IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

IOT BASED SMART ENERGY METER USING ESP8266 WITH GOOGLE SHEET INTEGRATION

M. SIVA RAMA KRISHNA*1, TANGA VENKATA NAGA SAI PRANATHI*2, NALAGARLA PENUSILA JAYANTH*3, DEEPIKA MADHURI*4, YALAKAPATI RAJA BABU*5

*1Assistant Professor, Electrical and Electronics Engineering, Bapatla Engineering College, Andhra Pradesh, India

*2,3,4,5, Students, Electrical and Electronics Engineering Department, Bapatla Engineering College, Bapatla, Andhra Pradesh, India.

Abstract: The goal of this project is to create an Internet of Things (IoT) smart energy meter utilizing the ESP8266 microcontroller platform in response to the growing demand for effective energy management and monitoring. By integrating Google Sheets, the energy monitoring system's usability and accessibility are improved through the user-friendly interface it offers for data logging and analysis.

The heart of the smart energy meter is the ESP8266 microprocessor, which gathers and processes data from sensors that measure voltage and current levels. Users are able to track their energy usage in real time thanks to these sensors, which detect energy consumption accurately.

This project's interaction with the cloud-based spreadsheet program Google Sheets is one of its main benefits. Users can access their energy usage by uploading data about their energy consumption to Google Sheets.

Keywords: IOT, ESP8266, GOOGLE SHEET INTEGRATION, CURRENT SENSOR, VOLTAGE SENSOR, RELAY.

I. INTRODUCTION

The primary purpose of the IOT-based smart energy meter is to track the energy consumption of the residential load automatically. This meter is capable of sending the consumption to the consumer as well as the electricity supplier. The objective of our project is to develop a real-time energy monitoring system using an ESP8266, Arduino Uno, 1-channel relay, ZMPT101B voltage sensor, ZMCT103C current sensor, and additional components. The system will track energy consumption data every 5 seconds, including voltage, current, power, and units consumed. This data will be transmitted to a Google spreadsheet for storage and analysis. Additionally, if the power consumption exceeds 100 watts, an alert will be triggered through a buzzer to notify users of potential energy overuse.

II. Methodology

The methodology for creating an IoT-based smart energy meter using ESP8266 with Google Sheets integration involves several key steps. First, set up the Google Sheets API by creating a new project in the Google Developers Console, enabling the Sheets API, and generating credentials. Next, assemble the hardware components, including the ESP8266 NodeMCU, current sensor, voltage sensor, and optional relay for power control. Connect these components according to the circuit diagram. Then, develop the firmware using the Arduino IDE, write code to read data from the sensors, and integrate Wi-Fi connectivity

for communication with Google Sheets. Implement code to establish a connection with the Google Sheets API and send energy consumption data for logging. Set up a Google Sheets document to store the data and configure the necessary permissions. Test the system thoroughly, ensuring accurate data logging and transmission. Finally, optimize the system for efficiency and reliability, and document the setup and deployment processes.

