Project Report On

CLOUD BASED IOT ASISTED ECG

Submitted By: -

Bikash Rao | *120BM0805*

Under the Guidance of:

Prof. Bala Chakravarthy Neelapu

Assistant Professor, Dept Of BMBT NIT-Rkl

November 2023

BM4701 : Microprocessor Microcontroller Laboratory

Department of Biotechnology & Medical Engineering

National Institute of Technology,

Rourkela-769008, Odisha, India

Introduction

In the rapidly advancing landscape of healthcare technology, the integration of Internet of Things (IoT) and cloud computing has ushered in a new era of personalized and connected medical devices. One such groundbreaking project is the development of an IoT-enabled, cloud-based Electrocardiogram (ECG) device, akin to a Holter monitor but with the added capability of real-time ECG display. This innovative device represents a convergence of cutting-edge technologies to provide a seamless and comprehensive solution for continuous cardiac monitoring.

Cardiovascular diseases remain a leading cause of morbidity and mortality worldwide, emphasizing the critical need for advanced monitoring tools that can offer timely and accurate insights into cardiac health. Traditional Holter monitors have been instrumental in providing extended ECG recordings, yet their limitations in real-time data transmission and accessibility have prompted the exploration of more sophisticated solutions.

The proposed IoT-enabled cloud-based ECG device not only captures continuous ECG data but also leverages the power of the Internet to transmit this information in real-time to a secure cloud platform. This cloud-based architecture enables healthcare professionals to remotely access and monitor patients' ECG data promptly, leading to quicker intervention and improved patient outcomes.

Key features of this project include:

Continuous Monitoring: The device ensures 24/7 monitoring, allowing for the early detection of irregularities and abnormalities in the cardiac rhythm.

Real-time Data Transmission: Utilizing IoT connectivity, the device facilitates instantaneous transmission of ECG data to a secure cloud server, enabling healthcare providers to access real-time information.

Cloud-Based Analytics: The cloud platform employs advanced analytics to process and interpret ECG data, providing meaningful insights into the patient's cardiac health.

User-Friendly Interface: The device boasts a user-friendly interface for both healthcare professionals and patients, ensuring ease of use and seamless integration into existing healthcare systems.

Secure Data Storage: Emphasis is placed on data security, with robust encryption protocols and compliance with healthcare data privacy standards to safeguard patient information.

This project represents a significant leap forward in cardiac monitoring technology, offering a holistic solution that addresses the limitations of traditional monitoring devices. By harnessing the capabilities of IoT and cloud computing, our aim is to enhance the quality of patient care, enable early intervention, and contribute to the advancement of personalized medicine in the realm of cardiovascular health.

Requirements and Specifications

Arduino Nano:

The Arduino Nano is a compact and versatile microcontroller board based on the ATmega328P. It provides a user-friendly platform for prototyping and developing electronic projects.

The Arduino Nano serves as the brain of your ECG device, handling data acquisition from the AD8232 module, processing the data, and facilitating communication with the HC-05 Bluetooth module for wireless connectivity.

HC-05 Bluetooth Module:

The HC-05 is a popular Bluetooth module that allows for wireless communication between devices over short distances.

The HC-05 module facilitates the wireless transmission of ECG data from the Arduino Nano to other Bluetooth-enabled devices, such as smartphones or computers. This enables real-time monitoring and data retrieval.

ECG Recording Module (AD8232):

The AD8232 is a dedicated integrated circuit for ECG signal acquisition and processing. It includes instrumentation amplifiers and filters to enhance the quality of ECG signals.

The AD8232 module captures the electrical signals generated by the heart through the surface electrodes. It amplifies and filters these signals, providing clean and reliable ECG data to the Arduino Nano for further processing.

LiPo Battery (2 units):

Lithium Polymer (LiPo) batteries are lightweight and rechargeable power sources commonly used in portable electronic devices.

The LiPo batteries power the entire ECG device. Their compact form factor is well-suited for wearable applications, ensuring a sufficient power supply for continuous monitoring while maintaining portability.

Surface Electrodes:

Surface electrodes are small sensors that adhere to the skin's surface to pick up electrical signals generated by the heart.

Placed strategically on the body, the surface electrodes capture the bioelectric signals produced during the cardiac cycle. These signals are then transmitted to the AD8232 for processing.

3 Lead Wires:

Lead wires are conductive cables that connect the surface electrodes to the AD8232 or other ECG monitoring devices.

The lead wires serve as conduits for the electrical signals picked up by the surface electrodes, channelling the signals to the AD8232 module for amplification and processing.

Casing:

The casing is the protective enclosure for your ECG device, providing physical support and shielding for the internal components.

The casing ensures the safety and durability of the electronic components. It also contributes to the overall aesthetics and user-friendliness of the device, making it suitable for daily use.

By combining these components, we create a holistic system that can capture, process, and transmit real-time ECG data wirelessly. The Arduino Nano serves as the central controller, orchestrating the functions of the various modules to provide a comprehensive and portable ECG monitoring solution.

Arduino IDE:

Arduino Integrated Development Environment (IDE) is an open-source software application that provides a platform to write, compile, and upload code to Arduino microcontrollers.
In your project, the Arduino IDE is used to write and upload the firmware code to the Arduino Nano. This code governs the data acquisition from the AD8232 module, processing of ECG signals, communication with the HC-05 Bluetooth module, and other functionalities of your ECG device.

