Energy Consumption Monitor

USN: 1RV22CS098

E-mail ID: lekhanaa.cs22@rvce.edu.in

Phone number: +91 9742689613

Manaswini Simhadri Kavali

USN: 1RV22CS106

E-mail ID: manaswinisk.cs22@rvce.edu.in

Phone number: +91 9900945811

Khushi Gupta

USN: 1RV22CS084

E-mail ID: khushigupta.cs22@rvce.edu.in

Phone number: +91 9480005846

Department of Computer Science Engineering, R.V. College of Engineering. Bengaluru-560059, India

ABSTRACT

In a world grappling with increasing energy demands and environmental concerns, the project's significance becomes apparent. As energy consumption rises across various sectors, there is a growing need for effective energy management solutions. This project, encompassing real-time power monitoring, consumption prediction, and automated alerting, directly addresses this need. By allowing users to monitor their electricity usage patterns, foresee potential consumption trends, and establish customized usage thresholds, the project tackles the essential task of curbing wasteful energy practices. The inclusion of timely telegram alerts further amplifies its relevance, empowering users to promptly address irregularities and adopt energy-efficient habits. Ultimately, this project harmonizes with sustainability objectives and cost-conscious initiatives, contributing to responsible energy consumption behaviours.

Keywords: Energy consumption, Predictive analysis, Automated alerts, Visualization, Power monitoring

TABLE OF CONTENTS

Abstract

Literature survey

1. Introduction
2. Problem statement
3. Objective
4. Components used
5. Snapshots of the model
6. Schematic Diagram

7. SENSORS USED AND THEIR WORKING

7.1 Sensors

7.1.1 SCT013 30 A Current Sensor

7.1.2 ZMPT101B Voltage Sensor

7.2 Voltage divider rule

7.2.1 Scaling factor for Current Sensor

7.2.2 Scaling factor for Voltage Sensor

8. DATA ANALYSIS

8.1 Arduino Code to read current and voltage values

8.2 Dataset generation, cleaning and pre-processing

8.3 Visualization

8.5 Automated alerts

8.6 Webpage to display visualized data

9. Future scope in the area

10. Conclusion

11. Acknowledgement

12. References

LITERATURE SURVEY

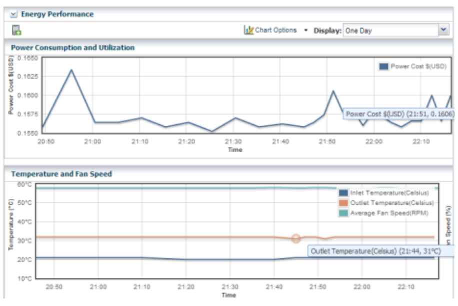
AUTHORS	PAPER TITLE	PAPER TITLE	SUMMARY	REMARKS
Sreevidhya C, Mukesh Kumar, Ilango K	Design and Implementation of Non-Intrusive Load using Machine Learning Algorithm for Appliance Monitoring	Published in: 2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)	Paper proposes Non-intrusive Load Monitoring (NILM) technique for energy conservation. It aims to monitor individual appliances with a single energy meter, reducing costs. MATLAB/Simulink used for modeling and classification.	Paper proposes Non-intrusive Load Monitoring (NILM) technique for energy conservation. It aims to monitor individual appliances with a single energy meter, reducing costs. MATLAB/Simulink used for modeling and classification.
Nurul Hafiqah Azmi, Nor Adni Mat Leh, Nur Atharah Kamaruzaman	Modeling of Energy Meter Using MATLAB or Simulink	Published in: 2018 9th IEEE Control and System Graduate Research Colloquium (ICSGRC)	This paper simulates an energy meter using MATLAB/Simulink to measure electric parameters in three-phase systems. Various loads are studied, confirming the meter's high accuracy and efficiency.	**1.**There is no discussion of the UI, ease of use, or UX aspects of the energy meter system. **2.**The paper mentions other design software used for energy metering but doesn't provide a detailed comparison with the proposed MATLAB/Simulink-based solution. **3.**The paper primarily focuses on individual load scenarios. It could be beneficial to discuss the system's scalability to handle multiple loads and its performance under varying conditions, such as load fluctuations and transient events.

Gitanjali Mehta; Ruqaiya Khanam;Vinod Kumar Yadav	A Novel IoT based Smart Energy Meter for Residential Energy Management in Smart Grid Infrastructure	Published in: 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN)	The energy industry faces challenges due to increased energy consumption & renewable energy generation. Smart electricity meters are essential components of the future smart grid, measuring energy flow, exchanging consumption info ,& monitoring home appliances & devices.	**1.**This technique only gives the total energy consumption without identifying individual appliances. **2.**The accuracy of the NILM technique relies on the quality of the data acquired from the sensors. Any noise, fluctuations, or inaccuracies in the data can impact the overall performance of appliance identification algorithm **3.**User Behavior Impact: Behavioral changes are crucial for achieving meaningful energy savings.
Hussain Shareef, Maytham S. Ahmed, Azah Mohamed, Eslam Al Hassan	Review on Home Energy Management System Considering Demand Responses, Smart Technologies, and Intelligent Controllers	Published in: IEEE Access (Volume: 6) Date of Publication: 30 April 2018	The rise of smart grids &increasing electricity demand have led to the development of HEMS that use demand response tools to improve energy consumption .This paper reviews research on DR programs, smart technologies, load scheduling controllers, & heuristic optimization techniques for optimal scheduling of electrical devices in smart homes.	1. **Comprehensive Approach:** The review takes a holistic view, covering technology, demand-side dynamics, and control strategies. 2. **Demand Response Focus:** It highlights the importance of consumers actively participating in energy management through demand response. 3. **Smart Technology Emphasis:** The review underscores the role of smart technologies and IoT devices in real-time data collection and analysis. 4. **Intelligent Controllers:** It discusses how advanced algorithms and machine learning improve energy efficiency through intelligent controllers.

1. INTRODUCTION

Power consumption has been on the rise of late, and there is an utmost need to monitor and keep a check on the consumption at an individual scale. Managing power consumption at the individual scale can eventually add up and help in the overall reduction in the consumption of power.