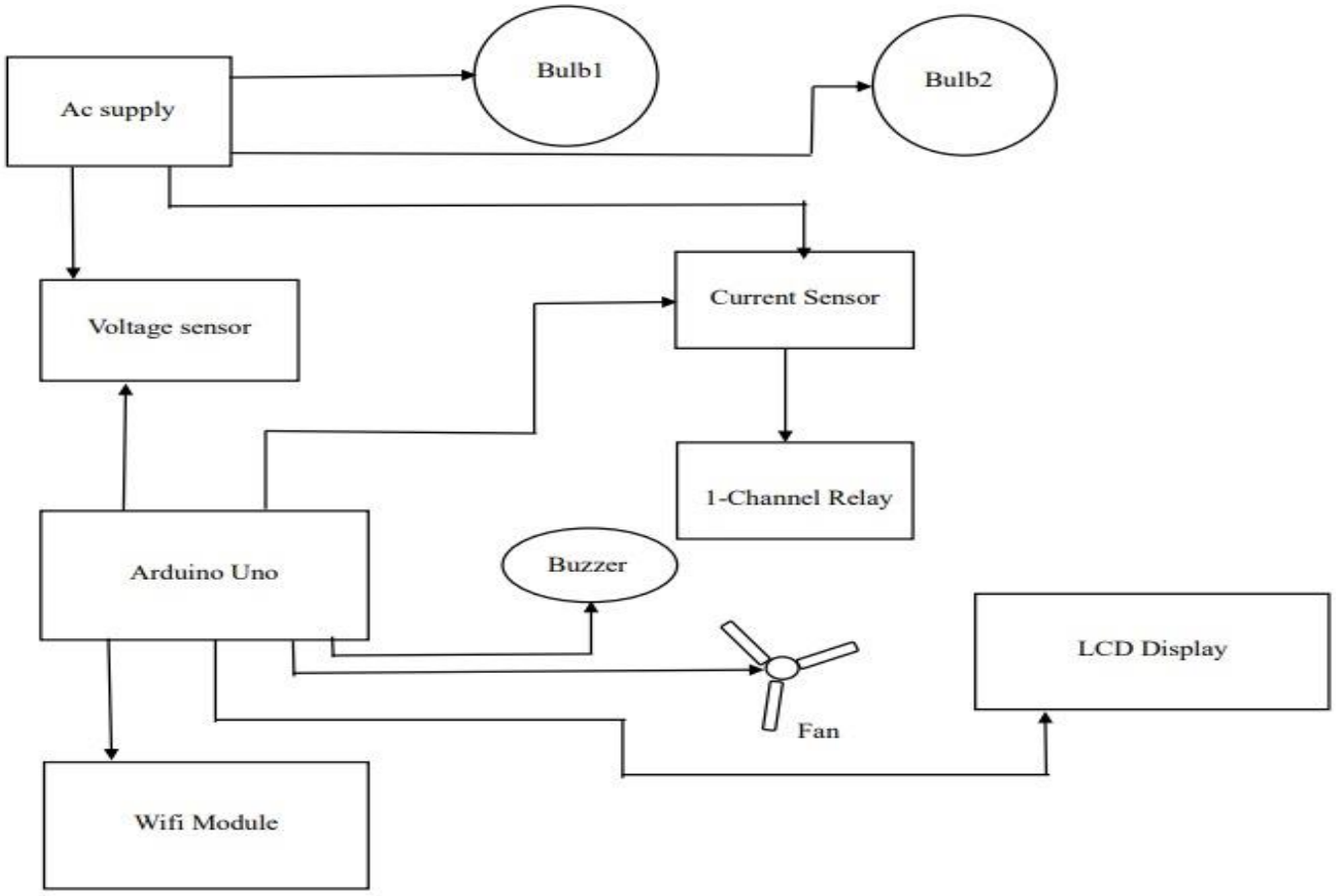

fig 1. block diagram of system

III. COMPONENTS USED

1. INTERNET OF THINGS (IOT)

The term IoT, or Internet of Things, refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves. While IoT has been in existence since the 90s, recent advances in a number of different technologies have made it more practical, such as:

- ➢ Access to affordable and reliable sensors
- ➢ Increase in the availability of cloud computing platforms
- ➢ Advances in machine learning and AI technologies

Fig2 : iot

IoT devices can range from small ordinary household cooking appliances to sophisticated industrial tools. Each IoT component has a Unique Identifier (UID) and they can also transmit data without the assistance of humans.

2. CURRENT SENSOR

Fig3 : current sensor

A current sensor is an electronic device used to measure the flow of electric current in a circuit. It works by detecting the magnetic field generated by the current passing through a conductor. This detection process is typically based on principles like the Hall Effect. Current sensors produce an output signal, usually voltage or current, that is proportional to the amount of current being measured. They are commonly used in various applications such as energy monitoring systems, motor control, and power management. In projects like smart energy meters, current sensors play a crucial role in accurately measuring the energy consumption of electrical devices, helping users track and manage their power usage effectively.

3. VOLTAGE SENSOR

Fig4 : voltage sensor

A voltage sensor is an electronic device used to measure the electrical potential difference between two points in a circuit. It works by converting the voltage into a proportional electrical signal, typically voltage or current, that can be easily measured by other electronic components. Voltage sensors are essential in various applications, including power supply monitoring, battery management, and safety systems. In projects like smart energy meters, voltage sensors play a critical role in measuring the voltage of the

electrical supply, allowing users to monitor and analyze power usage accurately. By providing insights into voltage levels, these sensors help ensure the stability and efficiency of electrical systems.

***Features**:*

1. It measures the voltage between two points in a circuit.
2. Provides accurate voltage measurements.
3. Easy to install and integrate into electrical systems.
4. Compact size for space-saving installation.
5. Wide operating voltage range for versatility.
6. Compatible with various voltage sources and systems.