Python:

Python is a high-level programming language known for its readability and versatility. It supports a wide range of libraries and frameworks, making it a popular choice for various applications, including data processing, web development, and automation.
Python is used in your project to create a bridge between the Arduino Nano and the cloud. It likely handles the reception of ECG data from the Arduino Nano via serial communication. Python scripts can process and prepare the data for transmission to the cloud.

Serial Library (in Python):

The serial library in Python provides tools for working with serial ports. It allows communication with devices connected to the serial port, such as Arduino microcontrollers.
In your project, the serial library is likely used to establish a serial communication link between the Arduino Nano and the computer running the Python script. This enables the Python script to receive real-time ECG data from the Arduino Nano for further processing and transmission.

Gspread Library:

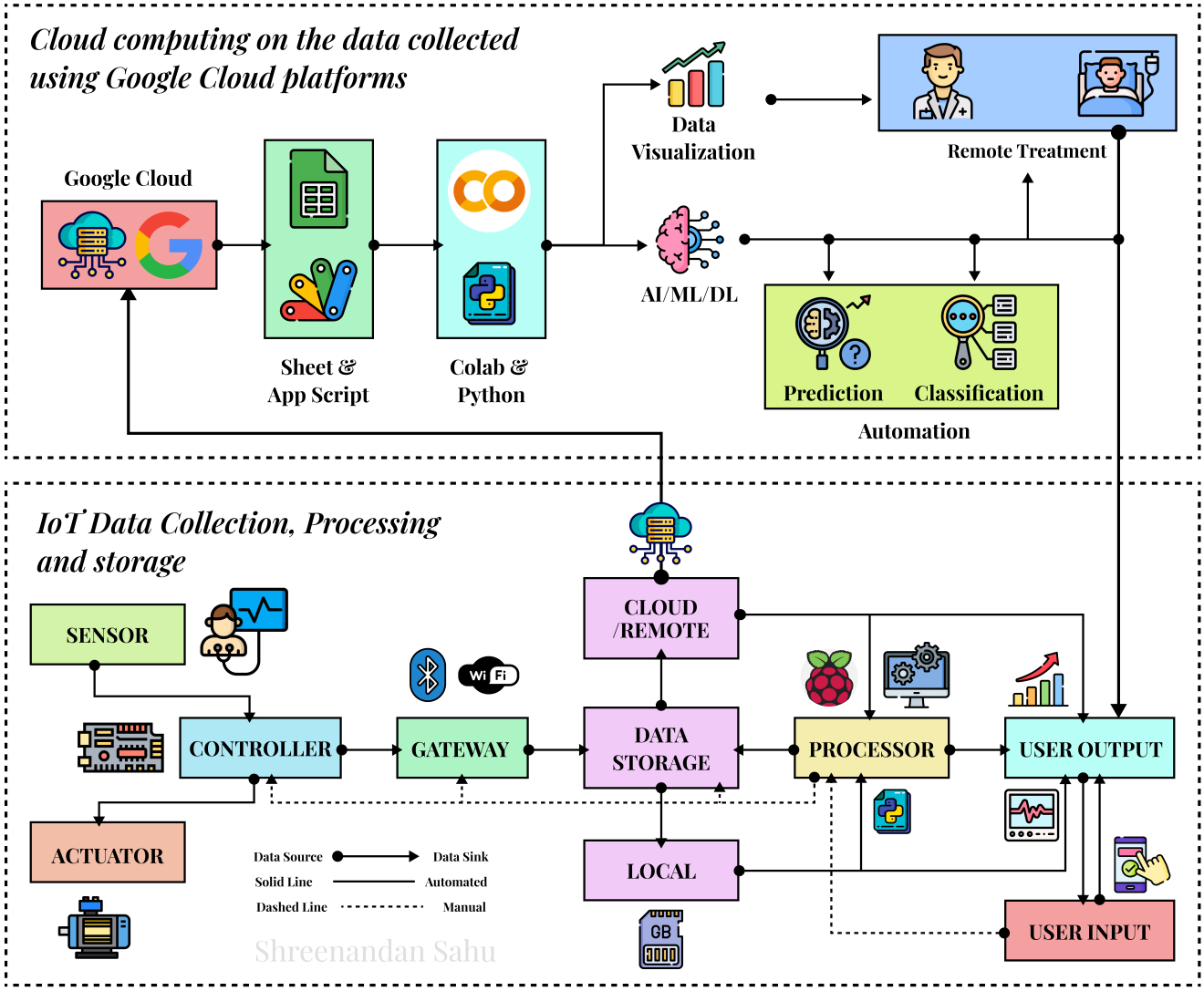
The gspread library is a Python wrapper for the Google Sheets API, allowing interaction with Google Sheets from Python scripts.
In your project, the gspread library is utilized to send ECG data to a Google Sheet. The Python script can authenticate with Google Sheets, open a specific sheet, and append or update the sheet with the incoming ECG data received from the Arduino Nano.