After looking through multiple articles about the rise in power consumption, we have made an attempt to help monitor power consumption in an individual household. We have also integrated a statistical model called the ARIMA (AutoRegressive Integrated Moving Averages) model to predict the power consumption, to allow the user to keep better track of their household power consumption.



2. PROBLEM STATEMENT

Overconsumption of power is a major concern, given the sources of energy that we are currently depending on. We currently mainly depend on non-renewable resources of energy, which can take millions of years to replenish upon depletion. We must be watchful and aware of the amount of energy we consume on a daily basis.
The major problems we are addressing in our project are-

1. **Lack of Reliable Energy Consumption Monitoring:** In our interconnected and energy-dependent world, there is a significant absence of reliable energy consumption monitoring systems, hindering the ability of individuals and industries to accurately track and manage their energy usage.
2. **Limited Informed Decision-Making for Environmental Impact Reduction:** This absence creates a barrier to informed decision-making for individuals and industries, preventing them from taking meaningful actions to reduce their environmental impact through more efficient energy use.
3. **Insufficient Promotion of Energy Efficiency and Renewable Sources:** Governments and organizations are failing to sufficiently promote energy efficiency and renewable energy sources, leading to missed opportunities in addressing global challenges such as climate change and energy sustainability.

3. OBJECTIVES

I. Programming the Arduino to take analog Current and voltage values as input

II. To make a working model to demonstrate the working of the interface.

III. To visualize graphically using Python the current, voltage and power consumption from household appliances.

IV. To use the ARIMA model to predict power consumption using time-series analysis

V. Making a website to display the readings and results to the user.

VI. Send telegram alerts to the user in case of anomalies in power consumption.

VII. Allow user access to history of consumption of energy.

4. COMPONENTS USED

I. Arduino Nano

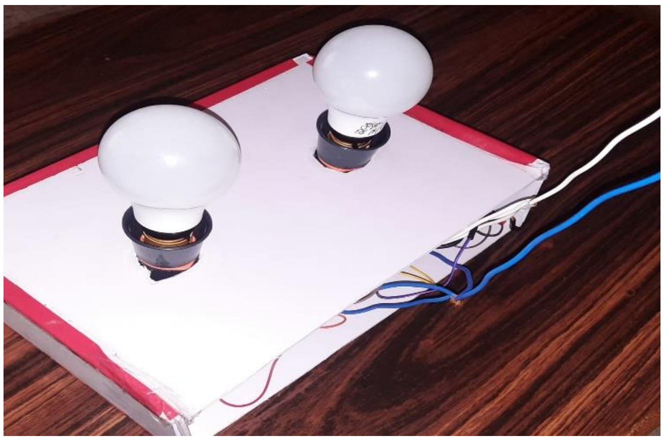
II. SCT-013 Current Sensor

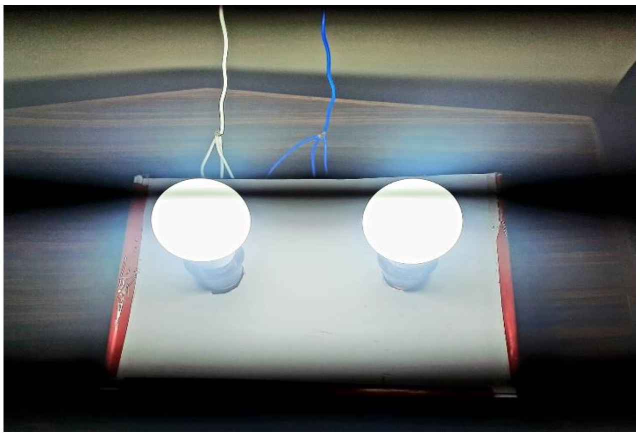
III. ZMPT101B-T Voltage Sensor

IV. RTC module

V. Light bulbs

5.SNAPSHOTS OF THE MODEL





6. SCHEMATIC DIAGRAM & SYSTEM ARCHITECTURE

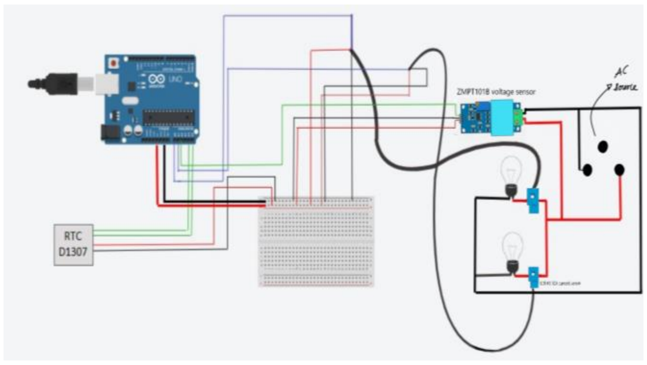


Fig: Schematic diagram for connections

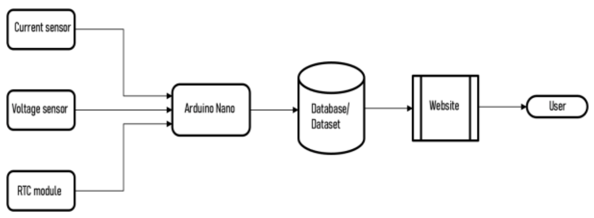


Fig: System Architecture

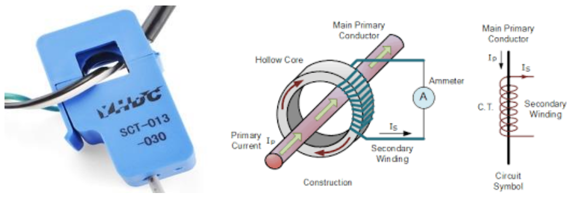
7. SENSORS AND THEIR WORKING

In our model, we have used two kinds of sensors:

- A current sensor – SCT013 30A
- A voltage sensor – ZMPT101B

Both the sensors output rms values and instantaneous values of current and voltage respectively.