4. RELAY MODULE

A relay is one kind of electro-mechanical component that functions as a switch. The relay coil is energized by DC so that contact switches can be opened or closed. A single-channel 5V relay module generally includes a coil and two contacts, normally open (NO) and normally closed (NC).

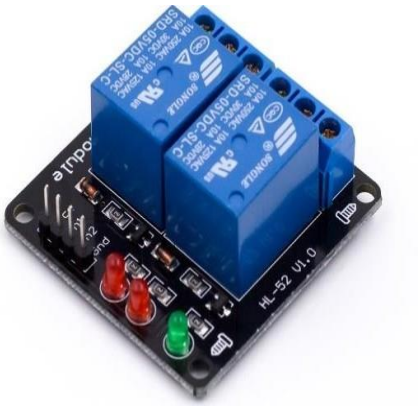

Fig5 : relay module

5V Relay Module Pin Configuration

The pin configuration of the 5V relay module is shown below. This module includes six pins, and each pin and its functionality are discussed below.

Normally Open (NO): This pin is normally open unless we provide a signal to the relay module signal pin. So, the common contact pin smashes its link through the NC pin to make a connection through the NO pin.

Common Contact: This pin is used to connect to the load that we desire to switch by using the module.
Normally Closed (NC): This NC pin is connected through the COM pin to form a closed circuit. However, this NC connection will break once the relay is switched, providing an active high/low signal toward the signal pin from a microcontroller.

Signal Pin: The signal pin is mainly used for controlling the relay. This pin works in two cases: active low and active high. So, in the active low case, the relay activates once we provide an active low signal toward the signal pin, whereas in the active high case, the relay will trigger once we provide a high signal toward the signal pin.

5V VCC: This pin needs 5V DC to work. So 5V DC power supply is provided to this pin.
Ground: This pin connects the GND terminal of the power supply.

IV. NODE MCU ESP8266

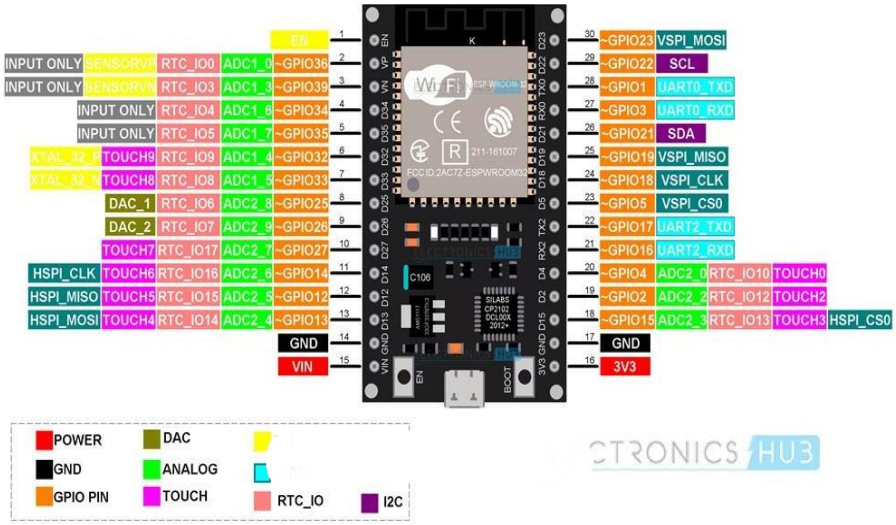

Fig6 : node mcu

ESP32 is a low-cost, low-power Microcontroller with an integrated Wi-Fi and Bluetooth. It is the successor to the ESP8266 which is also a low-cost Wi-Fi microchip albeit with limited vastly limited functionality.

It is an integrated antenna and RF balun, power amplifier, low-noise amplifiers, filters, and power management module. The entire solution takes up the least amount of printed circuit board area. This board is used with 2.4 GHz dual-mode Wi-Fi and Bluetooth chips by TSMC 40nm low power technology, power and RF properties best,which is safe, reliable, and scale-able to a variety of applications.

- Power Pins There are four power pins. VIN pin and three “3.3V” pins.
  - VIN can be used to directly supply the Node MCU/ESP32 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the Node MCU module – you can also supply 5V regulated to the VIN pin
  - “3.3V” pins are the output of the onboard voltage regulator and can be used to supply power to external components.
- GND are the ground pins of Node MCU/ESP32.
  - I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
  - GPIO Pins Node MCU/ESP32 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
  - SPI Pins Node MCU/ESP32 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80 MHz
- Up to 64-Byte FIFO
  - UART Pins Node MCU/ESP32 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to “4.5” Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin)

features only data transmit signal so, it is usually used for printing log.