Google Cloud:

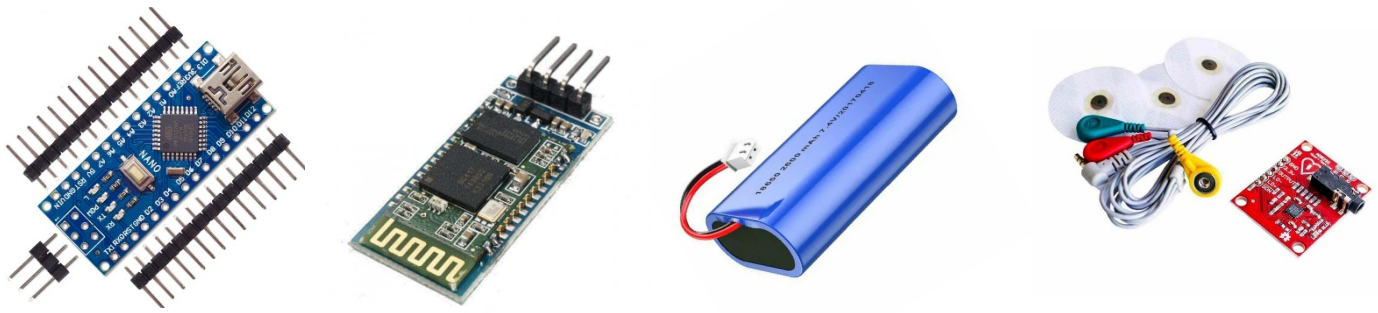
Google Cloud is a suite of cloud computing services offered by Google. It includes various services such as storage, databases, machine learning, and more.

Google Cloud may be used to store and process ECG data in the cloud. Depending on your project's requirements, you might leverage services like Google Cloud Storage for data storage or Google Cloud Platform (GCP) for more advanced processing and analytics.

Concluding the Arduino IDE is used for programming the Arduino Nano, Python scripts facilitate the communication between the Arduino Nano and the cloud using the serial and gspread libraries, and Google Cloud services provide a platform for storing and processing the ECG data in the cloud. Together, these components create a seamless flow of data from the ECG device to the cloud for real-time monitoring and analysis.



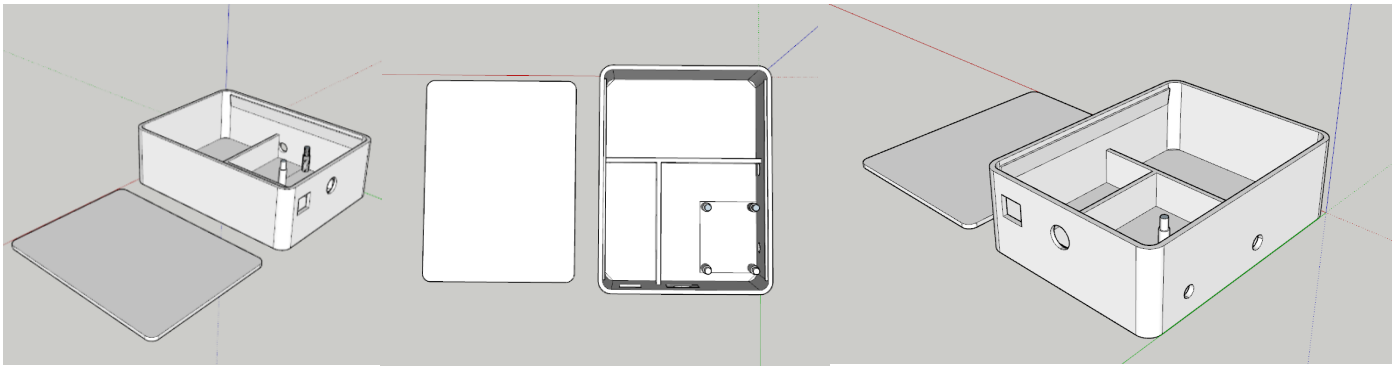
Schematic Showing the flow chart of DATA flow from the patient to the cloud Storage.



Arduino NANO | HC05 Bluetooth Module | LiPo Battery | AD8232 ECG module

Methodology

Device Casing Design



Python Code

Final Clouid sender.py | ckjsdbkvKVlvh.py | temperature_storing.py | bthr.py

Python > Final Clouid sender.py > ...

```python
import gspread as gc
import time
import serial
import serial.tools.list_ports


# Get a list of available serial ports
available_ports = list(serial.tools.list_ports.comports())

# Print information about each available port
for port in available_ports:
    gc=gc.service_account(filename='apikey.json')
    sheet=gc.open_by_key('1B5UxbJQ9euV5pNHbRVOFW1cLMwpBMbyf2WzxwmH5AWY')
    worksheet1=sheet.worksheet(title='AUGUST')
    worksheet2=sheet.worksheet(title='RECORD')
    worksheet2.clear()

    ard_data=serial.Serial(port.device,9600)
    time.sleep(1)
    #data_array=np.zeros(20)
    t=1
    t1=time.time
    while True:
        while(ard_data.inWaiting()==0):
            pass
        k=1
        data=ard_data.readline()
        data=str(data,'utf-8')
        data=data.strip('\r\n')
        data_array=data
        while(k<200):
            data=ard_data.readline()
            data=str(data,'utf-8')
            data=data.strip('\r\n')
            print(data)
            data_array=data_array+" "+data
            k=k+1
        data_array=data_array+" "
        worksheet2.update_cell(t,1,data_array)

        t=t+1
```