7.1 SCT013 30 A Current Sensor



The SCT013 is a type of current sensor designed for non-invasive, contactless measurement of alternating current (AC) in electrical circuits. It is commonly used in applications where monitoring and measuring AC currents are essential, such as energy management systems, power monitoring, and various industrial and electronics projects.

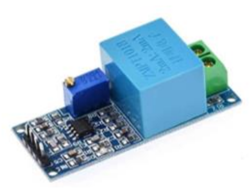
Key features of the SCT013 current sensor include:

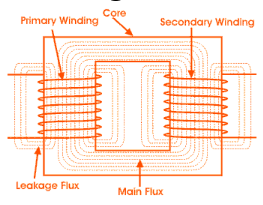
1. **Non-Invasive Design:** The SCT013 sensor does not require direct electrical contact with the current-carrying conductor. Instead, it utilizes a split-core design that can be easily clamped around the wire, making it safer and more convenient to install.
2. **Accuracy:** These sensors are designed to provide accurate measurements of AC currents. They come in various models with different current rating options to suit different applications.
3. **Output Signal:** The SCT013 typically provides an analog output signal, often in the form of a voltage or current that is proportional to the measured current. This signal can be interfaced with microcontrollers, data loggers, or other monitoring systems for data analysis and display.

The SCT013 current sensor, like other current transformers (CTs), operates on the principle of electromagnetic induction. Its working mechanism can be summarized as follows:

1. **Split-Core Design:** Current transformers often have a split-core design, allowing them to be easily clamped around the conductor without the need for disconnecting the circuit. This design facilitates convenient installation and measurements without interrupting the electrical flow.
2. **Magnetic Field Induction:** When an alternating current (AC) flows through a wire or conductor, it generates a magnetic field around the conductor, following Ampere's law. This magnetic field strength is directly proportional to the magnitude of the current.
3. **Magnetic Flux Creation:** The core of the CT is typically made of a magnetic material (e.g., iron or ferrite) that efficiently concentrates the magnetic field generated by the current-carrying conductor. This core serves to increase the magnetic flux passing through it, enhancing the sensitivity of the transformer.
4. **Induced Voltage:** The core of the CT is associated with a secondary winding, which is magnetically coupled to the primary winding (the conductor being measured). As the magnetic flux passing through the core changes with the AC current, it induces a voltage in the secondary winding. This induced voltage is proportional to the rate of change of magnetic flux and, therefore, to the primary current.
5. **Analog Output:** The induced voltage in the secondary winding is the output signal of the current transformer. This output voltage is typically an analog signal that represents the current in the primary circuit. The CT output is scaled to match the specific transformation ratio of the transformer, allowing for easy measurement and interpretation of the primary current.

7.2 ZMPT101B Voltage Sensor





The ZMPT101B voltage sensor is a specialized electronic component primarily designed for measuring alternating current (AC) voltage. It is commonly used in various applications where the measurement of AC voltage levels is required. Some key features and characteristics of the ZMPT101B voltage sensor:

1. **AC Voltage Measurement:** The ZMPT101B is designed specifically for the measurement of AC voltage. It can accurately measure the magnitude of AC voltage signals in a circuit.
2. **Non-Invasive:** Similar to a current transformer (CT), the ZMPT101B is non-invasive, meaning it does not require direct electrical contact with the voltage source. It can be connected to the circuit without disrupting the flow of electricity.
3. **Voltage Step-Down:** This sensor typically steps down the measured AC voltage to a lower, more manageable level for further processing. The output is often in the form of a scaled voltage signal.
4. **Output Signal:** The ZMPT101B provides an analog output signal that is proportional to the AC voltage being measured. The output voltage is typically scaled and can be interfaced with microcontrollers, analog-to-digital converters (ADCs), or other measurement and control systems.

A Voltage Transformer (VT) works on the principle of electromagnetic induction. This principle, established by Faraday's law of electromagnetic induction, states that a changing magnetic field within a closed loop of wire induces an electromotive force (EMF) or voltage in that wire.

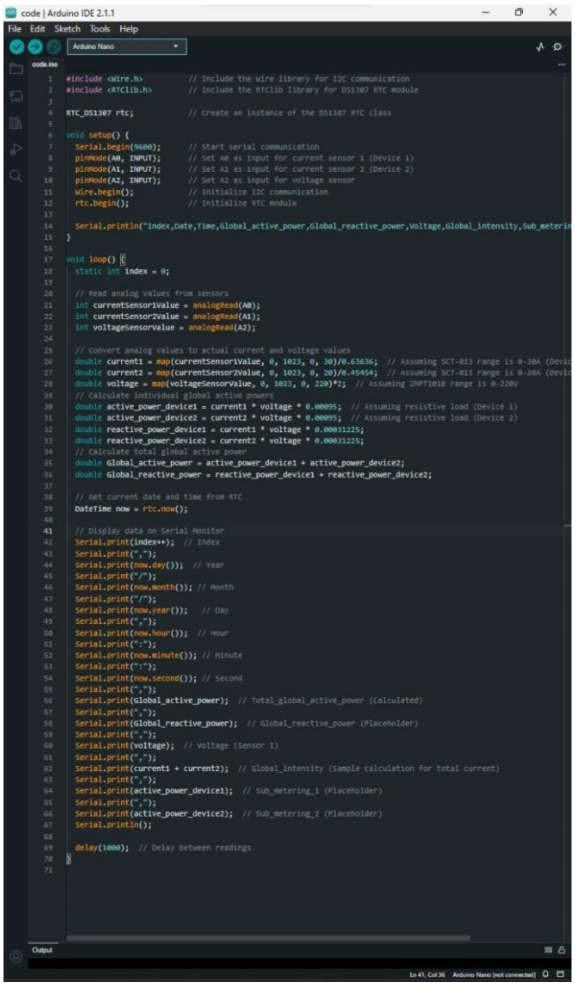
1. **Voltage Transformer (VT):** A Voltage Transformer steps down high-voltage AC signals for measurement and protection.
2. **Magnetic Flux and Induced Voltage:** VTs use changing magnetic flux to induce a proportional voltage in their secondary windings.

