time using line charts or data tables. This setup allows for low-cost, wireless ECG monitoring and remote access to biometric data through the cloud.



27.) Output. Note:
The QRS Complex
Peaks

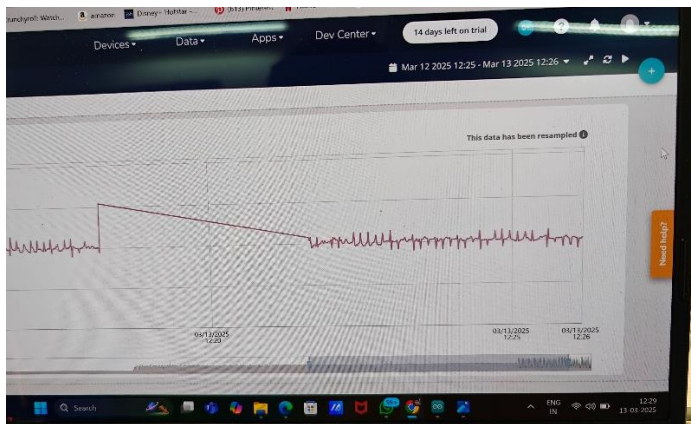
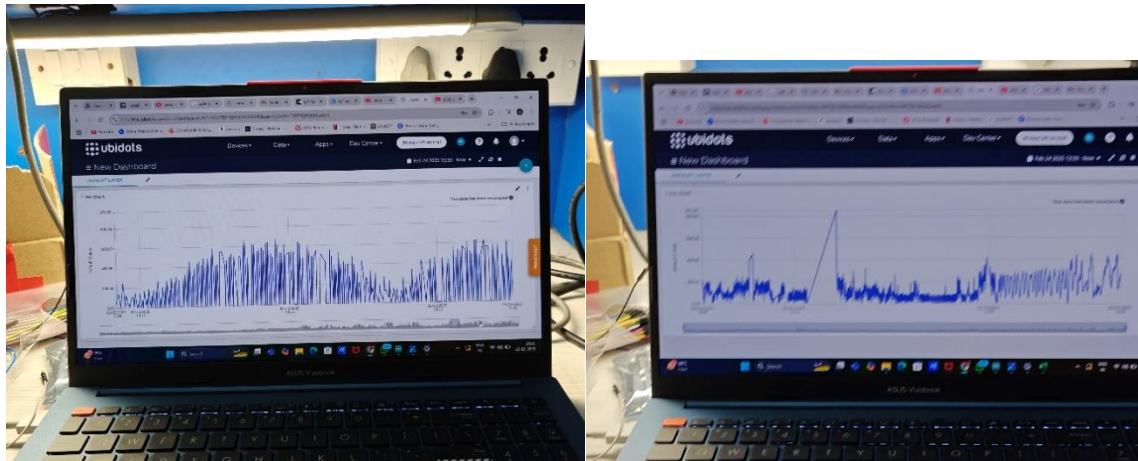
The output of the wireless ECG system is a real-time stream of ECG sensor readings displayed on the Ubidots dashboard. Each reading is time-stamped accurately using synchronized NTP time, ensuring proper tracking of heart activity over time. The data appears as a continuous waveform on a line chart, reflecting the electrical activity of the heart. Users can observe variations in the signal that correspond to heartbeats, enabling basic remote health monitoring. The dashboard also allows customization with widgets like tables or gauges, offering both raw data inspection and visual insights into the patient's cardiovascular condition.