- PWM Pins The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).

- Analog: Used to send/receive analog data using the following functions:
  # examples based on Arduino IDE analogRead();

  analogWrite();

  Digital: Used to send/receive digital data using the following functions:
  # examples based on Arduino IDE digitalRead();

  digitalWrite();

- DAC : ESP32 has two 8-bit DAC (digital to analog converter) channels internally which are connected to GPIO25 (Channel 1) and GPIO26 (Channel 2). 8-bit DAC means, ESP32 can convert the digital input (0 to 255) to equivalent analog output.

RTC timer: This timer allows time keeping in various sleep modes, and can also persist time keeping across any resets (with the exception of power-on resets which reset the RTC timer).

- Control Pins are used to control the Node MCU/ESP32 . These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin. EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

- RST: RST pin is used to reset the ESP8266 chip.
- WAKE: Wake pin is used to wake the chip from deep-sleep.

V. GOOGLE SPREAD SHEET INTEGRATION

Integrating Google Sheets into a project allows for seamless data logging and analysis. To achieve this integration, follow these simplified

steps:

1. **Set up the Google Sheets API**: Begin by creating a new project in the Google Developers Console. Enable the Google Sheets API for your project and generate credentials (the service account key), downloading the corresponding JSON file containing your API credentials.
2. **Share a Google Sheets document:** Create a new Google Sheets document or use an existing one. Ensure to share the document with the email address provided in the JSON file (the service account email) to grant access.
3. **Install Required Libraries**: Utilize libraries like `gspread` in Python to facilitate interaction with Google Sheets. Install the necessary library using pip (`pip install gspread`).
4. **Authenticate with Google Sheets API**: Authenticate your application using the JSON file containing your API credentials. This step establishes a secure connection between your project and Google Sheets.
5. **Access Google Sheets Data**: Open the Google Sheets document by its URL or ID. Access specific sheets within the document using their titles or indices.
6. **Read or Write Data**: Retrieve data from the spreadsheet by accessing specific cells or ranges. Update cell values or append new rows to write data to the spreadsheet.

Following these steps enables your project to interact with Google Sheets, facilitating tasks such as data logging, analysis, and visualization. Adjustments may be necessary based on your project's requirements and programming language.