8. DATA ANALYSIS

8.1 Arduino code

Arduino NANO was used to collect current and voltage data using the Current and Voltage sensors.

```
#include <Wire.h>          // Include the Wire library for I2C communication

#include <RTClib.h>        // Include the RTClib library for DS1307 RTC module

RTC_DS1307 rtc;            // Create an instance of the DS1307 RTC class

void setup() {

  Serial.begin(9600);      // Start serial communication

  pinMode(A0, INPUT);      // Set A0 as input for current sensor 1 (Device 1)

  pinMode(A1, INPUT);      // Set A1 as input for current sensor 2 (Device 2)

  pinMode(A2, INPUT);      // Set A2 as input for voltage sensor

  Wire.begin();            // Initialize I2C communication

  rtc.begin();                                                  // Initialize RTC module
Serial.println("Index,Date,Time,Global_active_power,Global_reactive_power,Voltage,Global_intensity,Sub_mete
ring_1,Sub_metering_2");

}

void loop() {

  static int index = 0;

  // Read analog values from sensors

  int currentSensor1Value = analogRead(A0);

  int currentSensor2Value = analogRead(A1);

  int voltageSensorValue = analogRead(A2);

  // Convert analog values to actual current and voltage values
  double current1 = map(currentSensor1Value, 0, 1023, 0, 30)/0.63636;  // Assuming SCT-013 range is 0-30A
(Device 1)
  double current2 = map(currentSensor2Value, 0, 1023, 0, 20)/0.45454;  // Assuming SCT-013 range is 0-20A
(Device 2)
  double voltage = map(voltageSensorValue, 0, 1023, 0, 220)*2;  // Assuming ZMPT101B range is 0-220V
  // Calculate individual global active powers
  double active_power_device1 = current1 * voltage * 0.00095;  // Assuming resistive load (Device 1)
  double active_power_device2 = current2 * voltage * 0.00095;  // Assuming resistive load (Device 2)
  double reactive_power_device1 = current1 * voltage * 0.00031225;
  double reactive_power_device2 = current2 * voltage * 0.00031225;
  // Calculate total global active power
  double Global_active_power = active_power_device1 + active_power_device2;
  double Global_reactive_power = reactive_power_device1 + reactive_power_device2;
  // Get current date and time from RTC
  DateTime now = rtc.now();
  // Display data on Serial Monitor
  Serial.print(index++);  // Index
  Serial.print(",");
  Serial.print(now.day());  // Year
  Serial.print("/");
  Serial.print(now.month()); // Month
  Serial.print("/");
  Serial.print(now.year());   // Day
  Serial.print(",");
  Serial.print(now.hour());  // Hour
  Serial.print(":");
  Serial.print(now.minute()); // Minute
  Serial.print(":");
  Serial.print(now.second()); // Second
  Serial.print(",");
  Serial.print(Global_active_power);  // Total_global_active_power (Calculated)
  Serial.print(",");
  Serial.print(Global_reactive_power);  // Global_reactive_power (Placeholder)
  Serial.print(",");
  Serial.print(voltage);  // Voltage (Sensor 1)
  Serial.print(",");
  Serial.print(current1 + current2);  // Global_intensity (Sample calculation for total current)
  Serial.print(",");
  Serial.print(active_power_device1);  // Sub_metering_1 (Placeholder)
  Serial.print(",");
  Serial.print(active_power_device2);  // Sub_metering_2 (Placeholder)
  Serial.println();
  delay(1000);  // Delay between readings
}
```