Use of Alternative IoT platforms

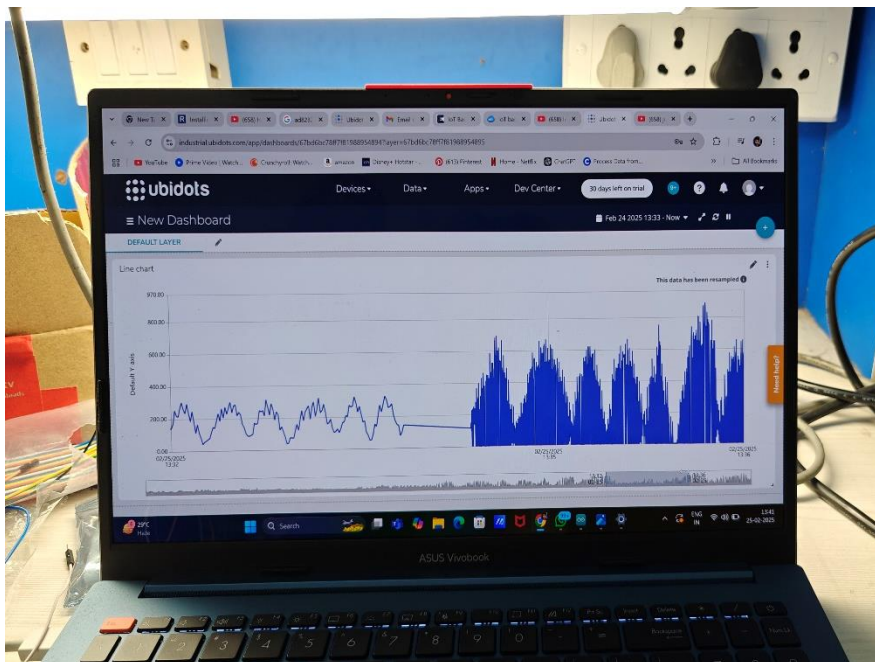
This same project can be implemented using other IoT platforms besides Ubidots. Platforms such as **ThingSpeak**, **Adafruit IO**, **Blynk**, **AWS IoT Core**, and **Google Cloud IoT** offer similar capabilities for data ingestion, visualization, and cloud storage. For example, using **ThingSpeak**, we can send ECG sensor data via HTTP or MQTT protocols and visualize the signal in real-time using built-in MATLAB-based charts. **Adafruit IO** offers a user-friendly interface and MQTT support for pushing sensor data, and its dashboard can display ECG data through gauges or time series graphs. **Blynk**, on the other hand, provides a mobile-first approach where we can create custom app interfaces for viewing ECG readings live on our smartphones. For enterprise-level deployment, **AWS IoT Core** or **Google Cloud IoT** allow scalable data handling, enhanced security, and integration with analytics or AI services to process ECG data for anomaly detection or health trend prediction. While the core principles remain the same—Wi-Fi connectivity, MQTT or HTTP communication, and cloud dashboard setup—the choice of platform depends on scalability, ease of use, integration capabilities, and specific project goals. Thus, the system is flexible and portable across multiple IoT ecosystems with only minor modifications to the code.

	A	B	C	D	E	F	G	H	I	J
1	date	createdAt	timestamp	context	value					
2	2025-02-21	1.74E+12	1.74E+12	()	0					
3	2025-02-21	1.74E+12	1.74E+12	()	393					
4	2025-02-21	1.74E+12	1.74E+12	()	0					
5	2025-02-21	1.74E+12	1.74E+12	()	544					
6	2025-02-21	1.74E+12	1.74E+12	()	48					
7	2025-02-21	1.74E+12	1.74E+12	()	377					
8	2025-02-21	1.74E+12	1.74E+12	()	80					
9	2025-02-21	1.74E+12	1.74E+12	()	41					
10	2025-02-21	1.74E+12	1.74E+12	()	3					
11	2025-02-21	1.74E+12	1.74E+12	()	217					
12	2025-02-21	1.74E+12	1.74E+12	()	89					
13	2025-02-21	1.74E+12	1.74E+12	()	195					
14	2025-02-21	1.74E+12	1.74E+12	()	0					
15	2025-02-21	1.74E+12	1.74E+12	()	366					
16	2025-02-21	1.74E+12	1.74E+12	()	48					
17	2025-02-21	1.74E+12	1.74E+12	()	124					
18	2025-02-21	1.74E+12	1.74E+12	()	0					
19	2025-02-21	1.74E+12	1.74E+12	()	371					
20	2025-02-21	1.74E+12	1.74E+12	()	17					
21	2025-02-21	1.74E+12	1.74E+12	()	297					
22	2025-02-21	1.74E+12	1.74E+12	()	0					
23	2025-02-21	1.74E+12	1.74E+12	()	406					
24	2025-02-21	1.74E+12	1.74E+12	()	0					
25	2025-02-21	1.74E+12	1.74E+12	()	321					
26	2025-02-21	1.74E+12	1.74E+12	()	0					
27	2025-02-21	1.74E+12	1.74E+12	()	495					
28	2025-02-21	1.74E+12	1.74E+12	()	0					
29	2025-02-21	1.74E+12	1.74E+12	()	267					
30	2025-02-21	1.74E+12	1.74E+12	()	0					
31	2025-02-21	1.74E+12	1.74E+12	()	348					
32	2025-02-21	1.74E+12	1.74E+12	()	0					
33	2025-02-21	1.74E+12	1.74E+12	()	402					
34	2025-02-21	1.74E+12	1.74E+12	()	0					
35	2025-02-21	1.74E+12	1.74E+12	()	505					
36	2025-02-21	1.74E+12	1.74E+12	()	0					
37	2025-02-21	1.74E+12	1.74E+12	()	269					
38	2025-02-21	1.74E+12	1.74E+12	()	0					
39	2025-02-21	1.74E+12	1.74E+12	()	448					
40	2025-02-21	1.74E+12	1.74E+12	()	0					
41	2025-02-21	1.74E+12	1.74E+12	()	574					
42	2025-02-21	1.74E+12	1.74E+12	()	0					

