Minor Project Final Report

ON

Smart Energy Meter

Submitted in partial fulfilment of the requirements
For the award of the degree of
B.Tech (Electronics and communication)

Submitted to -
Supervisor: Dr. Jaspreeti Singh

Submitted by -
Anand Kirar
Enrollment No- 06516412821

UNIVERSITY SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
Sector - 16C Dwarka, Delhi - 110075, India

Acknowledgements -

I would like to express my deepest gratitude to **Dr. Jaspreeti Singh**, my project supervisor, for her invaluable guidance, encouragement, and support throughout the course of this minor project on

Smart Energy Meter. Her expertise and insightful feedback have been instrumental in shaping the direction and outcome of this work.

I also wish to extend my heartfelt thanks to my institution, **Guru Gobind Singh Indraprastha University**, and the faculty of the **University School of Information and Communication Technology** for providing the resources and conducive environment necessary for the successful completion of this project.

Lastly, I am deeply thankful to my family and friends, whose continuous motivation and assistance helped me overcome challenges during the development of this project. This project has been an incredible learning experience, and I am grateful for the support and opportunities I have received.

Anand Kirar
B.Tech (Electronics and Communication)
Enrollment No: 06516412821

Index

1. Introduction and Overview -

The sustainable use of energy is one of the most significant challenges that our society is facing today. With the rapid growth in population, urbanization, and technological advancement, the demand for energy, particularly electricity, is increasing at an unprecedented rate. This growing demand underscores the critical need for innovative solutions to manage and utilize energy efficiently. One such solution lies in developing systems that provide accurate and real-time data on energy consumption. This project focuses on designing and implementing a smart energy meter using Arduino, a microcontroller-based platform that facilitates the monitoring of electrical parameters like current and voltage. The aim is to provide real-time information about energy usage, which can enable users to optimize consumption and reduce wastage effectively. Smart energy meters are a cornerstone of modern energy management systems, as they enable accurate monitoring, efficient utilization of energy resources, and enhanced awareness of consumption patterns.

While the project is still in progress and incomplete, the current phase demonstrates local energy monitoring functionality. This includes measuring current, calculating power consumption in a simulation environment, and displaying the results on an LCD. Additionally, the project has been designed with the potential for future scalability, including the incorporation of advanced features such as remote data transmission and IoT connectivity. These enhancements would enable remote monitoring and control, opening up possibilities for cloud-based analytics, trend analysis, and anomaly detection in energy usage. The implementation of such advanced systems has the potential to revolutionize energy consumption practices, enabling individuals and industries to make data-driven decisions for energy savings, reduced costs, and improved reliability. This aligns closely with global sustainability goals, promoting the reduction of carbon footprints and the responsible use of natural resources.

The smart energy meter designed as part of this project employs an Arduino microcontroller to measure current (and voltage in simulation), calculate energy usage, and display the results on an LCD using I2C. In the practical implementation, the ACS712 sensor is used for current measurement, providing a cost-effective and reliable solution for small-scale applications. The LCD serves as an accessible interface, allowing users to view energy consumption data in real time without requiring additional devices or software. While the current prototype does not include voltage measurement or remote transmission of data, the groundwork has been laid for these features. Future developments could involve integrating IoT-based components to enable cloud-based monitoring and remote access. These improvements would make the system more versatile, suitable for a wide range of applications, and aligned with the vision of a fully connected smart grid.

This report outlines the design, implementation, and evaluation of the smart energy meter, presenting detailed insights into both the simulation and the practical setup. It provides an in-depth analysis of the project's architecture, hardware components, software algorithms, and operational performance. Additionally, it discusses the challenges encountered during the development process, the limitations of the current system, and the prospects for future advancements. By addressing these aspects comprehensively, this report aims to offer a thorough understanding of the smart energy meter and its potential contributions to energy management and sustainability.

2. Objective -

---

a) Monitor power consumption locally by measuring voltage and current using sensors.

- **Objective:** The goal is to create a system capable of real-time monitoring of power usage by measuring key electrical parameters such as voltage and current to calculate power consumption.
- **How it's achieved:**
  - In the simulation, both voltage and current are measured using sensors, and the calculated power is displayed on an LCD screen.
  - In the practical implementation, only the ACS712 current sensor is used to measure the AC current, with the data displayed on the serial monitor and LCD screen.
- **Importance:** Monitoring energy usage locally provides insights into consumption patterns, helping users make informed decisions about energy efficiency and resource optimization.

b) Display power consumption using LCD.

- **Objective**: To present real-time data to users in a simple, readable format without requiring external devices like smartphones or computers.
- **How it's achieved**:
  - In the simulation, an LCD (e.g., 16x2) is used to display real-time values of voltage, current, and power.
  - In the practical project, data is displayed on the serial monitor and LCD using I2C, and only current measurement is implemented.
- **Importance**: The LCD in the simulation eliminates the need for complex setups, ensuring that even non-technical users can easily understand their power usage at a glance.

c) Simulate the circuit using Proteus to ensure functionality before hardware implementation.

- **Objective**: Use simulation tools to validate the design and functionality of the smart energy meter before building the hardware prototype.
- **How it's achieved:**
  - The Proteus simulation models both the ACS712 current sensor and the voltage sensor. It displays power data on a virtual LCD.
  - This simulation allows testing of various scenarios, debugging of code, and ensures the accuracy of calculations before moving to physical hardware.
- **Importance:** Simulation helps identify potential issues early, reduces development time, minimizes hardware costs, and facilitates a smoother transition to physical implementation.

d) Provide real-time data on energy usage to help users manage and optimize consumption.

- **Objective:** Enable users to make informed decisions about their energy usage by providing them with up-to-date information on power consumption.
- **How it's achieved:**
  - In the simulation, both voltage and current sensors contribute to the real-time data displayed on the LCD. In the practical project, current measurements are displayed on the serial monitor and LCD screen.
  - Continuous monitoring helps users track and potentially reduce excessive energy consumption.