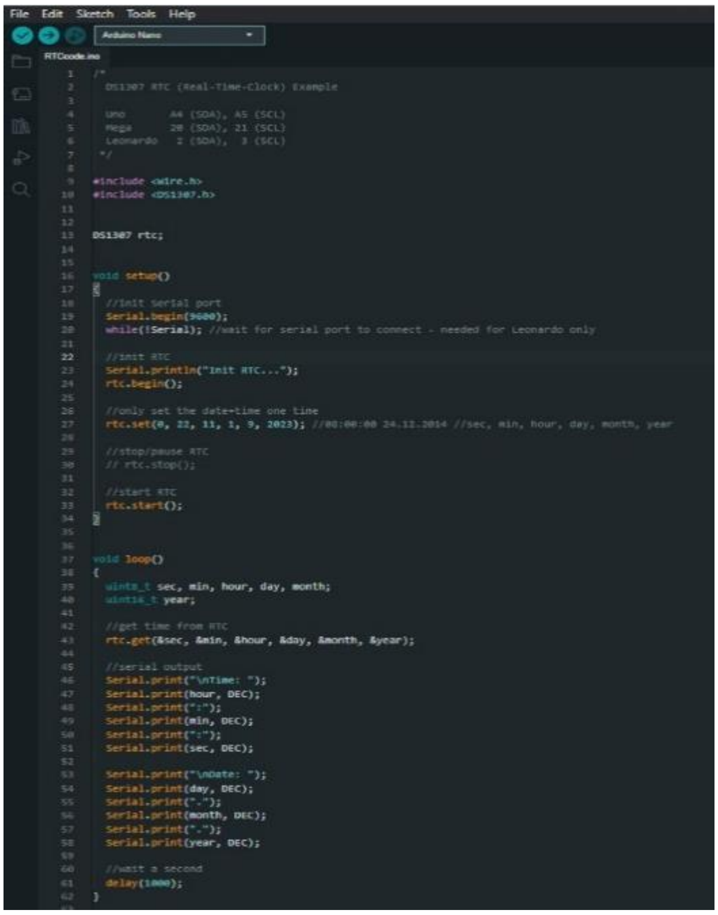
```
#include <Wire.h>          // Include the Wire library for I2C communication
#include <RTClib.h>        // Include the RTClib library for DS1307 RTC module

RTC_DS1307 rtc;            // Create an instance of the DS1307 RTC class

void setup() {
  Serial.begin(9600);      // Start serial communication
  pinMode(A0, INPUT);      // Set A0 as input for current sensor 1 (Device 1)
  pinMode(A1, INPUT);      // Set A1 as input for current sensor 2 (Device 2)
  pinMode(A2, INPUT);      // Set A2 as input for voltage sensor
  Wire.begin();            // Initialize I2C communication
  rtc.begin();             // Initialize RTC module

  Serial.println("Index,Date,Time,Global_active_power,Global_reactive_power,Voltage,Global_intensity,Sub_meterin
}

void loop() {
  static int index = 0;

  // Read analog values from sensors
  int currentSensor1Value = analogRead(A0);
  int currentSensor2Value = analogRead(A1);
  int voltageSensorValue = analogRead(A2);

  // Convert analog values to actual current and voltage values
  double current1 = map(currentSensor1Value, 0, 1023, 0, 30)/0.63636;  // Assuming SCT-013 range is 0-30A (Devic
  double current2 = map(currentSensor2Value, 0, 1023, 0, 20)/0.45454;  // Assuming SCT-013 range is 0-20A (Devic
  double voltage = map(voltageSensorValue, 0, 1023, 0, 220)*2;  // Assuming ZMPT101B range is 0-220V
  // Calculate individual global active powers
  double active_power_device1 = current1 * voltage * 0.00095;  // Assuming resistive load (Device 1)
  double active_power_device2 = current2 * voltage * 0.00095;  // Assuming resistive load (Device 2)
  double reactive_power_device1 = current1 * voltage * 0.00031225;
  double reactive_power_device2 = current2 * voltage * 0.00031225;
  // Calculate total global active power
  double Global_active_power = active_power_device1 + active_power_device2;
  double Global_reactive_power = reactive_power_device1 + reactive_power_device2;

  // Get current date and time from RTC
  DateTime now = rtc.now();

  // Display data on Serial Monitor
  Serial.print(index++);  // Index
  Serial.print(",");
  Serial.print(now.day());  // Year
  Serial.print("/");
  Serial.print(now.month()); // Month
  Serial.print("/");
  Serial.print(now.year());   // Day
  Serial.print(",");
  Serial.print(now.hour());  // Hour
  Serial.print(":");
  Serial.print(now.minute()); // Minute
  Serial.print(":");
  Serial.print(now.second()); // Second
  Serial.print(",");
  Serial.print(Global_active_power);  // Total_global_active_power (Calculated)
  Serial.print(",");
  Serial.print(Global_reactive_power);  // Global_reactive_power (Placeholder)
  Serial.print(",");
  Serial.print(voltage);  // Voltage (Sensor 1)
  Serial.print(",");
  Serial.print(current1 + current2);  // Global_intensity (Sample calculation for total current)
  Serial.print(",");
  Serial.print(active_power_device1);  // Sub_metering_1 (Placeholder)
  Serial.print(",");
  Serial.print(active_power_device2);  // Sub_metering_2 (Placeholder)
  Serial.println();

  delay(1000);  // Delay between readings
}
```

8.2 Code for reading date and time values using RTC module

```
/*
  DS1307 RTC (Real-Time-Clock) Example
  Uno       A4 (SDA), A5 (SCL)
  Mega      20 (SDA), 21 (SCL)
  Leonardo  2 (SDA),  3 (SCL)
 */
#include <Wire.h>
#include <DS1307.h>
```

```
DS1307 rtc;
void setup()
{
  //init Serial port
  Serial.begin(9600);
  while(!Serial); //wait for serial port to connect - needed for Leonardo only
  //init RTC
  Serial.println("Init RTC...");
  rtc.begin();
  //only set the date+time one time
  rtc.set(0, 22, 11, 1, 9, 2023); //08:00:00 24.12.2014 //sec, min, hour, day, month, year
  //stop/pause RTC
  // rtc.stop();
  //start RTC
  rtc.start();
}
void loop()
{
  uint8_t sec, min, hour, day, month;
  uint16_t year;
  //get time from RTC
  rtc.get(&sec, &min, &hour, &day, &month, &year);
  //serial output
  Serial.print("\nTime: ");
  Serial.print(hour, DEC);
  Serial.print(":");
  Serial.print(min, DEC);
  Serial.print(":");
  Serial.print(sec, DEC);
```

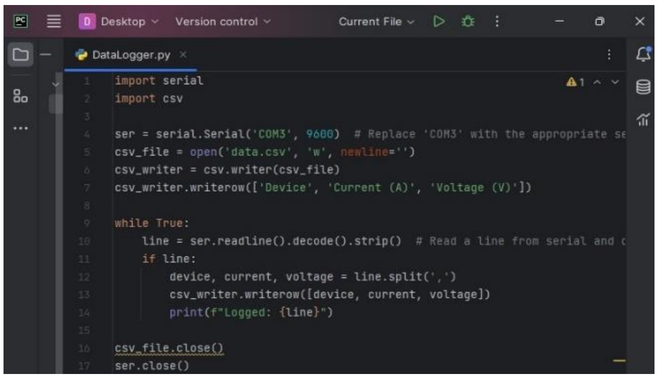
```
 Serial.print("\nDate: ");
 Serial.print(day, DEC);
 Serial.print(".");
 Serial.print(month, DEC);
 Serial.print(".");
 Serial.print(year, DEC);
 //wait a second
 delay(1000);
}
```



```
/*
  DS1307 RTC (Real-Time-Clock) Example

  Uno       A4 (SDA), A5 (SCL)
  Mega      20 (SDA), 21 (SCL)
  Leonardo   2 (SDA),  3 (SCL)
 */

#include <Wire.h>
#include <DS1307.h>


DS1307 rtc;


void setup()
{
  //init Serial port
  Serial.begin(9600);
  while(!Serial); //wait for serial port to connect - needed for Leonardo only

  //init RTC
  Serial.println("Init RTC...");
  rtc.begin();

  //only set the date+time one time
  rtc.set(0, 22, 11, 1, 9, 2023); //08:00:00 24.12.2014 //sec, min, hour, day, month, year

  //stop/pause RTC
  // rtc.stop();

  //start RTC
  rtc.start();
}


void loop()
{
  uint8_t sec, min, hour, day, month;
  uint16_t year;

  //get time from RTC
  rtc.get(&sec, &min, &hour, &day, &month, &year);

  //serial output
  Serial.print("\nTime: ");
  Serial.print(hour, DEC);
  Serial.print(":");
  Serial.print(min, DEC);
  Serial.print(":");
  Serial.print(sec, DEC);

  Serial.print("\nDate: ");
  Serial.print(day, DEC);
  Serial.print(".");
  Serial.print(month, DEC);
  Serial.print(".");
  Serial.print(year, DEC);

  //wait a second
  delay(1000);
}
```