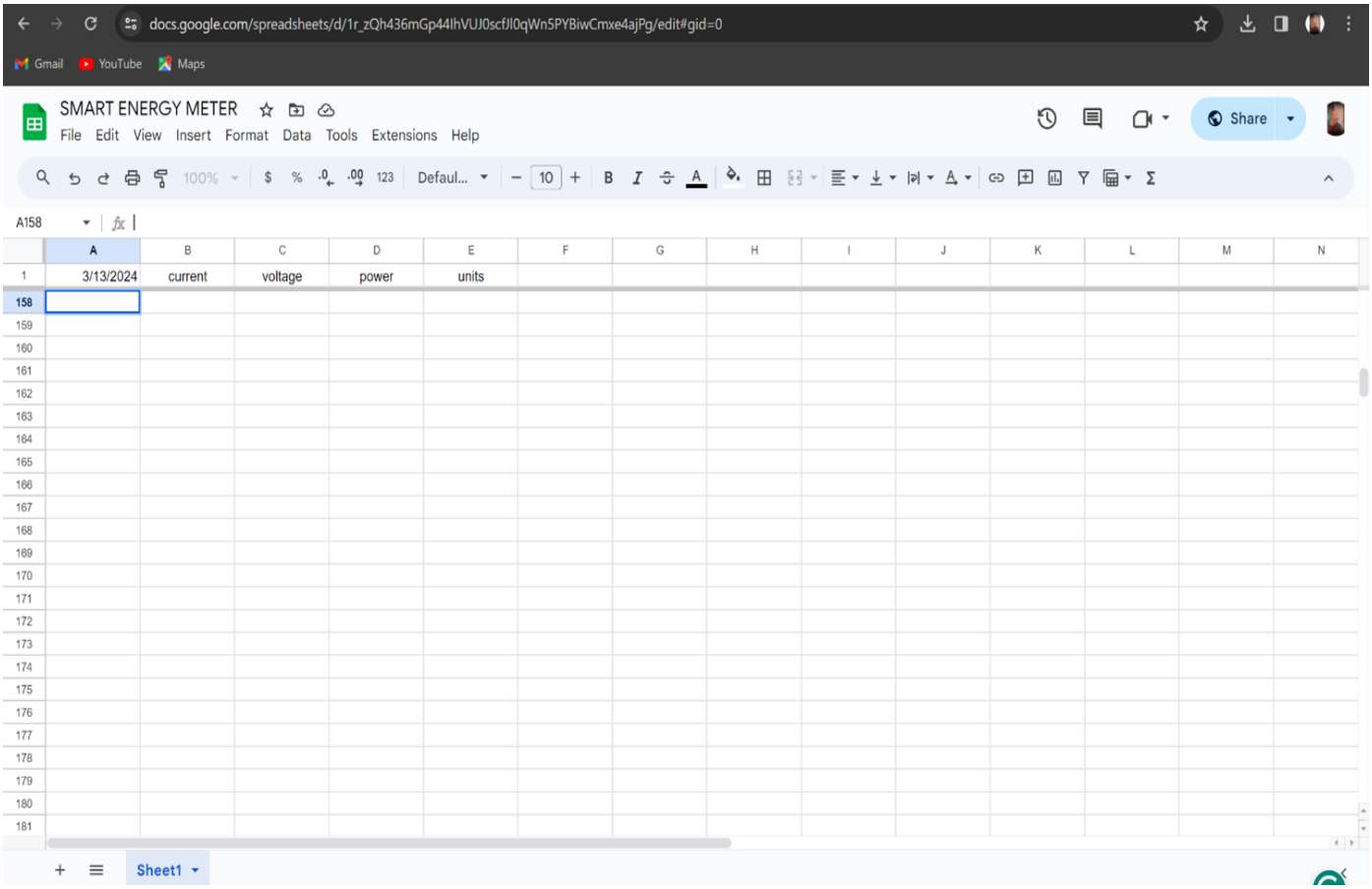

Fig7 : google spreadsheet

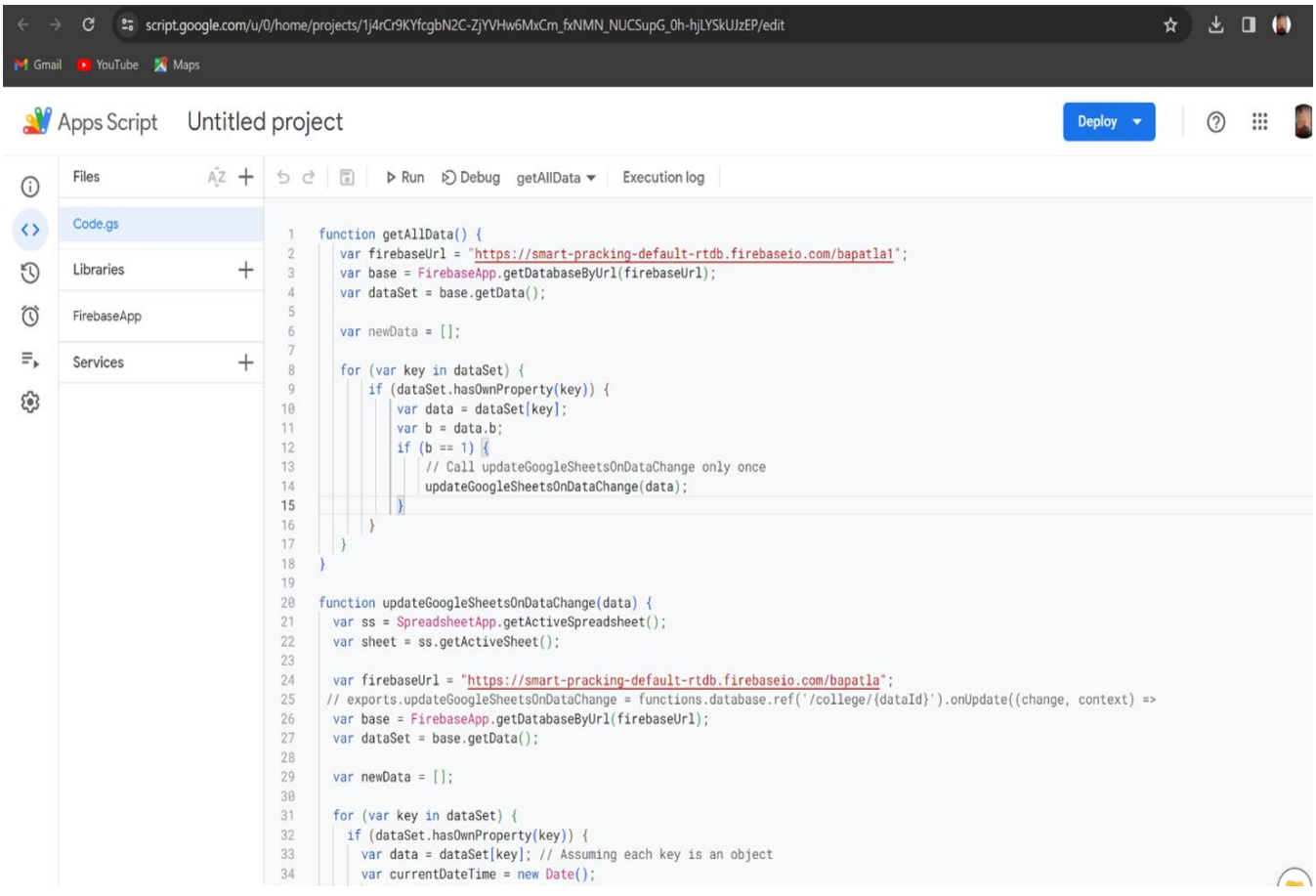

Fig8 : google apps script

VI. FIREBASE REALTIME DATABASE INTEGRATION

Firebase can serve as a robust intermediary between the WiFi module and Google Sheets, offering additional functionalities beyond simple data transmission. With the Firebase Realtime Database, data can be synchronized instantly across connected devices and platforms, ensuring real-time updates and collaboration. Additionally, Firebase Authentication can be implemented to secure access to the database, allowing only authorized users or devices to interact with the data.

Furthermore, Firebase Cloud Functions can be employed to automate the process of transferring data from the Firebase database to Google Sheets. By triggering a cloud function whenever new data is added to the Firebase database, the energy consumption data can be seamlessly pushed to a designated Google Sheets document, eliminating the need for manual intervention and ensuring data accuracy and reliability.

Moreover, Firebase offers a range of analytics and monitoring tools that can be leveraged to gain insights into energy usage patterns and trends. By analyzing the data stored in Firebase, stakeholders can make informed decisions regarding energy optimization and efficiency improvements.

Overall, integrating Firebase into the project enhances the scalability, security, and efficiency of the solution, providing a seamless bridge between the WiFi module and Google Sheets for effective energy monitoring and management.