- **Importance:** This feature promotes energy conservation, cost savings, and increased awareness of consumption patterns, supporting sustainable energy use.

e) Future Objective: Incorporate remote data transmission for cloud-based monitoring (currently pending).

- **Objective:** Prepare for future integration of remote monitoring capabilities, such as sending energy data to a cloud platform for advanced analytics and management.
- **How it will be achieved:**
  - Future iterations could incorporate modules like ESP8266 or GSM with the Arduino to transmit data to a remote server.
  - Users would be able to access the data remotely via mobile or web platforms for better control and analysis**.**
- **Importance:** Remote monitoring enables trend analysis, anomaly detection, and the ability to manage energy use from anywhere, providing a modern IoT-enabled solution.

3. Components -

Arduino UNO (ATmega328P)

Pins (3.3, 5, GND, Vin)

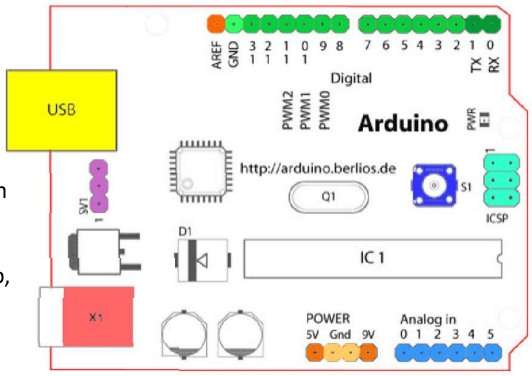
3.3V (6) – Supply 3.3 output volt

5V (7) – Supply 5 output volt

Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.

GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.

Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.



The Arduino UNO.

Digital I/O Pins:14 (of which 6 provide PWM output)

Analog Input Pins: 6 (DIP) or 8 (SMD)

DC Current per I/O Pin: 40 mA

Flash Memory: 32 KB

SRAM: 2 KB

EEPROM: 1 KB

Why Use Arduino in the Smart Energy Meter Project?

Arduino is chosen as the central processing unit for the smart energy meter project for several compelling reasons. Below are the key advantages and features that make Arduino ideal for this application:

1. Ease of Use

- **User-Friendly Platform:** Arduino boards are designed to be beginner-friendly, with an intuitive interface and easy-to-use IDE for programming. This makes it suitable for both novice and experienced developers.
- **Simple Programming:** The Arduino language is based on C/C++, which is well-documented and easy to learn, allowing for quick prototyping and implementation.

2. Versatility

- **Wide Range of Applications:** Arduino is highly versatile and can be adapted to numerous projects, from simple data collection to complex IoT systems.
- **Compatibility with Sensors and Modules:** It supports a wide range of sensors (e.g., ACS712 current sensor) and modules (e.g., LCD, GSM), making it a flexible choice for smart energy monitoring.

3. Open-Source Ecosystem

- **Access to Libraries:** Arduino offers a vast library ecosystem that simplifies working with various components, such as sensors, displays, and communication modules.
- **Community Support:** Arduino has a large and active community of developers who contribute tutorials, forums, and solutions to common issues.

4. Cost-Effectiveness

- **Affordable Hardware:** Arduino boards, such as the Arduino Uno, are inexpensive compared to other microcontrollers, making them accessible for small-scale projects.
- **Minimal Development Costs:** The open-source software and availability of resources reduce the cost of development.

5. Real-Time Data Processing

- **Analog and Digital Inputs:** Arduino can read both analog and digital signals, enabling it to interface seamlessly with sensors like the ACS712.
- **Efficient Data Processing:** It processes real-time sensor data to calculate current and power consumption, displaying the results on an LCD.

6. Scalability

- **Expandable Functionality:** Arduino projects can be easily expanded to include more advanced features, such as remote monitoring via GSM modules or IoT integration.

- **Future Upgrades:** The Arduino platform supports adding features like Wi-Fi connectivity or data storage, which are useful for future iterations of the smart energy meter.

7. Low Power Consumption

- Arduino boards are energy-efficient, consuming minimal power during operation. This makes them suitable for continuous monitoring in projects like energy meters.

8. Educational Value

- **Learning Opportunity:** Arduino serves as a learning tool, allowing developers to understand embedded systems, programming, and circuit design.
- **Practical Application:** By using Arduino, the project bridges theoretical concepts with hands-on practical implementation.

Why Not Use Other Microcontrollers?

- While there are alternatives like Raspberry Pi, ESP32, or STM32, Arduino is preferred for this project because:
    - **Simpler Setup:** Arduino requires less setup compared to platforms like Raspberry Pi, which involves an operating system.
    - **Low Complexity:** For a project focused on basic current measurement and power calculation, Arduino provides all the necessary functionality without overcomplicating the design.
    - **Cost Efficiency:** Compared to advanced boards, Arduino is more affordable and sufficient for the project's requirements.

LCD (16x2)

A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A Read/Write (R/W) pin that selects reading mode or writing mode

An Enable pin that enables writing to the registers

8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

I2C (Inter-Integrated Circuit)

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave communication protocol commonly used to connect low-speed peripherals to microcontrollers and processors. Developed by Philips Semiconductor (now NXP) in the 1980s, I2C has become a standard for inter-device communication due to its simplicity and efficiency.

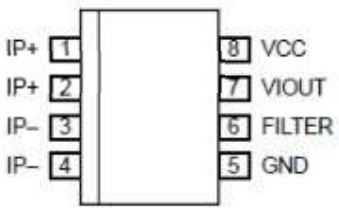
Key Features of I2C

1. **Two-Wire Interface:**
   - **SDA (Serial Data):** Transfers data between devices.
   - **SCL (Serial Clock):** Synchronizes data transfer.
2. **Addressable Devices:**
   - Each device on the I2C bus has a unique 7-bit or 10-bit address, allowing multiple devices to connect to the same bus.
3. **Bidirectional Communication:**
   - Both master and slave devices can send and receive data, enabling full-duplex communication.
4. **Multi-Master Capability:**
   - Multiple master devices can coexist on the same bus, although only one can control the bus at a time.
5. **Clock Speed:**
   - Supports various speeds, such as standard mode (100 kbps), fast mode (400 kbps), and high-speed mode (up to 3.4 Mbps).