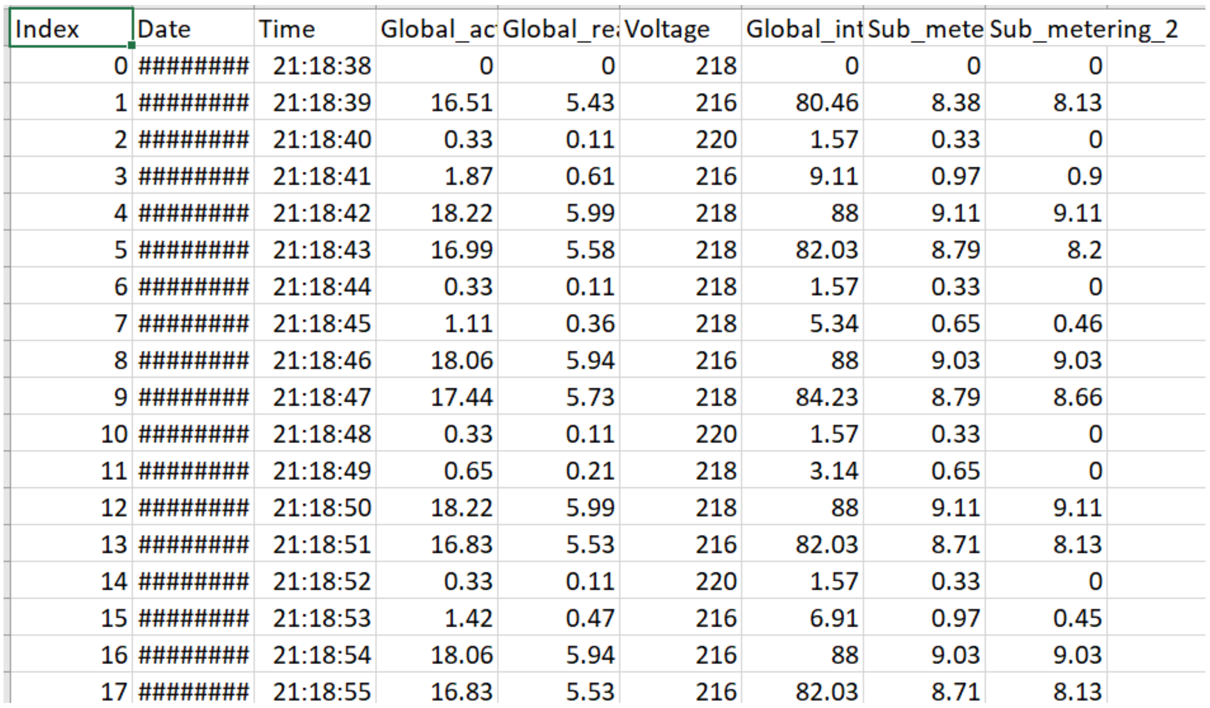
8.3 Python code to write into CSV file

A python code was written to write the data obtained from Arduino into a csv file of the format required.



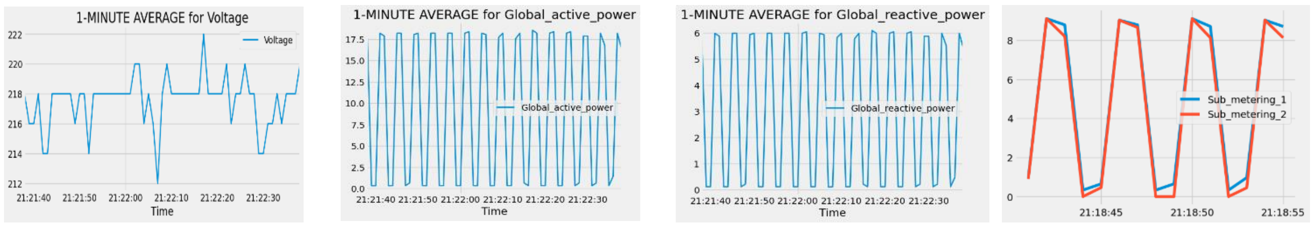
```python
import serial
import csv

ser = serial.Serial('COM3', 9600)  # Replace 'COM3' with the appropriate se
csv_file = open('data.csv', 'w', newline='')
csv_writer = csv.writer(csv_file)
csv_writer.writerow(['Device', 'Current (A)', 'Voltage (V)'])

while True:
    line = ser.readline().decode().strip()  # Read a line from serial and c
    if line:
        device, current, voltage = line.split(',')
        csv_writer.writerow([device, current, voltage])
        print(f"Logged: {line}")

csv_file.close()
ser.close()
```

DATASET OBTAINED

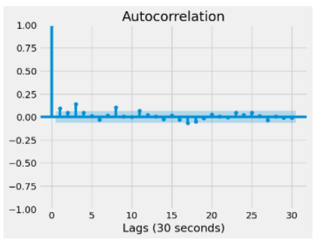


Index	Date	Time	Global_ac	Global_re	Voltage	Global_int	Sub_mete	Sub_metering_2
0	########	21:18:38	0	0	218	0	0	0
1	########	21:18:39	16.51	5.43	216	80.46	8.38	8.13
2	########	21:18:40	0.33	0.11	220	1.57	0.33	0
3	########	21:18:41	1.87	0.61	216	9.11	0.97	0.9
4	########	21:18:42	18.22	5.99	218	88	9.11	9.11
5	########	21:18:43	16.99	5.58	218	82.03	8.79	8.2
6	########	21:18:44	0.33	0.11	218	1.57	0.33	0
7	########	21:18:45	1.11	0.36	218	5.34	0.65	0.46
8	########	21:18:46	18.06	5.94	216	88	9.03	9.03
9	########	21:18:47	17.44	5.73	218	84.23	8.79	8.66
10	########	21:18:48	0.33	0.11	220	1.57	0.33	0
11	########	21:18:49	0.65	0.21	218	3.14	0.65	0
12	########	21:18:50	18.22	5.99	218	88	9.11	9.11
13	########	21:18:51	16.83	5.53	216	82.03	8.71	8.13
14	########	21:18:52	0.33	0.11	220	1.57	0.33	0
15	########	21:18:53	1.42	0.47	216	6.91	0.97	0.45
16	########	21:18:54	18.06	5.94	216	88	9.03	9.03
17	########	21:18:55	16.83	5.53	216	82.03	8.71	8.13