VII. SOURCE CODE FOR ARDUINO

```
#include <LiquidCrystal.h>

const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);



long int prv=0;
int cs=A0;
int vs=A1;
float cval,cmax,vval,vmax,rpm=0,tempC;
int cnt=0;
int rly=A4;
float pwr=0;
int vl=0;
float unts=0;
int fan=2;
int led=3;
int buz=4;
void setup() {

Serial.begin(9600);
delay(2000);
lcd.begin(16, 2);
lcd.print("   WELCOME");
pinMode(rly,OUTPUT);
pinMode(fan,OUTPUT);
```

```
pinMode(led,OUTPUT);
pinMode(buz,OUTPUT);
digitalWrite(rly,0);
digitalWrite(fan,0);
digitalWrite(led,0);
delay(2000);

}

void loop() {
  // put your main code here, to run repeatedly:

long int ss=millis();
cmax=0;
vmax=0;
while(millis()-ss<5000)
{
  cval=analogRead(cs);
  vval=analogRead(vs);
  if(vval>vmax)
  vmax=vval;
  if(cval>cmax)
  cmax=cval;
}
cmax=cmax/1000;
//if(cmax<20)
//cmax=0;
vmax=(vmax-520)*1.2;
if(vmax<20)
vmax=0;
pwr=(cmax*vmax);
 unts=unts+(pwr)/1000;
lcd.clear();
lcd.print("I:"+String(cmax,1) + " V:"+String(vmax,1));
//+ " L:"+String(1-digitalRead(rly))
lcd.setCursor(0,1);
lcd.print("P:"+String(pwr,1)+ " U:"+ String(unts,1));
Serial.print(String(vmax)+","+String(cmax)+","+String(pwr)+","+String(unts)+",\n");

if(pwr>150)
{
  vl=vl+1;
  if(vl>3)
  {
    digitalWrite(buz,1);
```

```
      digitalWrite(rly,1);
      lcd.clear();
      lcd.print("OVER LOAD");
      while(1);
     }
    }
    else
    vl=0;
    cnt=cnt+1;
    if(cnt>15)
    {
     cnt=0;

    }
    }
```