Acs712

ACS712 is a Hall Effect-Based Linear Current Sensor it can measure both DC(Direct Current) and AC(Alternating Current).

Should always be connected in series.



Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

ACS712ELCTR-05B-T can measure 5 to -5 Ampere current. Where 185mV change in Output voltage from initial state represents 1-Ampere change in Input current.

ACS712ELCTR-20A-T can measure 20 to -20 Ampere current. Where 100mV change in Output voltage from initial state represents 1-Ampere change in Input current.

ACS712ELCTR-30A-T can measure 30 to -30 Ampere current. Where 66mV change in Output voltage from initial state represents 1-Ampere change in Input current.

What is the Hall Effect?

The Hall Effect occurs when a current-carrying conductor is placed in a magnetic field. Due to the interaction of the magnetic field with the moving charges in the conductor, a voltage is generated perpendicular to both the current and the magnetic field. This voltage is called the **Hall Voltage**, and it is directly proportional to the strength of the magnetic field and the current flowing through the conductor.

How the ACS712 Utilizes the Hall Effect

The ACS712 is a Hall-effect-based current sensor that can measure both **AC (alternating current)** and **DC (direct current)**. Here's how it works:

1. **Internal Current Path:**
    - The ACS712 has an internal conductive path where the measured current flows.

    - As current flows through this path, it generates a magnetic field around it.
2. **Hall Effect Sensor:**
    - Inside the ACS712, there is a Hall-effect sensor positioned near the internal conductive path.
    - This sensor detects the magnetic field generated by the current and converts it into a proportional Hall voltage.
3. **Signal Processing:**
    - The Hall voltage is amplified and processed within the ACS712 to produce an output voltage that is linearly proportional to the input current.
    - This voltage can then be read by a microcontroller like the Arduino for further calculations.

Application of the Hall Effect in the Smart Energy Meter Project

In this project, the Hall Effect allows the ACS712 to measure **AC current** without the need for direct electrical contact, providing several advantages:

1. **Current Measurement:**
    - The ACS712 detects the current flowing through a load by measuring the magnetic field generated by the current.
    - The Arduino reads the sensor's output voltage and translates it into the corresponding current value using a calibration factor.
2. **Safety:**
    - The Hall Effect enables non-invasive current measurement. There is no direct electrical connection between the sensor and the high-current path, reducing the risk of electrical hazards.
3. **Accuracy:**
    - The ACS712 provides precise readings of both AC and DC currents, making it versatile for various applications.
    - In this project, it is used specifically to measure **AC current** from household appliances.
4. **Power Calculation:**
    - The Arduino uses the current readings from the ACS712 and assumes a known or fixed voltage (if not measured) to calculate power consumption using the formula: P=V×IP = V \times IP=V×I
5. **Real-Time Monitoring:**
    - By continuously measuring current, the project provides real-time data on energy usage, displayed on the LCD.

Advantages of the Hall Effect in the ACS712 for This Project

1. **Isolation:**
    - The Hall Effect ensures electrical isolation between the current-carrying conductor and the sensing circuit, making it safe to measure high currents.
2. **AC and DC Compatibility:**
    - The ACS712 can measure both AC and DC currents, but in this project, it is focused on AC current measurement.

3. **Compact Design:**
    - The ACS712 integrates the Hall sensor and the signal processing circuitry into a single compact module, simplifying the project's hardware design.
4. **Ease of Integration:**
    - The sensor outputs an analog voltage proportional to the current, which can be easily read by the Arduino's ADC (Analog-to-Digital Converter) pins.

Limitations and Considerations

While the Hall Effect and the ACS712 are effective, there are some considerations to keep in mind:

- **Accuracy:** The ACS712's accuracy can be affected by external magnetic fields or noise, so shielding or careful placement is required.
- **Voltage Measurement:** In this project, only current is measured. To measure power accurately, voltage must either be assumed constant or measured separately.
- **Temperature Drift:** The sensor's performance may vary slightly with temperature changes, which could affect long-term stability.

Proteus Simulation

The Proteus simulation played a critical role in the development of the smart energy meter, offering several advantages during the design phase. Below are the key purposes and benefits of using Proteus for this project:

1. Circuit Validation

The primary purpose of using Proteus for simulation was to validate the circuit design. It ensured that all components, including the Arduino Uno, ACS712 current sensor, and LCD, were correctly connected and functioned as intended. By simulating the circuit before building the physical prototype, potential design errors could be identified and corrected early in the development process. This step minimized the risk of hardware failure and ensured that the system would work efficiently once implemented physically.

2. Behavioural Analysis

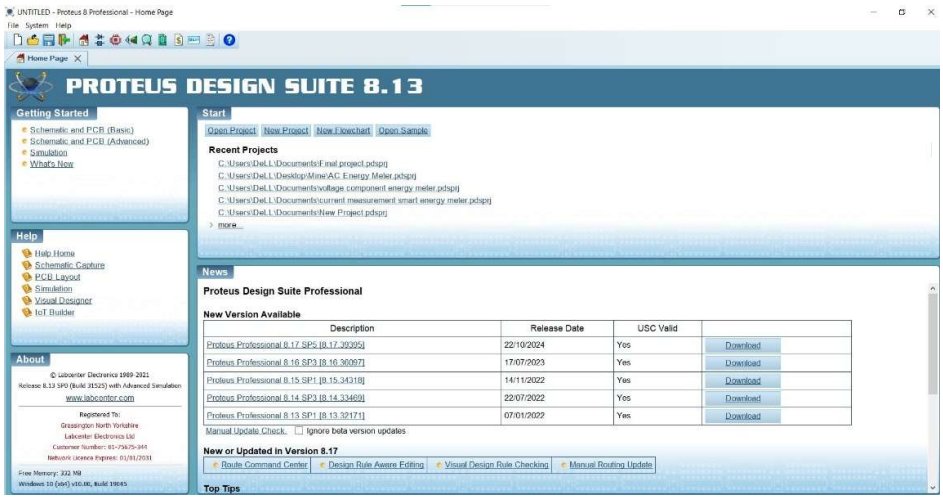
Proteus allowed for the observation of the system's behaviour in real-time. This included monitoring how the ACS712 current sensor measured current and how the Arduino processed these measurements. It also enabled verification of how the results were displayed on the LCD screen. Through simulation, the system's responsiveness to changes in load and how accurately the readings were displayed were analyzed. This step ensured the functionality of the energy meter before physical assembly.