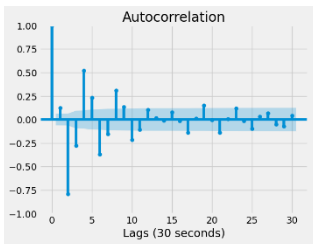
9.3.1 OUTPUT GRAPHS FROM OUR MODEL



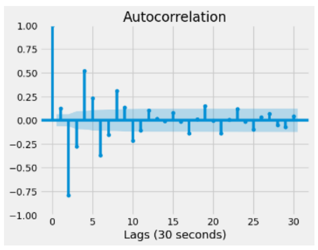
Plotted graphs of Voltage, Active power, Reactive power, and Submetering power readings



Autocorrelation for Voltage



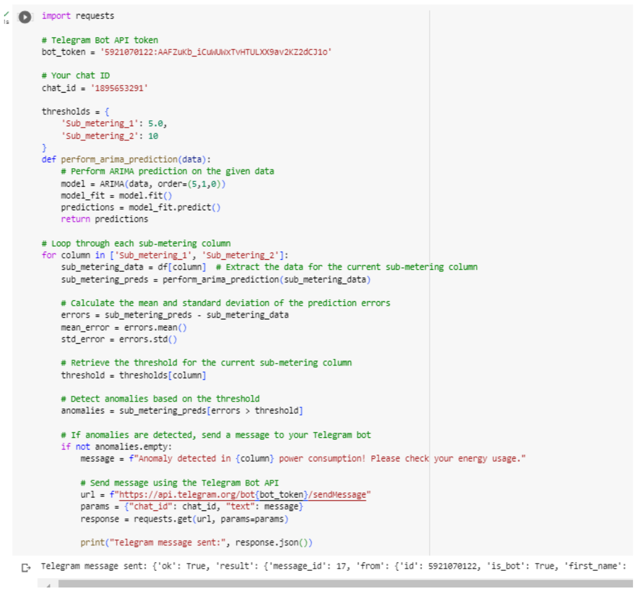
Autocorrelation for Active Power



Autocorrelation for Reactive Power

9.5 AUTOMATED ALERTS

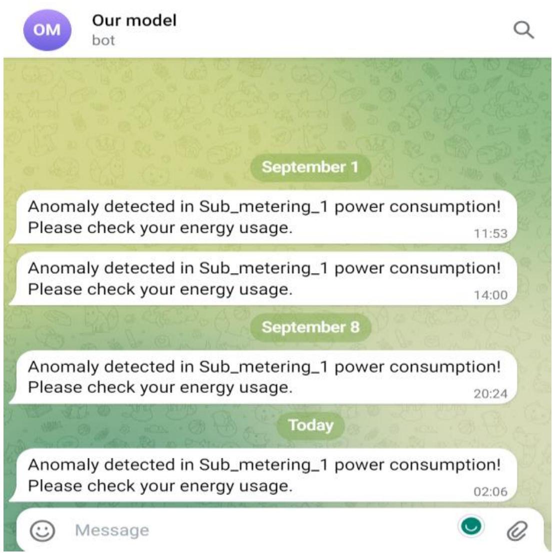
We first analysed and identified the maximum power our model has gone up to 9.11. After determining the highest power, we set a threshold for the power.
Once the threshold is exceeded, an alert is sent to the user via Telegram.
We used the Telegram bot API to do the same.



```python
import requests

# Telegram Bot API token
bot_token = '5921070122:AAFZuKb_iCuWUWxTvHTULXX9av2KZ2dCJ1o'

# Your chat ID
chat_id = '1895653291'

thresholds = {
    'Sub_metering_1': 5.0,
    'Sub_metering_2': 10
}
def perform_arima_prediction(data):
    # Perform ARIMA prediction on the given data
    model = ARIMA(data, order=(5,1,0))
    model_fit = model.fit()
    predictions = model_fit.predict()
    return predictions

# Loop through each sub-metering column
for column in ['Sub_metering_1', 'Sub_metering_2']:
    sub_metering_data = df[column]  # Extract the data for the current sub-metering column
    sub_metering_preds = perform_arima_prediction(sub_metering_data)

    # Calculate the mean and standard deviation of the prediction errors
    errors = sub_metering_preds - sub_metering_data
    mean_error = errors.mean()
    std_error = errors.std()

    # Retrieve the threshold for the current sub-metering column
    threshold = thresholds[column]

    # Detect anomalies based on the threshold
    anomalies = sub_metering_preds[errors > threshold]

    # If anomalies are detected, send a message to your Telegram bot
    if not anomalies.empty:
        message = f"Anomaly detected in {column} power consumption! Please check your energy usage."

        # Send message using the Telegram Bot API
        url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
        params = {"chat_id": chat_id, "text": message}
        response = requests.get(url, params=params)

        print("Telegram message sent:", response.json())
```

```
Telegram message sent: {'ok': True, 'result': {'message_id': 17, 'from': {'id': 5921070122, 'is_bot': True, 'first_name':
```

Code to send a telegram alert to user as soon as Power consumption in Sub_Metering 1 exceeds 5 W



Our model
bot

September 1

Anomaly detected in Sub_metering_1 power consumption! Please check your energy usage. 11:53