28.) ECG data excel sheet snapshot
(Note: the intermittent high values caused by peaks of the ecg graph)



29.) Outputs taken at various times during the experimentation performed in the laboratory. Note: The noise is significant if the patient is not still.



OUTPUT NOISE

There is slight noise present as a result of connectivity issues, internet bandwidth limitations and sensor accuracy limitations. In these kinds of projects, noise in the output can arise from various sources, and is seen to result in inaccurate or distorted readings. Common causes include, electromagnetic interference (EMI) from nearby electronic devices, power supply fluctuations, and motion artifacts caused by movement or muscle contractions. Also, poor sensor contact with the skin can lead to fluctuating or inconsistent signals. The AD8232 ECG sensor sometimes also picks up noise due to ambient electrical fields or long wire connections. Some common noise types that affected the output in our project probably came from powerline interference, which often manifests as a 50/60 Hz hum and motion artifacts that mimics heartbeats. Another common cause of error is when there is patient movement or improper electrode placement causing baseline wander. These noises can be minimized by using shielded cables and ensuring proper grounding. Placing the device in environments free from strong electromagnetic fields helps avoid unnecessary EMI disturbances. Low-pass filters can also be implemented to filter out high-frequency noise and using notch filters is effective for removing powerline interference. To address motion artifacts, proper electrode placement and the usage of wet electrodes is recommended. Additionally, ensuring stable power supply regulation can reduce power supply noise. By using various signal processing techniques (for example, digital filtering) the ECG can be cleared up further improving clarity and accuracy.

FUNDAMENTAL PILLARS OF THIS PROJECT

The core concepts of this wireless ECG monitoring project revolve around the integration of biomedical signal acquisition with modern IoT technologies to enable real-time, remote health monitoring. At the heart of the system is the **AD8232 ECG sensor**, which captures the electrical activity of the heart and outputs it as an analog signal. This signal is then read by the **ESP32 microcontroller**, which uses its built-in **Analog-to-Digital Converter (ADC)** to digitize the ECG waveform. The ESP32 also plays a critical role in the **wireless communication** aspect of the project, as it connects to a Wi-Fi network to transmit the data. To ensure the ECG readings are time-stamped accurately, the system uses **Network Time Protocol (NTP)** via the NTPClient library to fetch epoch time from internet servers. This enables precise synchronization between the sensor data and its corresponding timestamps. Data is transmitted using the **MQTT protocol**, a lightweight and efficient method suited for IoT applications. The readings, along with their timestamps, are published to the **Ubidots IoT platform**, where they are stored, processed, and visualized through customizable dashboards. The cloud platform serves as both a data logger and a real-time monitoring interface. This approach demonstrates essential IoT principles like **device-to-cloud communication**, **cloud-based data visualization**, and **remote monitoring**, while also incorporating key biomedical engineering concepts like ECG signal acquisition and analysis. Altogether, the project showcases a practical and scalable solution for wireless, cloud-connected health monitoring, particularly useful in telemedicine, patient mobility, and rural healthcare scenarios.

THE CLOUD INTEGRATION

The cloud integration, NTP time synchronization, and the MQTT protocol are fundamental components of this wireless ECG monitoring system, enabling seamless communication, data accuracy, and real-time visualization. Cloud integration is achieved through **Ubidots**, an Internet of Things (IoT) platform that allows the ESP32 microcontroller to send ECG data over the internet and display it in a user-friendly interface. The ESP32 connects to the internet via Wi-Fi and publishes sensor data directly to Ubidots using the MQTT protocol. This integration allows users to remotely monitor ECG signals on dashboards that support real-time updates, graphical visualizations, and historical data storage.