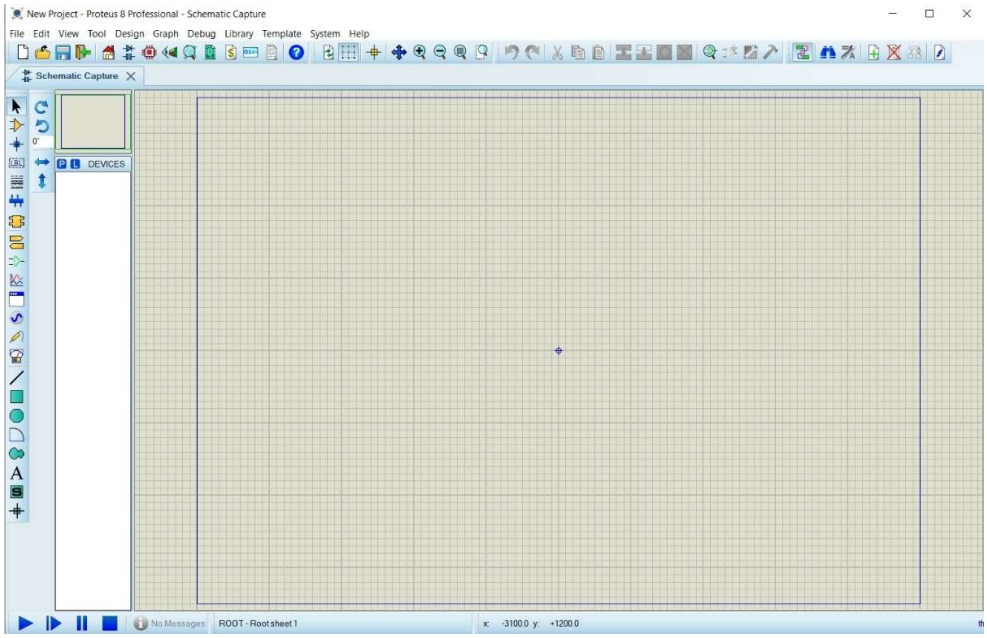
3. Cost-Effectiveness

Simulating the smart energy meter circuit in Proteus provided a cost-effective method to identify and resolve any potential issues without the need for purchasing physical components. This approach prevented unnecessary expenditure on parts and reduced the risk of damaging hardware during testing. Additionally, the ability to simulate different scenarios and conditions (such as varying the load or sensor behaviour) allowed for troubleshooting without requiring expensive trial-and-error with actual components.

4. Prototyping

Proteus offered the opportunity to create a virtual prototype of the smart energy meter. This virtual prototype allowed for iterative improvements before committing to the physical build. Changes to the design could be easily implemented in the simulation, tested, and refined. This iterative approach ensured that the final physical implementation of the energy meter would be both reliable and efficient.