App Script Code

Apps Script bio-data

Deploy

Files

splitrowcolumn.gs
live graph.gs
2minecg.gs

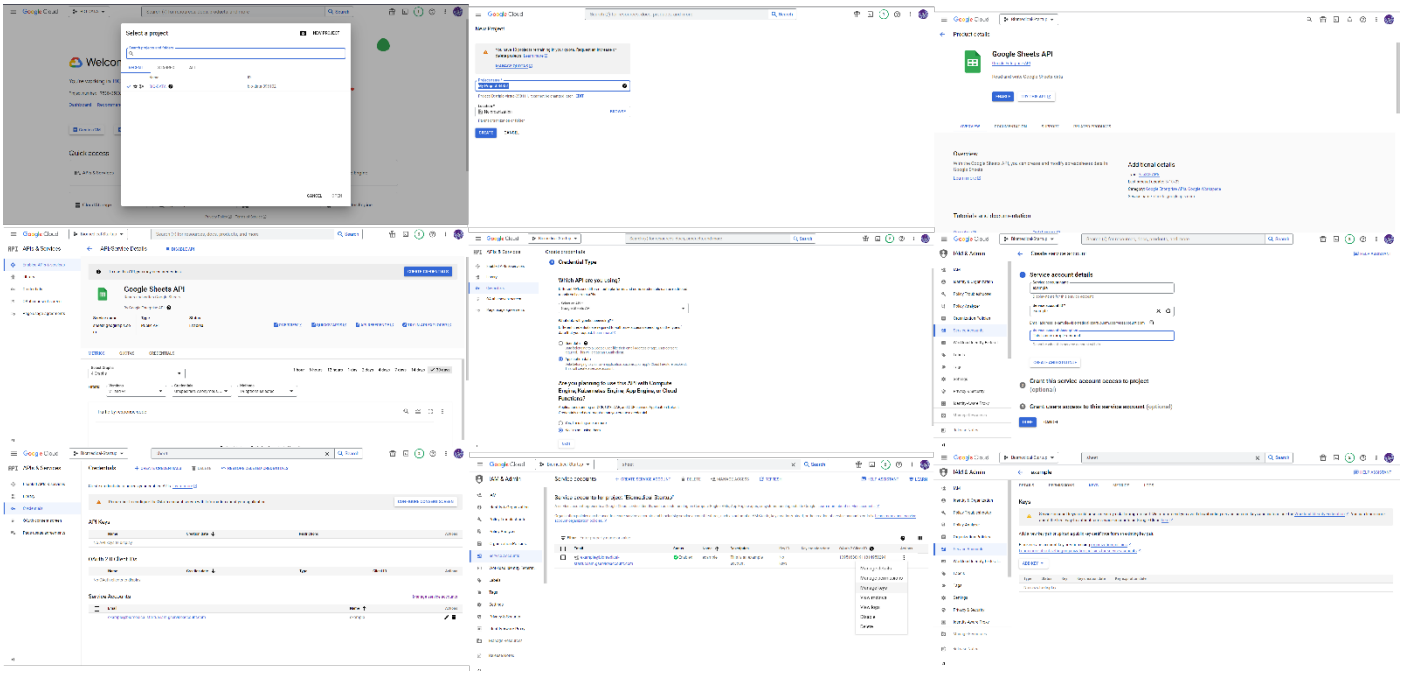
Libraries

Services

Run Debug row_split Execution log

```
function row_split() {
  var ss = SpreadsheetApp.openById('1B5UxbJQ9euV5pNHbRVOFW1cLMwpBMbyf2WzxwmH5AWY');
  var source = ss.getSheetByName('RECORD');
  var destination = ss.getSheetByName('MIN');
  var lastrow = source.getLastRow();
  if (lastrow >= 4) {
    for(let i=lastrow-3;i<=lastrow;i++){
      var value1 = source.getRange(lastrow-3,1).getValue();
      var value2 = source.getRange(lastrow-2,1).getValue();
      var value3 = source.getRange(lastrow-1,1).getValue();
      var value4 = source.getRange(lastrow,1).getValue();
      var mainvalue=value1+value2+value3+value4;
      destination.getRange(1, 1).setValue(mainvalue);
    }
    var range1 = destination.getRange(1, 1);
    range1.splitTextToColumns(' ');
    // var range2 = source.getRange(lastrow - 4, 1);
    // range2.splitTextToColumns(' ');
  }
}
```