To ensure that each ECG reading is correctly time-stamped, the system utilizes **NTP (Network Time Protocol)**. The ESP32 connects to an NTP server (such as pool.ntp.org) to fetch the current epoch time (number of seconds since January 1, 1970). This epoch time is converted to milliseconds and used to synchronize the timestamps of the ECG data. Without this step, readings would lack temporal accuracy, making clinical interpretation difficult.

MQTT (Message Queuing Telemetry Transport) is a lightweight, efficient communication protocol designed for constrained devices and low-bandwidth networks. It follows a publish-subscribe model where the ESP32 publishes ECG data to a specific topic on the Ubidots MQTT broker. This method reduces overhead and ensures fast, reliable data transmission. Additionally, the use of MQTT enables bi-directional communication if needed—commands or configuration changes can also be sent back to the ESP32 from the cloud.

Other important aspects include the conversion of analog sensor data into digital format using the ESP32's **ADC**, formatting the data in **JSON**, and organizing readings with precise timestamps. These technologies work together to form a reliable, real-time ECG monitoring system that demonstrates modern IoT and biomedical integration.

FUTURE SCOPE

This wireless ECG monitoring system, using the AD8232 sensor, ESP32 microcontroller, and Ubidots platform holds significant potential for future development, especially in the context of remote healthcare, personal health tracking, and IoT-enabled medical diagnostics. One of the most impactful future directions is **real-time visualization and alert systems**—integrating mobile or web-based dashboards that display ECG waveforms in real time, with automatic alerts for abnormal heart activity (e.g., arrhythmia, bradycardia, or tachycardia). This can be enhanced with **edge computing techniques**, where basic signal processing (such as peak detection, heart rate calculation, and noise filtering using moving average or bandpass filters) is done on the ESP32 itself, reducing reliance on cloud processing and making the system faster and more reliable. For greater accuracy, **hardware enhancements** like using higher-resolution ADCs (analog-to-digital converters), shielded cables, and proper electrode placement can reduce noise and improve signal quality. The ECG sampling rate can also be optimized; currently around 6–7 Hz due to delay(150), it could be raised to 200–250 Hz (the standard for ECG) by using timer interrupts or RTOS-based scheduling on the ESP32. Data compression techniques or more efficient payload formatting (e.g., CBOR or MsgPack) could be implemented to reduce bandwidth and MQTT payload size, allowing more frequent and efficient data uploads. Additionally, **on-device machine learning models** trained to detect cardiac anomalies could be integrated using TensorFlow Lite for Microcontrollers, enabling intelligent edge health monitoring. To support clinical usage, multi-lead ECG monitoring can be considered using multiple AD8232 modules and synchronized readings. For security and privacy, especially in healthcare environments, **encryption** of MQTT messages (e.g., using SSL/TLS) and **OAuth-based authentication** with Ubidots or similar cloud services will be important. Scalability can be improved by supporting batch uploads and offline data caching in flash memory when Wi-Fi is unavailable. A rechargeable battery and power-efficient design (e.g., deep sleep modes) can make the system wearable and portable for continuous long-term monitoring. The project could also integrate **GPS or motion sensors** to add context to the ECG data, helping to differentiate between heart rate changes due to physical activity versus medical issues. Furthermore, interoperability with **FHIR/HL7** medical data standards could allow integration with hospital information systems, making the system suitable for professional use. Lastly, user interfaces such as mobile apps or voice assistants can be added to give users intuitive access to their cardiac health information, enabling proactive health management. Overall, this project, with its low cost, wireless connectivity, and cloud integration, can evolve from a basic biomedical prototype into a powerful, scalable tool for personal and professional cardiac monitoring through a combination of signal quality improvement, advanced analytics, edge AI, and patient-centric interfaces.