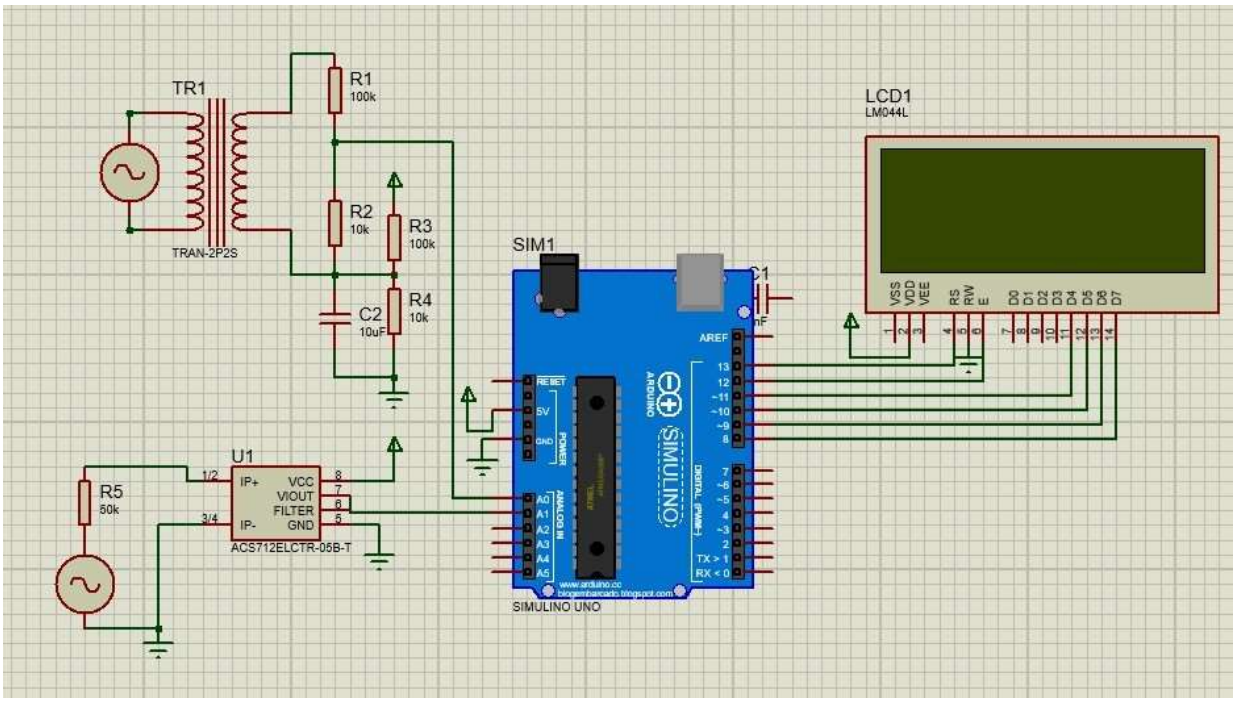


4. Circuit Design and Simulation -

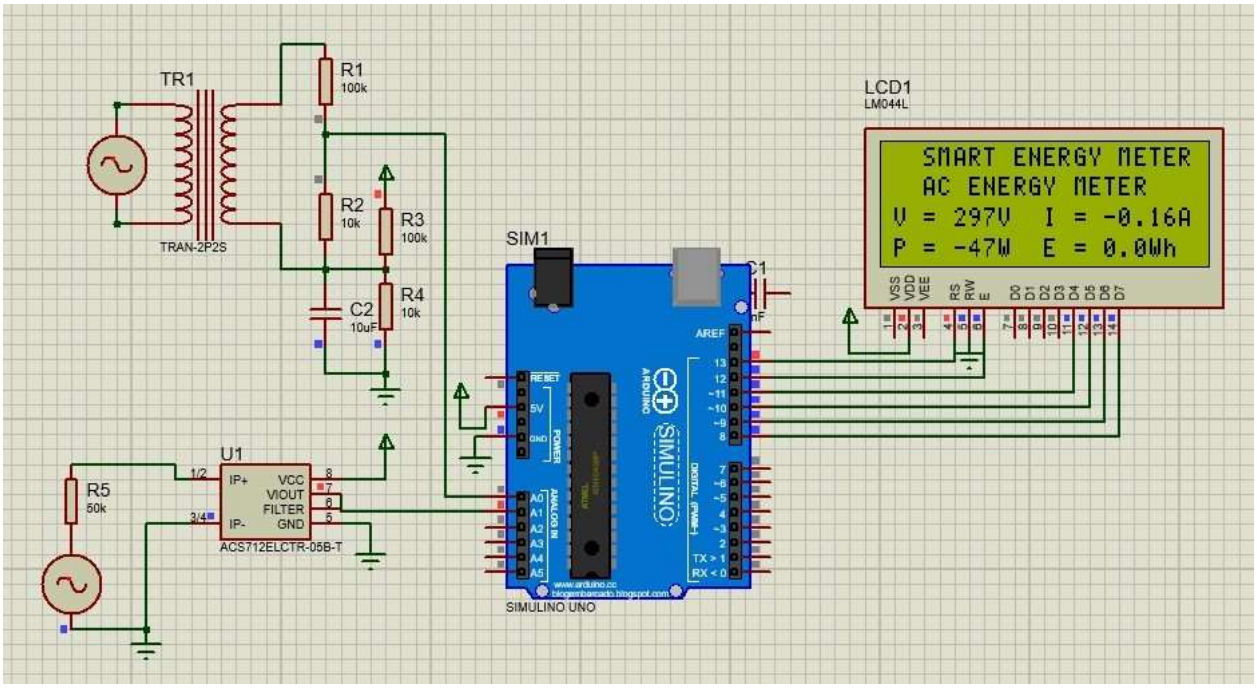
For the smart energy meter project, I used **Proteus** for circuit simulation to ensure the system's functionality before moving to physical implementation.

1. **Voltage Measurement**:
    - A **step-down transformer** is used to measure the voltage in the circuit. It steps down the AC voltage to a manageable level, which is then measured using the Arduino.
2. **Current Measurement**:
    - The **ACS712 current sensor** is used to measure the AC current. This sensor is ideal for monitoring the current without direct contact with the high-voltage part of the circuit, providing accurate measurements for energy consumption.
3. **Display**:
    - The measured voltage, current, and power consumption are displayed on a **16x2 LCD**. This provides a simple, readable interface for users to view their real-time energy consumption data.
4. **Arduino UNO**:
    - The **Arduino UNO** is used as the controller for processing the sensor data. It reads the data from both the voltage and current sensors and computes the power consumption. This data is then sent to the LCD for display.
5. **Proteus Simulation**:
    - The simulation ensures the entire system works as expected by modelling the interaction between the components. It helps visualize how the data flows from the sensors to the Arduino and how the final output is presented on the LCD screen. The simulation also allows for testing and debugging before the physical setup.

In the simulation, both voltage and current are displayed on the LCD, along with the calculated power (in watts). The **ACS712** provides accurate real-time current measurements, while the **step-down transformer** helps simulate voltage readings. This allows for a comprehensive simulation of the energy meter.



Smart Energy Meter Circuit Diagram (Idle Mode)



Smart Energy Meter Circuit Diagram (Run Mode)

Need to convert AC current to low voltage

In this project, we are working with AC mains voltage, which is typically 220V. These high voltage levels are dangerous and far exceed the operating limits of the Arduino. Arduino boards are designed to operate at low voltages, typically between 5V and 12V, and cannot directly handle high-voltage AC signals. Therefore, it is crucial to step down the high AC voltage to a safe, low voltage before interfacing it with the Arduino. This is achieved through a voltage transformer and voltage divider circuits:

The voltage transformer steps down the high-voltage AC signal to a much lower AC voltage. The stepped-down voltage is then further conditioned using resistors and capacitors to bring it within the safe operating range of the Arduino's analog input pins.

Without this conversion, directly applying high voltage to the Arduino would result in severe damage to the microcontroller and could potentially cause safety hazards. Additionally, Arduino's analog inputs are designed to measure low-voltage signals (in the 0-5V range), so proper voltage conversion ensures that the system can safely read and process the voltage data for energy monitoring.

By stepping down the voltage, we ensure:

Safety for both the user and the components.

Accurate measurement of the voltage and current for calculating power consumption.