Google Developer Console Setting

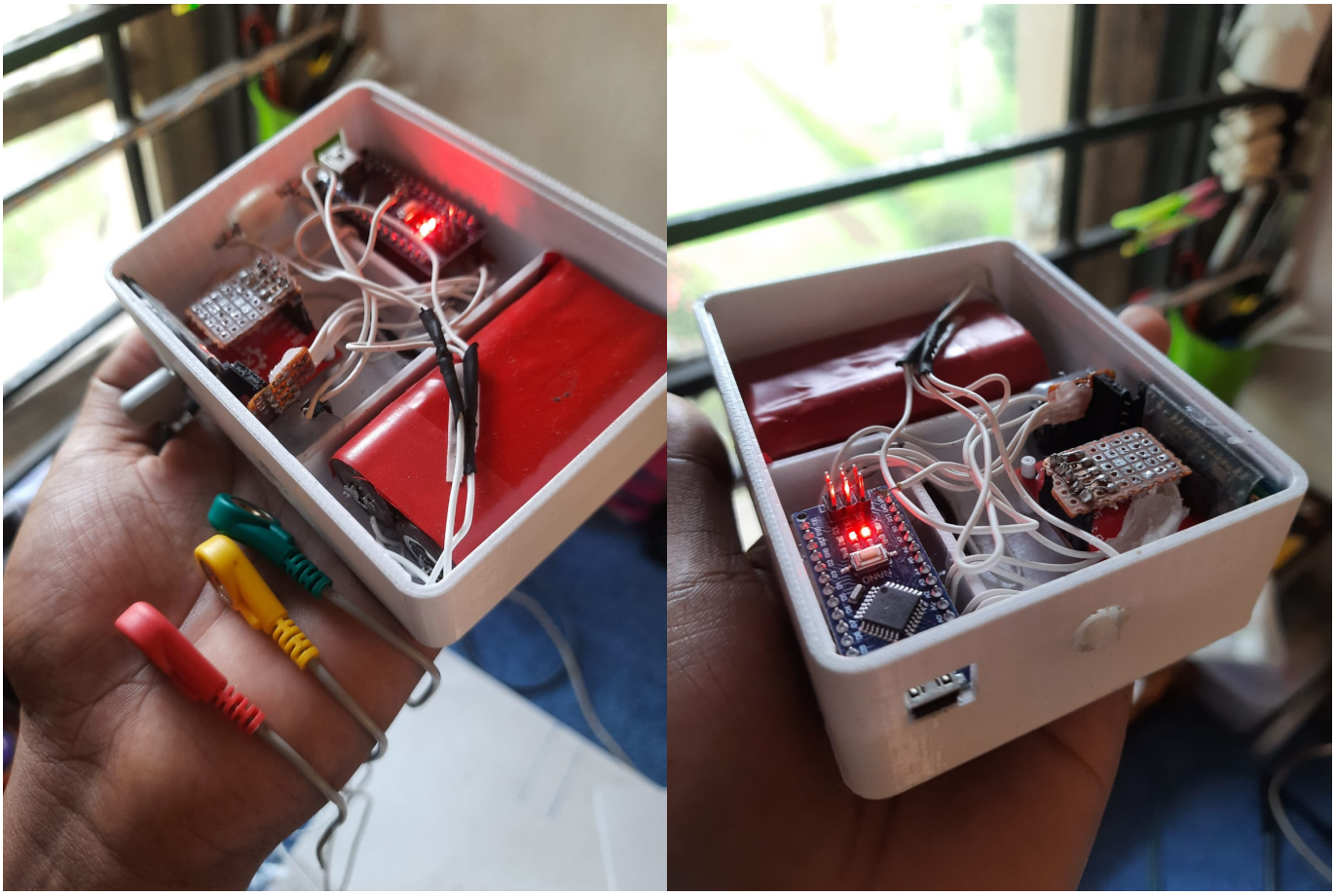


Arduino Code

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

int buzzerPin = 13;

void setup() {
  Serial.begin(9600);

  // Initialize MPU6050
  Wire.begin();
  mpu.initialize();