VIII. SOURCE CODE FOR ESP8266

```
    #include "FirebaseESP8266.h"  // Install Firebase ESP8266 library
    #include <ESP8266WiFi.h>
    #include <SoftwareSerial.h>
    SoftwareSerial mySerial(D2, D3);
    #define WIFI_SSID "project"  //WiFi SSID
    #define WIFI_PASSWORD "12345678"  //WiFi Password
    #define FIREBASE_HOST "smart-pracking-default-rtdb.firebaseio.com"         //Firebase Project URL
Remove "https:" , "\" and "/"
    #define FIREBASE_AUTH "pysUR0viqcOWKhzRPeJeGByt0krgXgsoYfCUHA4D"

    //Define FirebaseESP8266 data object
    FirebaseData firebaseData;
    FirebaseData ledData;
    FirebaseJson json;
    String prv="0";
    int statusCode = 0;
    String strs[8]={"0","0","0","0","0","0","0","0"};
    int StringCount = 0;
    int LEDPIN_1;
    void setup()
    {
     pinMode(LEDPIN_1,OUTPUT);
     Serial.begin(9600);
      mySerial.begin(9600);
```

```
  delay(2000);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
   Serial.print(".");
   delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);

}

void loop()
{



while (Serial.available())
{
   String rcv = Serial.readStringUntil('\n');
   StringCount=0;
  // Serial.println(rcv);
    while (rcv.length() > 0)
    {
    int index = rcv.indexOf(',');
    if (index == -1) // No space found
    {
      strs[StringCount++] = rcv;
      break;
    }
    else
    {
      strs[StringCount++] = rcv.substring(0, index);
      rcv = rcv.substring(index+1);
    }
  }



 Firebase.setString(firebaseData, "/hari/ sensor1", (strs[0]));
 Firebase.setString(firebaseData, "/hari/ sensor2", (strs[1]));
```

```
// Firebase.setString(firebaseData, "/sensors/ sensor3", (strs[2]));
// Firebase.setString(firebaseData, "/sensors/ sensor4", (strs[3]));
//Firebase.setString(firebaseData, "/FirebaseIOT/VIT_BAT_GY", (strs[4]));
// Firebase.setString(firebaseData, "/FirebaseIOT/VIT_BAT_GZ", (strs[5]));
// Firebase.setString(firebaseData, "/FirebaseIOT/VIT_BAT_FL", (strs[6]));
// Firebase.setString(firebaseData, "/FirebaseIOT/VIT_BAT_FR", (strs[7]));




 }
  if (Firebase.getString(ledData,"/hari/ sensor3")){
   if(ledData.stringData() != prv)
   {
      Serial.print(ledData.stringData());
      prv=ledData.stringData();
    }
}
```

IX. SOURCE CODE FOR GOOGLE APPS SCRIPT

```
function getAllData() {
  var firebaseUrl = "https://smart-pracking-default-rtdb.firebaseio.com/hari";
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl);
  var dataSet = base.getData();

  var newData = [];

  for (var key in dataSet) {
    if (dataSet.hasOwnProperty(key)) {
      var data = dataSet[key];
      var ser = data.ser;
      if (ser == 1) {
        // Call updateGoogleSheetsOnDataChange only once
        updateGoogleSheetsOnDataChange(data);
      }
    }
  }
}

function updateGoogleSheetsOnDataChange(data) {
 var ss = SpreadsheetApp.getActiveSpreadsheet();
 var sheet = ss.getActiveSheet();
```

```
  var firebaseUrl = "https://smart-pracking-default-rtdb.firebaseio.com/gpt1";
 // exports.updateGoogleSheetsOnDataChange = functions.database.ref('/college/{dataId}').onUpdate((change, context) =>
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl);
  var dataSet = base.getData();

  var newData = [];

  for (var key in dataSet) {
   if (dataSet.hasOwnProperty(key)) {
    var data = dataSet[key]; // Assuming each key is an object
    var currentDateTime = new Date();
     var name = data.name;
     var id  = data.id;
     var branch = data.branch;
     var sec = data.sec;
     var year = data.year;
      var state = data.state;

      var newData = [[currentDateTime, name, id, branch, sec, year, state]];
      writeDataToFirebase();

   }
  }

  var lastRow = sheet.getLastRow();
  var lastColumn = sheet.getLastColumn();

  var newRange = sheet.getRange(lastRow + 1, 1, newData.length, newData[0].length);
  newRange.setValues(newData);
 }
 function writeDataToFirebase() {
  var firebaseUrl = "https://smart-pracking-default-rtdb.firebaseio.com/";
  var secret = "pysUR0viqcOWKhzRPeJeGByt0krgXgsoYfCUHA4D"; // Replace with your Firebase secret key
  var base = FirebaseApp.getDatabaseByUrl(firebaseUrl, secret);

  // Define the data to be sent
  var dataToImport = {
   ser: 2
  };

  // Specify the path where you want to store the data
  var path = "/hari/sensor6";
```

```
  // Send the data to Firebase
  base.setData(path, dataToImport);
}
```