5. Building the Physical Smart Energy Meter-

Challenges Encountered in Building the Physical Smart Energy Meter

1. **Hardware Challenges:**
    - **Wiring and Connections:** Issues with ensuring proper connections between the ACS712, Arduino, and LCD. Loose connections or incorrect wiring caused unreliable readings initially.
    - **I2C Communication:** Configuring the I2C LCD required debugging to ensure proper data transmission, especially addressing conflicts with other I2C devices on the same bus.
    - **Sensor Placement:** The ACS712 required careful placement to avoid interference from external magnetic fields, which affected measurement accuracy.
2. **Software Challenges:**
    - **Code Calibration:** Calibrating the ACS712 output to map its analog voltage to accurate current values took significant effort due to variations in the sensor's sensitivity.
    - **Display Issues:** Ensuring the data was displayed correctly on the LCD and serial monitor required fine-tuning the code, especially formatting the output for readability.
    - **Noise Filtering:** Addressing noise in the sensor's output to ensure stable and accurate current readings was a challenge.
3. **Time Constraints:**
    - Balancing debugging efforts and time available for the project's completion proved challenging.
4. **Learning Curve:**
    - Understanding how the ACS712 sensor works and integrating the I2C protocol with the Arduino required research and trial-and-error.

5.1 Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Constants for AC current measurement
const int CurrentAnalogInputPin = A1;   // Pin connected to ACS712 sensor
const float mVperAmpValue = 185.0;      // Sensitivity for ACS712-5A (adjust as needed)
const float voltageAC = 230.0;          // Assume fixed voltage (230V AC)
const int decimalPrecision = 2;         // Decimal precision for printing

// Variables for current measurement
float offsetSampleRead = 0;
float currentSampleRead = 0;
float currentLastSample = 0;
float currentSampleSum = 0;
float currentSampleCount = 0;
float currentMean;
float RMSCurrentMean;
float adjustRMSCurrentMean;
float FinalRMSCurrent;
float power;  // To store calculated power

// Offset variables to remove DC bias from AC readings
int OffsetRead = 0;
float currentOffset1 = 0;
float currentOffset2 = 0;
float offsetSampleSum = 0;
float offsetCurrentMean = 0;
float offsetLastSample = 0;
float offsetSampleCount = 0;

// Initialize LCD with I2C address 0x27
LiquidCrystal_I2C lcd(0x27, 16, 2); // 16x2 LCD

void setup() {
  Serial.begin(9600);  // Start serial communication
  delay(1000);         // Allow sensor to stabilize

  // Initialize the LCD
  lcd.init();
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("Initializing...");
  delay(2000); // Display message for 2 seconds
  lcd.clear();
}

void loop() {
  // AC Current Measurement
  if (millis() >= currentLastSample + 1) {
    // Remove DC bias (midpoint of 512 for 10-bit ADC)
    offsetSampleRead = analogRead(CurrentAnalogInputPin) - 512;
    offsetSampleSum += offsetSampleRead;

    // Read current sample and add offset
    currentSampleRead = analogRead(CurrentAnalogInputPin) - 512 + currentOffset1;
    currentSampleSum += sq(currentSampleRead);
    currentSampleCount++;

    // Update last sample time
    currentLastSample = millis();
  }

  // Calculate RMS Current after 1000 samples
  if (currentSampleCount == 1000) {
    offsetCurrentMean = offsetSampleSum / currentSampleCount;

    // Calculate RMS (Root Mean Square) current
    currentMean = currentSampleSum / currentSampleCount;
    RMSCurrentMean = sqrt(currentMean);

    // Adjust for any secondary offset if needed
    adjustRMSCurrentMean = RMSCurrentMean + currentOffset2;

    // Calculate final RMS current in amps
    FinalRMSCurrent = (((adjustRMSCurrentMean / 1024) * 5000) / mVperAmpValue);

    // Calculate apparent power (Power = Voltage * RMS Current)
    power = voltageAC * FinalRMSCurrent;

    // Print the RMS current and power to LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("RMS Curr: ");
    lcd.print(FinalRMSCurrent, decimalPrecision);
```

```
  lcd.print(" A");

  lcd.setCursor(0, 1);
  lcd.print("Power: ");
  lcd.print(power, decimalPrecision);
  lcd.print(" W");

  // Print to Serial Monitor for debugging
  Serial.print("Current RMS value: ");
  Serial.print(FinalRMSCurrent, decimalPrecision);
  Serial.print(" A, Power: ");
  Serial.print(power, decimalPrecision);
  Serial.println(" W");

  // Reset sums and counts
  offsetSampleSum = 0;
  currentSampleSum = 0;
  currentSampleCount = 0;
}

// Offset AC Current Calculation
if (OffsetRead == 1) {
  currentOffset1 = 0;

  if (millis() >= offsetLastSample + 1) {
    offsetSampleCount++;
    offsetLastSample = millis();
  }

  if (offsetSampleCount == 1500) {
    currentOffset1 = -offsetCurrentMean;
    OffsetRead = 2;
    offsetSampleCount = 0;
  }
}

if (OffsetRead == 2) {
  currentOffset2 = 0;

  if (millis() >= offsetLastSample + 1) {
    offsetSampleCount++;
    offsetLastSample = millis();
  }
```