  // Set the buzzer pin as an output
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // Read accelerometer data
  int16_t ax, ay, az;
  mpu.getAcceleration(ax, ay, az);
```

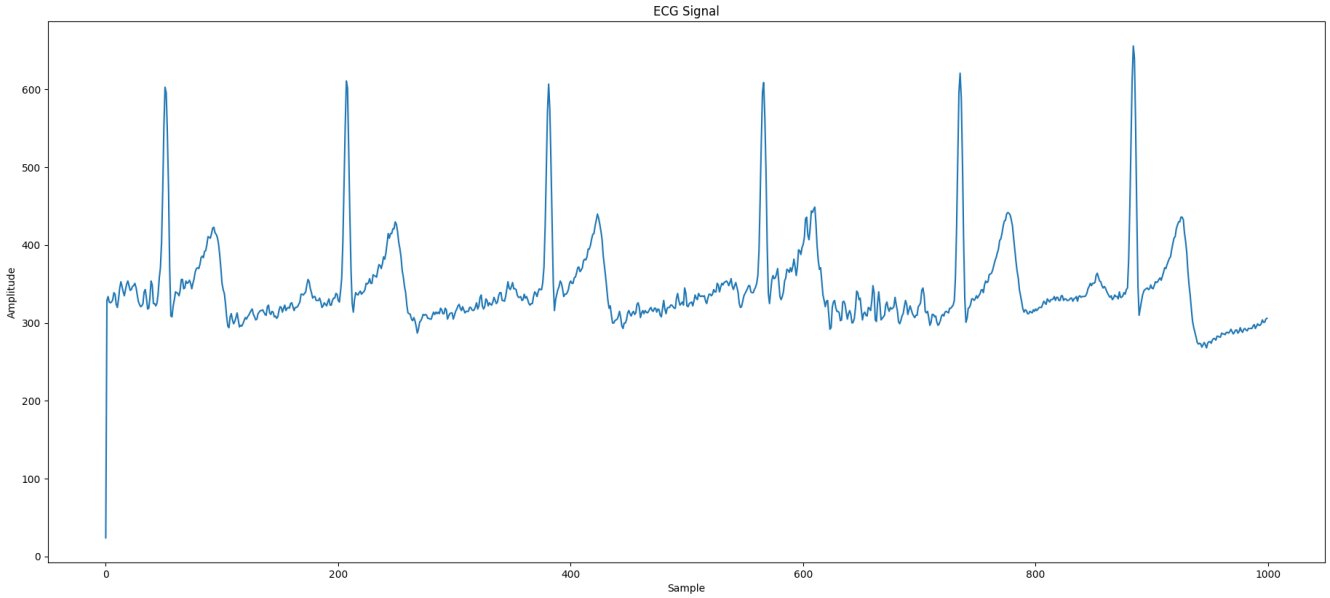
1. **Hardware Setup:**
   Assemble the hardware components, including the Arduino Nano, HC-05 Bluetooth module, AD8232 ECG recording module, LiPo batteries, surface electrodes, and lead wires.
   Connect the components according to the hardware specifications, ensuring proper power supply and signal connections.
2. **Arduino Programming:**
   Use the Arduino IDE to write firmware code for the Arduino Nano.
   Implement code to read ECG data from the AD8232 module, process the signals, and establish communication with the HC-05 Bluetooth module.
   Ensure that the Arduino Nano can reliably transmit ECG data over Bluetooth.
3. **Python Script for Serial Communication:**
   Develop a Python script using the serial library to establish serial communication with the Arduino Nano.
   Receive real-time ECG data from the Arduino Nano through the serial port.
   Implement error handling and data parsing to ensure the integrity of received data.
4. **Google Sheets Integration:**
   Use the gspread library to authenticate and connect to Google Sheets.
   Create a Google Sheet to serve as the cloud-based storage for ECG data.
   Develop Python code to append or update the Google Sheet with the incoming ECG data.
5. **Real-time Data Transmission:**
   Integrate the Python script with the Arduino firmware to enable real-time data transmission.
   Optimize data transmission protocols to minimize latency and ensure reliable communication between the device and the cloud.
6. **Data Security and Privacy:**
   Implement encryption for the serial communication between the Arduino Nano and the Python script.
   Ensure compliance with data privacy standards (e.g., HIPAA) by anonymizing or securing sensitive patient information.
7. **Power Management:**