REFERENCES

- [1] Wikipedia contributors, “ESP32,” *Wikipedia*, <https://en.wikipedia.org/wiki/ESP32> (accessed May 4, 2025).
- [2] Wikipedia contributors, “Arduino Uno,” *Wikipedia*, https://en.wikipedia.org/wiki/Arduino_Uno (accessed May 4, 2025).
- [3] Wikipedia contributors, “AD8232,” *Wikipedia*, <https://en.wikipedia.org/wiki/AD8232> (accessed May 4, 2025).
- [4] How2Electronics, “ECG Graph Monitoring with AD8232 ECG Sensor & ESP32 on Ubidots,” *How2Electronics*, [Online]. Available: <https://how2electronics.com/ecg-monitoring-ad8232-esp32-ubidots-cloud/> (accessed May 4, 2025).
- [5] Electronics Innovation, “AD8232 ECG Module with Arduino,” *Electronics Innovation*, [Online]. Available: <https://electronicsinnovation.com/interfacing-ad8232-ecg-module-with-arduino/> (accessed May 4, 2025).
- [6] Ubidots Help Center, “Getting Started with Ubidots MQTT API,” *Ubidots*, [Online]. Available: <https://help.ubidots.com/en/articles/1979518-getting-started-with-the-ubidots-mqtt-api> (accessed May 4, 2025).
- [7] Espressif Systems, *ESP32 Technical Reference Manual*, [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (accessed May 4, 2025).
- [8] M. Mahfouz and Y. Zhang, “Wireless ECG Monitoring Systems: A Review,” *Journal of Healthcare Engineering*, vol. 5, no. 3, pp. 203–228, 2014.
- [9] Ubidots, “ESP32 MQTT Example Code,” *GitHub*, [Online]. Available: <https://github.com/ubidots/ubidots-mqtt-esp> (accessed May 4, 2025).
- [10] Analog Devices Inc., *AD8232: Heart Rate Monitor Front End Datasheet*, [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf> (accessed May 4, 2025).
- [11] F. Akhter and F. Ahmed, “IoT-Based Health Monitoring Systems for Remote Patient Care,” in *IoT Applications for Healthcare Systems*, IGI Global, 2020.
- [12] Arduino, “analogRead() – Arduino Reference,” *Arduino.cc*, [Online]. Available: <https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/> (accessed May 4, 2025).
- [13] Random Nerd Tutorials, “Getting Started with ESP32 using Arduino IDE,” *Random Nerd Tutorials*, [Online]. Available: <https://randomnerdtutorials.com/getting-started-with-esp32/> (accessed May 4, 2025).
- [14] NTP.org, “NTP: The Network Time Protocol,” *ntp.org*, [Online]. Available: <https://www.ntp.org/> (accessed May 4, 2025).
- [15] M. Rasheed, A. Qamar, and M. R. Asghar, “A Low-Cost Real-Time IoT-Based ECG Monitoring System Using ESP32,” in *Proc. IEEE Int. Conf. on Emerging Technologies (ICET)*, Islamabad, Pakistan, Nov. 2021, pp. 1–5.
- [16] A. Elayan, O. Aloqaily, and R. Guizani, “Edge AI for Health: Vision and Challenges,” *IEEE Access*, vol. 9, pp. 1101–1117, 2021, doi: 10.1109/ACCESS.2020.3045609.