```
    if (offsetSampleCount == 2500) {
      currentOffset2 = -RMSCurrentMean;
      OffsetRead = 0;
      offsetSampleCount = 0;
    }
  }
}
```

This code measures AC current using the **ACS712 current sensor** connected to an Arduino. It calculates the **Root Mean Square (RMS)** of the current, removes DC offsets, and computes apparent power based on a fixed AC voltage of 230V. The results are displayed on an LCD via I2C and printed to the Serial Monitor.

Conceptual Flow

---

1. **Initialization**:
   - Sets up the Arduino, Serial Monitor, and LCD.
   - Displays an initialization message on the LCD.
2. **Data Collection**:
   - Reads analog values from the ACS712 sensor.
   - Removes the DC offset (midpoint value of 512 for 0A).
   - Squares and accumulates readings for RMS calculation.
3. **RMS Current Calculation**:
   - After 1000 samples, computes the mean of squared values and calculates the RMS.
   - Converts the RMS value to actual current in amps.
4. **Power Calculation**:
   - Multiplies the RMS current by the fixed AC voltage (230V) to calculate power.
5. **Output**:
   - Displays the RMS current and power on the LCD and Serial Monitor.

Key Concepts in the Code

---

1. Analog-to-Digital Conversion (ADC)

The ACS712 sensor outputs an analog voltage proportional to the current flowing through it. The Arduino's **ADC** converts this analog signal into a digital value (0–1023 for a 10-bit ADC). This is used for computation in the code.

- **How ADC works in Arduino**:
    - The input voltage (0–5V for most Arduinos) is divided into 1024 steps:

$$\text{Digital Value} = \frac{\text{Input Voltage} \times 1024}{\text{Reference Voltage (5V)}}$$

    - This provides a digital representation of the sensor's analog output.

2. AC Signal and DC Offset

The ACS712 sensor's output includes:

- A **DC Offset**: At 0A, the sensor outputs a constant voltage (e.g., 2.5V for ACS712). This is the "midpoint."
- An **AC Component**: This represents the current fluctuations.

To focus on the AC signal (current), the DC offset is removed by subtracting the midpoint value (512 for a 10-bit ADC at 2.5V).

3. Root Mean Square (RMS) Current Measurement

AC current is not constant—it varies sinusoidally. To measure its effective value, the **RMS (Root Mean Square)** technique is used. RMS gives the equivalent value of a DC current that would produce the same heating effect.

- **RMS Formula**:

$$I_{\text{RMS}} = \sqrt{\frac{\sum(\text{current samples})^2}{\text{number of samples}}}$$

- **Steps in the Code**:
    1. Each current sample is squared and accumulated (`currentSampleSum`).
    2. The mean is calculated by dividing the sum by the number of samples.
    3. The square root of the mean gives the RMS value.

4. Apparent Power Calculation

The code calculates the **apparent power** in watts (W). Apparent power is the product of RMS current and voltage:

$$P = V_{\text{AC}} \times I_{\text{RMS}}$$

- **Voltage (V)**: Assumed to be constant at 230V AC.
- **Current (I)**: Measured using the RMS technique.

This calculation is useful for understanding the energy usage of connected appliances.

5. Sensor Sensitivity (mV per Amp)

The ACS712 outputs a voltage proportional to the current, based on its sensitivity. For example:

- ACS712-5A: 185 mV per amp.
- ACS712-20A: 100 mV per amp.
- ACS712-30A: 66 mV per amp.

In the code:

$$I_{\text{RMS}} = \frac{V_{\text{RMS}}}{\text{mV per Amp}}$$

- The sensor's output voltage is converted to current using the sensitivity value (`mVperAmpValue`).

6. Sampling and Data Averaging

The code samples the current signal 1000 times in a loop to ensure accuracy:

- By taking multiple samples and averaging them, the effect of noise or transient fluctuations is minimized.

Why 1000 samples?

- AC signals typically have a frequency of 50–60 Hz. At 50 Hz, one cycle lasts 20 ms. Sampling for 1000 ms (1 second) ensures multiple full cycles are captured for an accurate RMS calculation.

7. Removing DC Bias (Offset Calibration)

The DC offset in the ACS712 output needs correction. The code does this in two stages:

1. **Initial Offset Removal**:
    - Subtracts the midpoint value (512) directly.
2. **Dynamic Adjustment**:
    - Fine-tunes the offset based on calculated means.

This ensures the AC signal is centered around zero for accurate RMS calculations.

8. Timing and Sampling Rate

The code uses **millis()** to ensure samples are taken at regular intervals (1 ms per sample). This avoids inconsistent sampling rates, which can distort RMS calculations.

9. I2C Communication

The code interfaces with the LCD using the **I2C protocol**:

- **SCL (Serial Clock Line)**: Synchronizes communication between devices.
- **SDA (Serial Data Line)**: Transmits data.
- **Address (0x27)**: Identifies the specific I2C device.

I2C reduces the number of wires needed (compared to parallel communication) and simplifies connections.

10. Power Display on LCD

The code uses the `LiquidCrystal_I2C` library to print the RMS current and power on the LCD. This allows real-time monitoring of current and power without relying on a serial monitor.

Formulas Used

- **ADC to Voltage Conversion**:

$$V_{\text{out}}(\text{mV}) = \frac{\text{ADC Reading} \times 5000}{1024}$$

  - Converts the ADC reading (0–1023) to a voltage in millivolts.

- **Current in Amps**:

$$I_{\text{RMS}} = \frac{V_{\text{RMS}}}{\text{mV per Amp}}$$

  - Converts the RMS voltage output of the sensor to RMS current.

- **Apparent Power**:

$$P = V_{\text{AC}} \times I_{\text{RMS}}$$

- **Root Mean Square (RMS)**:

$$I_{\text{RMS}} = \sqrt{\frac{\sum(\text{sample values})^2}{\text{number of samples}}}$$

  - Calculates the effective AC current.

Explanation of Variables

Constants

```
cpp
Copy code
const int CurrentAnalogInputPin = A1;
const float mVperAmpValue = 185.0;
const float voltageAC = 230.0;
const int decimalPrecision = 2;
```

- **`CurrentAnalogInputPin`**: Defines the analog pin connected to the ACS712 sensor's output.
- **`mVperAmpValue`**: The sensitivity of the ACS712 sensor model. For example:
  - ACS712-5A: 185 mV/A
  - ACS712-20A: 100 mV/A
  - ACS712-30A: 66 mV/A
- **`voltageAC`**: Assumed constant voltage of 230V AC.
- **`decimalPrecision`**: Number of decimal places to display in outputs.

Offset Variables