Anomaly detected in Sub_metering_1 power consumption! Please check your energy usage. 14:00

September 8

Anomaly detected in Sub_metering_1 power consumption! Please check your energy usage. 20:24

Today

Anomaly detected in Sub_metering_1 power consumption! Please check your energy usage. 02:06

Alert sent to user once threshold is crossed

9 WEBPAGE TO DISPLAY VISUALIZED DATA

A static webpage was created using HTML CSS to display the visualised data, i.e., graphs to the user. The main purpose was to provide the user with a platform to view the consumption with ease and make timely decisions. The webpage has the following features :

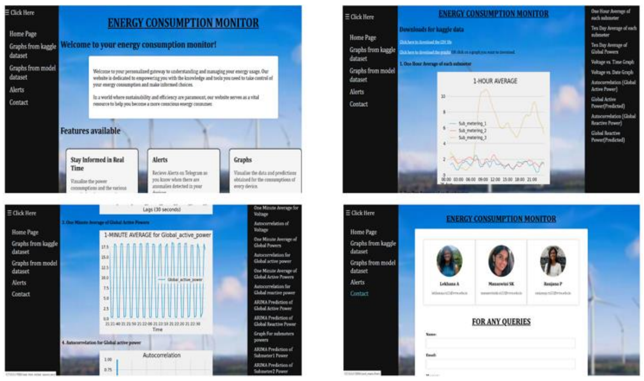
I. **Menu:** An option to switch between the various tabs as mentioned. It is present in the top left corner and the three line button can be used to collapse the menu.

II. **Home Page:** An introductory page to welcome the user and the different options for the user like alerts and graph are provided with buttons linking to that page.

III. **Graphs:** This is the main purpose behind the creation of the website. On the right hand side, there is an option to navigate within the page. Here the user can view all the graphs like Autocorrelation, ARIMA prediction for each device, V vs. T, Global active/reactive powers(predicted), etc. In this page, the user can download the csv file as well as any of the graphs by simply clicking on any one of them.

IV. **Alerts:** On clicking this tab, the user will be taken to the telegram page where they can view the recent alerts for any of their devices.

V. **Contact:** For any queries, the users can send a mail to any of the provided email-ids. A form is provided to input any queries or suggestions.


ENERGY CONSUMPTION MONITOR
Welcome to your energy consumption monitor!
Features available
Stay Informed in Real Time
Alerts
Graphs
Home Page
Graphs from kaggle dataset
Graphs from model dataset
Alerts
Contact
1-HOUR AVERAGE
1-MINUTE AVERAGE for Global_active_power
Autocorrelation
FOR ANY QUERIES

10. FUTURE SCOPE IN AREA

There is great potential for development in this area, especially because of the increasing demand for energy in everyday life. The following are certain areas where further research and integration can be done:

1. **Automating Data Input:** Streamlining the process by eliminating manual dataset uploads and Python script execution, reducing the need for human intervention.
2. **Utilizing Advanced Libraries:** Leveraging libraries such as Apache Kafka and Apache Spark to achieve this automation efficiently.
3. **Anomaly Detection Focus:** Emphasizing efforts on comprehending the root causes behind anomalies in the data.
4. **Tailored Energy Saving Tips:** Providing personalized recommendations for energy consumption optimization.
5. **Establishing Alert History Database:** Creating a historical repository for alerts, accomplished through technologies like SQL databases or SD card storage.
6. **Carbon Footprint Calculation:** Implementing tools to compute and

manage carbon footprints, contributing to environmental sustainability.

7. **Comparison Features:** Enabling users to assess their consumption against similar households or industry benchmarks through comparative tools.

11. CONCLUSION

In conclusion, the development and implementation of an energy consumption monitor are crucial steps towards achieving a more sustainable and responsible approach to energy utilization. Such a monitor not only empowers individuals, businesses, and governments to make informed decisions for reducing their environmental impact but also promotes energy efficiency and the adoption of renewable energy sources. By eliminating manual processes, providing personalized tips, and facilitating historical data analysis, an energy consumption monitor becomes a valuable tool in the journey towards a greener and more efficient energy future. It offers the potential to drive positive changes in behaviour, reduce carbon footprints, and foster a collective effort to address global energy challenges.

12. ACKNOWLEDGEMENT

We extend our heartfelt gratitude to our professors Dr. Girish S, Department of Mechanical Engineering, RVCE, for his guidance throughout this project. His encouragement and expertise have been instrumental in enabling us to carry out this project on Energy Consumption Monitor. We are truly grateful for his help and the opportunity they have provided us to pursue this endeavour. We would also like to thank Pavan Shetty from Wirin Lab and the lab assistants for their help.

13. REFERENCES

[1] Yi Wang, Qixin Chen, Tao Hong, Chongqing Kang, “Review of Smart Meter Data Analytics: Applications, Methodologies, and Challenges”, 10, 3, March 2018.

[2] Md Abul Midul, Shahadat Pranta, Al Shaharea Biddut, Sifatul Siam, Md. Rifat Hazari, Md Mannan, “Design and Implementation of IoT-Based Smart Energy Meter to Augment Residential Energy Consumption”, March 2023.

[3] Gitanjali Mehta, Vinod Kumar Yadav, Ruqaiya Khanam, “A Novel IoT based Smart Energy Meter for Residential Energy Management in Smart Grid Infrastructure”, October 2021.

[4] Jungsuk Kwac, June Flora, Ram Rajagopal, “Household Energy Consumption Segmentation Using Hourly Data”, 5, 1, January 2014.

[5] Hussain Shareef, Maytham S. Ahmed, Azah Mohamed, Eslam Al Hassan, “Review on Home Energy Management System Considering Demand Responses,Smart Technologies,and Intelligent Controllers”, 6, April 2018.

[6] Kalluri Srinivasa Rao, S Ramana Kumar Joga, V. Hari Shankar Dinesh, S. Mounika, B. Musili

Naidu, M. Sai, “Innovative Digital Energy Meter with Overload Indication and Power Theft Monitoring”, July 2023.