APPENDIX I

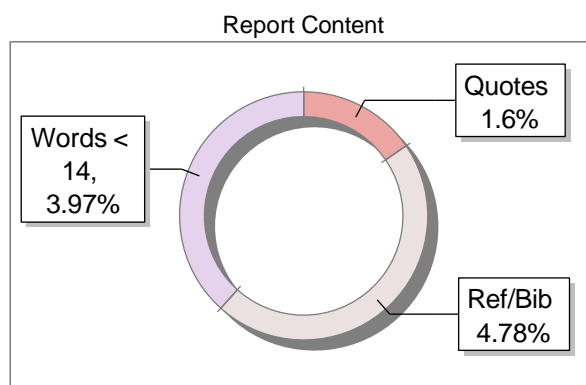
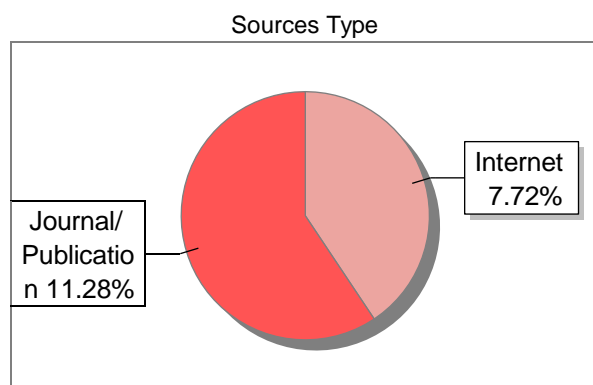
As part of the academic integrity and quality assurance process, our project report titled "*Design and Development of Wireless ECG Device*" was submitted to the DrillBit plagiarism detection software for verification. The similarity check returned a result of **19%**, which falls under the **Grade B (Upgrade)** category, as per DrillBit's classification scale. This percentage indicates that the report contains some similarity with existing sources, primarily from journal publications and publicly available web content. Importantly, no direct copy-pasting of material without understanding or acknowledgment has occurred. Most of the detected similarities are related to standard technical terms, procedural descriptions, and properly referenced technical background content. The plagiarism detection system also includes small word matches (less than 14 words) and reference sections, which can naturally raise the percentage. Our team has ensured that all relevant sources are cited appropriately, and originality has been maintained throughout the report. The results confirm that the document upholds acceptable academic standards and reflects genuine effort and understanding by the authors. This verification step strengthens the credibility of our work and demonstrates our commitment to ethical academic practices and responsible research writing.

Submission Information

Author Name	Aniket Sarkar , Ritwik Saha , Swastika Banerjee , Mousumi Sinha
Title	DESIGN AND DEVELOPMENT OF WIRELESS ECG DEVICE
Paper/Submission ID	3588193
Submitted by	ece@heritageit.edu
Submission Date	2025-05-07 13:34:44
Total Pages, Total Words	36, 6125
Document type	Thesis

Result Information

Similarity **19 %**



Exclude Information

Quotes	Not Excluded
References/Bibliography	Not Excluded
Source: Excluded < 14 Words	Not Excluded
Excluded Source	0 %
Excluded Phrases	Not Excluded

Database Selection

Language	English
Student Papers	Yes
Journals & publishers	Yes
Internet or Web	Yes
Institution Repository	Yes

A Unique QR Code use to View/Download/Share Pdf File





DrillBit Similarity Report

19

SIMILARITY %

36

MATCHED SOURCES

B

GRADE

A-Satisfactory (0-10%)

B-Upgrade (11-40%)

C-Poor (41-60%)

D-Unacceptable (61-100%)

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	203.201.63.468080	3	Publication
2	www.scribd.com	3	Internet Data
3	www.sonatech.ac.in	3	Publication
4	www.scribd.com	2	Internet Data
5	engessays.com	1	Internet Data
6	www.heritageit.edu	<1	Publication
7	scholar.sun.ac.za	<1	Publication
8	Topological foundations of electrocardiololgy, by Greensite, Fred, Yr-1985	<1	Publication
9	arxiv.org	<1	Publication
10	ijaem.net	<1	Publication
11	pg.its.edu.in	<1	Publication
12	arxiv.org	<1	Publication
13	ad.tgnddoors.com	<1	Internet Data
14	translate.google.com	<1	Internet Data

15	patents.google.com	<1	Internet Data
16	www.elprocus.com	<1	Internet Data
17	A Case Study on Applying MQTT Communication to Improve Cold Storage Safety and By Huseyin Karaolu, Tark Kabak, Yr-2024,4,30	<1	Publication
18	www.dx.doi.org	<1	Publication
19	ijetms.in	<1	Publication
20	turcomat.org	<1	Publication
21	www.ijcrt.org	<1	Publication
22	Continuing Medical Education Exam 2 October 2008 by -2008	<1	Publication
23	how2electronics.com	<1	Internet Data
24	www.studocu.com	<1	Internet Data
25	www.tcetmumbai.in	<1	Publication
26	casino-ilman-rekisteroitymista.fi	<1	Internet Data
27	docplayer.net	<1	Internet Data
28	frontiersin.org	<1	Internet Data
29	ijarsct.co.in	<1	Publication
30	pdfcookie.com	<1	Internet Data
31	Thesis Submitted to Shodhganga Repository	<1	Publication
32	Thesis Submitted to Shodhganga Repository	<1	Publication
33	www-pub.iaea.org	<1	Publication

34	www.academia.edu	<1	Internet Data
35	www.analog.com	<1	Publication
36	www.ijert.org	<1	Internet Data