X. WORKING

The IoT-based smart energy meter utilizing ESP8266 with Google Sheets integration functions through a series of cohesive steps. Initially, the ESP8266 microcontroller gathers data from current and voltage sensors, which monitor energy consumption within the electrical circuit. This raw data undergoes processing within the ESP8266, where analog signals are converted to digital format and algorithms calculate power consumption. Subsequently, the ESP8266 connects to a local Wi-Fi network, facilitating communication with external services. Leveraging the Google Sheets API, the ESP8266 establishes a link with a designated Google Sheets document, where processed energy consumption data is promptly transmitted and logged into specified cells or rows. Users can conveniently access this document online to monitor real-time energy consumption data and analyze historical trends, empowering them to make informed decisions regarding energy management. Additionally, the system can be configured to trigger alerts or notifications if energy consumption surpasses predefined thresholds, ensuring proactive management. Optionally, a user interface may be developed to offer graphical representations of energy consumption data, enhancing user accessibility and comprehension. Through this streamlined process, the IoT-based smart energy meter seamlessly tracks and analyzes energy usage, promoting efficient resource management and sustainability.

XI. ADVANTAGES

1. It lets you monitor energy usage in real-time.
2. You can easily store and manage energy consumption data.
3. Access your data from anywhere with the internet.
4. Analyze historical data to make informed decisions.
5. Automate data transfer to save time and reduce errors.
6. Expand functionality by adding more sensors or devices.
7. It's affordable and uses widely available components.
8. Compatible with other IoT devices and platforms.
9. It helps optimize energy usage for efficiency.
10. Contributes to a greener environment by reducing waste.

XII. CONCLUSION

In conclusion, the project "IoT-based Smart Energy Meter Using ESP8266 with Google Sheets Integration" offers a practical and efficient solution for monitoring and managing energy consumption. By leveraging IoT technology, ESP8266 microcontrollers, and Google Sheets integration, the project enables real-time monitoring, convenient data logging, and remote accessibility of energy consumption data. This facilitates informed decision-making, enhances energy efficiency, and contributes to environmental sustainability. With its scalability, cost-effectiveness, and compatibility with existing systems, the project presents a valuable tool for individuals and organizations seeking to optimize energy usage and reduce costs. Overall, the project demonstrates the potential of IoT applications in addressing real-world challenges and promoting efficient resource management.

XIII. REFERENCES

[1] Somchai Thepphaeng; Chaiyod Pirak. Design and Implementation of Wireless Sensor Network and Protocol for Smart Energy Meter. 2011 International Conference on Circuits, System and Simulation IPCSIT vol.7 (2011) © (2011)IACSIT Press, Singapore.

[2] Chih-Yung Chang, Chin-Hwa Kuo,Jian-Cheng Chen and Tzu-Chia Wang Design and Implementation of an IoT Access Point for Smart Home. Applied Science; ISSN 2076-3417. www.mdpi.com/journal/applsci

[3] Su, J.H.; Lee, C.S.; Wu, W.C. The Design and Implementation of a Low-Cost and Programmable Home Automation Module. IEEE Trans. Consum. Electron. 2006, 52, 1239–1244.

[4] S. Metering, S. Visalatchi and K. K. Sandeep, "Smart energy metering and power theft control using arduino & GSM," 2nd International Conference for Convergence in Technology (I2CT), Mumbai, 2017, pp. 858-961**.**

[5] Mr. Rajesh Kumar, D. Modi, Mr. Rajesh Sukhadi, "A Analysis on IOT Based Smart Electricity Meter ", International Paper For Technical Research In Engineering, Vol. 2, Issue 3, 2016.

[6] Md Redwanul Islam, Supriya Sarker, Md Shahraduan Mazumder, Mehnaj Rahman Ranim, "An IoT-based Realtime Low-Cost Smart Energy Meter Monitoring System using Android Application", International Journal of Engineering and Techniques - Volume 5 Issue 3, June 2019.