   Optimize power consumption to maximize the device's battery life.
   Implement low-power modes during idle periods to conserve energy.
8. **User Interface and Experience:**
   Design a user-friendly interface for the ECG device, providing visual feedback and alerts for users and healthcare professionals.
   Consider usability factors such as button controls, indicator LEDs, and display screens.

Results and Final Product

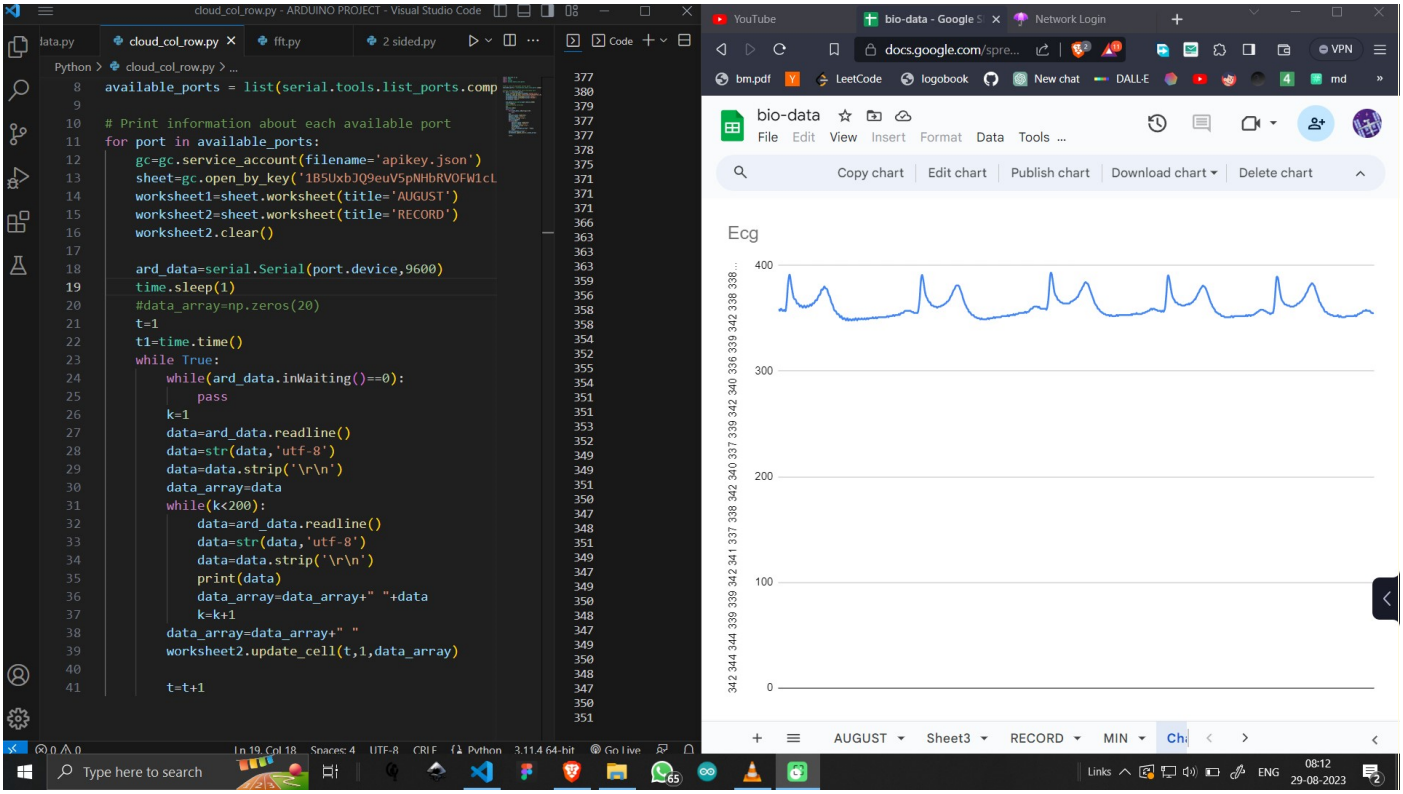


Internal Circuitry of the device with the 3 leads for ECG

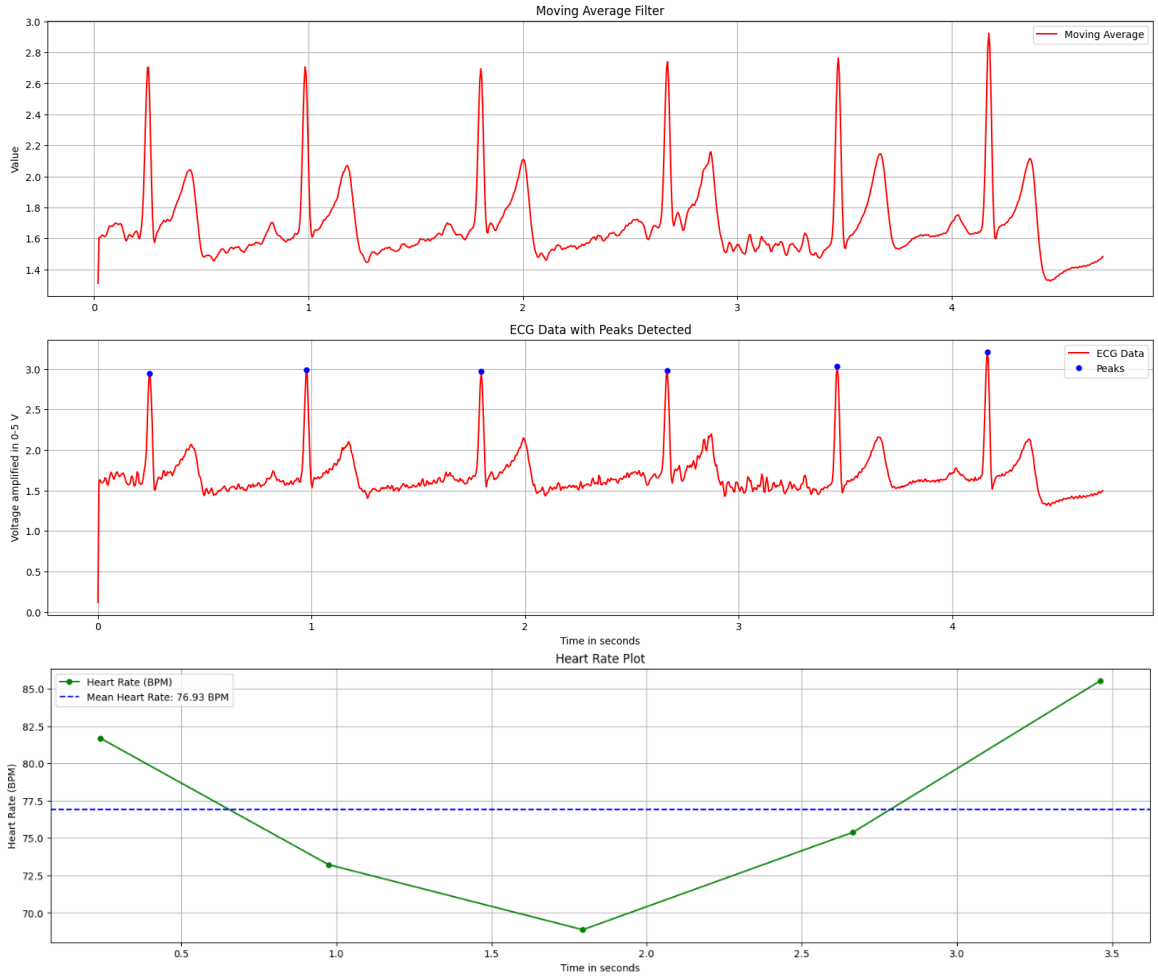
Patient using our device to monitor ECG



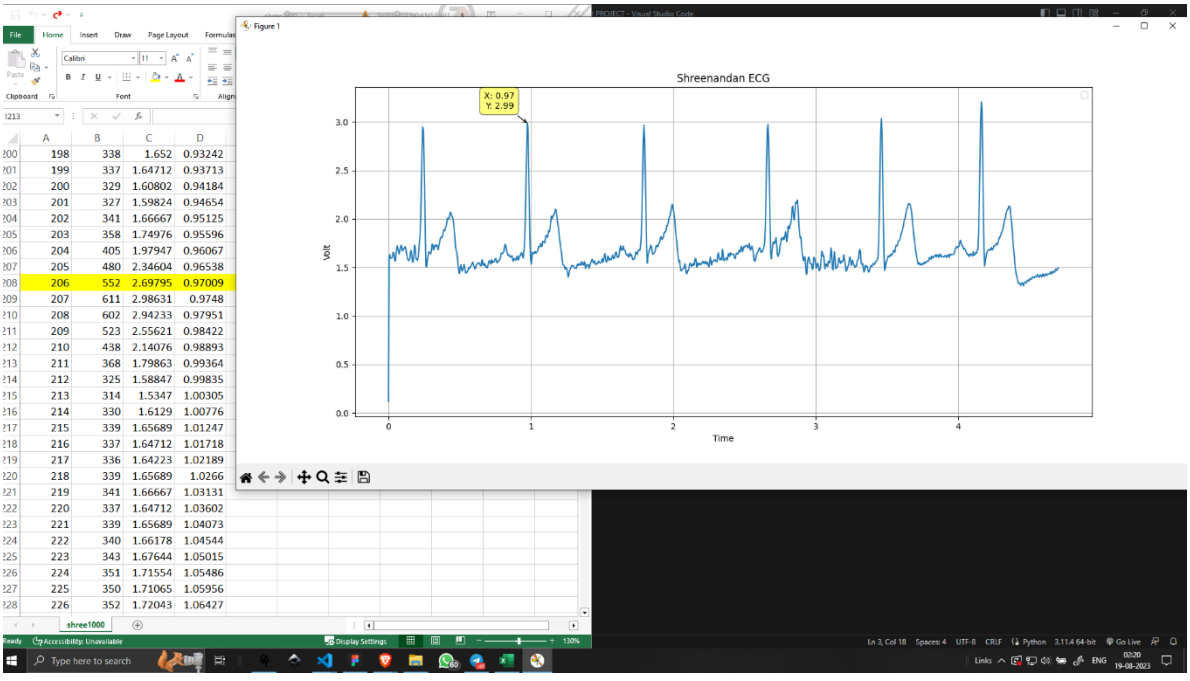
ECG plot of a patient using our device



ECG plot of a patient using our device plotted on the cloud (google sheet.)



Different plots showing ECG plot of a patient using our device and processed in google collab.



Recording and storing ECG data in CSV format for further study.

Conclusion and Future Work

Future Work

Enhanced Data Analytics:

Integrate advanced data analytics algorithms to derive more meaningful insights from ECG data. This could include trend analysis, anomaly detection, and predictive modelling for early detection of cardiac issues.

Integration with Health Platforms:

Explore integration with existing health platforms and electronic health record (EHR) systems to streamline the integration of ECG data into a patient's overall health record.

Mobile Application Development:

Develop a dedicated mobile application that connects to the IoT ECG device, providing users with a user-friendly interface for real-time monitoring, historical data review, and personalized health insights.

Machine Learning for Diagnosis:

Investigate the use of machine learning models for automated diagnosis or preliminary identification of irregularities in ECG data. This could contribute to more efficient healthcare practices.

Long-Term Monitoring Studies:

Conduct long-term monitoring studies involving a larger population to assess the device's efficacy in real-world scenarios and to gather more diverse data for analysis.

Integration with Wearables:

Explore partnerships or integrations with wearable devices to enhance user experience and provide a more comprehensive approach to health monitoring.

Customization and Personalization:

Implement features that allow users and healthcare professionals to customize alert thresholds, visualization preferences, and other settings to better suit individual needs.

Remote Firmware Updates:

Develop a mechanism for remote firmware updates to ensure that the device can be easily maintained and upgraded without requiring physical intervention.

Security and Privacy Enhancements:

Continuously assess and enhance the security measures of the device and its data transmission to meet evolving standards and address emerging cybersecurity threats.

Expand Device Compatibility:

Investigate the possibility of making the device compatible with a broader range of platforms and operating systems to maximize accessibility.

Conclusion

In conclusion, the development of an IoT-based ECG device with cloud storage represents a significant step toward personalized and connected healthcare. The integration of Arduino Nano, Python, gspread, and Google Cloud has enabled real-time monitoring and secure storage of ECG data.

The project has the potential to revolutionize cardiac monitoring by providing continuous and remote access to critical health data. Future work can further enhance the device's capabilities, including advanced analytics, machine learning for diagnosis, and integration with broader health ecosystems.

Overall, this project lays the foundation for a scalable and innovative solution to cardiac health monitoring, contributing to the broader landscape of IoT-enabled healthcare technologies. Continuous collaboration with healthcare professionals, regulatory bodies, and end-users will be crucial for refining and advancing the device in line with evolving medical standards and user needs.