```
cpp
Copy code
float offsetSampleRead = 0;
float offsetSampleSum = 0;
float offsetCurrentMean = 0;
float offsetLastSample = 0;
float offsetSampleCount = 0;
```

- **`offsetSampleRead`**: Raw sensor reading with DC offset removed.
- **`offsetSampleSum`**: Sum of offset readings for calculating the average.
- **`offsetCurrentMean`**: Average offset value used to remove the DC bias.

Current Measurement Variables

```
cpp
Copy code
float currentSampleRead = 0;
float currentLastSample = 0;
float currentSampleSum = 0;
float currentSampleCount = 0;
float currentMean;
float RMSCurrentMean;
float adjustRMSCurrentMean;
float FinalRMSCurrent;
float power;
```

- **`currentSampleRead`**: Instantaneous current value after offset removal.

- **`currentSampleSum`**: Sum of squared current readings for RMS calculation.
- **`currentMean`**: Mean of the squared current values.
- **`RMSCurrentMean`**: Square root of `currentMean`, representing the RMS value.
- **`FinalRMSCurrent`**: Converted RMS current in amperes.
- **`power`**: Apparent power calculated using:

$$\text{Power (W)} = \text{Voltage (V)} \times \text{RMS Current (A)}$$

6.Methodology

---

1. System Design and Planning

- **Objective Identification**: Clearly defined objectives, such as monitoring energy usage, ensuring real-time data display, and exploring potential for IoT integration.
- **Component Selection**:
  - Arduino Uno as the microcontroller for its versatility, cost-effectiveness, and ease of integration.
  - ACS712 current sensor for precise and isolated current measurement using the Hall effect.
  - 16x2 LCD with I2C communication for simple real-time data display.
  - Step-down transformer and voltage divider for voltage measurement in simulations.
- **Software Tool Selection**: Proteus for circuit simulation and Arduino IDE for software development.

2. Circuit Simulation and Validation

- **Proteus Simulation**:
  - The circuit was simulated to validate the functionality of the ACS712 sensor, Arduino Uno, and LCD display.
  - Simulation included modeling current and voltage measurements and displaying the calculated power on a virtual LCD.
  - Multiple scenarios were tested to debug potential errors and ensure stability.
- **Behavioral Testing**:
  - Analyzed system behavior by simulating varying load conditions and verifying accurate current readings and power calculations.

3. Hardware Implementation

- **Assembly**:
  - Physical integration of Arduino, ACS712, and LCD on a breadboard.
  - Proper wiring ensured connections matched the simulation for consistent results.
  - Used pull-up resistors and capacitors where necessary for stable data flow.
- **Current Measurement**:
  - ACS712 sensor was configured to measure AC current non-invasively, minimizing electrical hazards.

    - Offset calibration was performed to account for the sensor's DC bias and ensure accurate RMS readings.
- **Display Integration**:
    - Connected the LCD to the Arduino via I2C for efficient communication.
    - Programmed the LCD to display current, power consumption, and additional parameters in real time.

4. Software Development

- **Programming**:
    - Developed Arduino code for reading analog signals from the ACS712, calculating RMS current, and estimating power consumption based on a fixed voltage (230V AC).
    - Implemented offset correction and noise filtering techniques for stable readings.
    - Used the LiquidCrystal_I2C library to control the LCD display.
- **Testing**:
    - Debugged software using the serial monitor to verify outputs.
    - Refined algorithms to improve accuracy and responsiveness of the system.

5. Validation and Testing

- **Testing the Physical Model**:
    - Compared outputs from the hardware setup with simulation results to validate consistency.
    - Ensured stable readings across varying current levels by testing under different load conditions.
- **Iterative Improvements**:
    - Addressed issues like noise interference and I2C conflicts to refine system performance.

7.Planned Enhancements and Future Developments

---

1. IoT Integration

- **Remote Monitoring**:
    - Integrate Wi-Fi modules like ESP8266 or GSM modules for wireless data transmission.
    - Users can view real-time energy data on mobile or web dashboards.
- **Cloud Analytics**:
    - Store data on cloud platforms for long-term analysis.
    - Enable advanced features like trend visualization, anomaly detection, and predictive energy usage reports.

2. Advanced Measurement Capabilities

- **Voltage Sensing**:
    - Add voltage sensors to measure real-time voltage in conjunction with current for more accurate power calculations.
- **Power Factor Calculation**:

- Extend the system to calculate power factor for monitoring reactive and apparent power.
- Beneficial for industrial users to optimize energy efficiency.

3. Smart Grid Compatibility

- Adapt the system to work with smart grids by enabling two-way communication with utility companies.
- Allow dynamic load adjustment and participation in demand-response programs to reduce overall grid stress.

4. Energy Conservation Features

- **Threshold Alerts**:
  - Set thresholds for energy consumption and trigger alerts via SMS or mobile notifications if exceeded.
- **Energy Budgeting**:
  - Provide tools for users to set energy usage targets and monitor progress.

5. Enhanced User Interface

- Replace the 16x2 LCD with advanced touch displays or integrate with mobile applications for an interactive user experience.
- Enable customization options for users to define what parameters are displayed.

6. Automation and Control

- Incorporate relays or smart plugs to allow the system to automatically turn off devices when idle or overconsuming energy.
- Provide users with remote control capabilities through a mobile app.

7. Cost and Efficiency Optimization

- **Component Upgrades**:
  - Use more efficient sensors or microcontrollers, like ESP32, to reduce power consumption and cost.
- **Scalability**:
  - Develop a modular version that can be expanded for multi-appliance monitoring or industrial use cases.

8. Environmental Impact

- Explore sustainable materials and manufacturing methods for physical devices.
- Highlight the role of the system in reducing carbon footprints by promoting energy-saving practices.

8. References -

1. IoT Based Smart Energy Meter

Birendrakumar Sahani 1, Tejashree Ravi 2, Akibjaved Tamboli 3, Ranjeet Pisal 4

2. FARADAY: SYNTHETIC SMART M ETER G ENERATOR FOR THE SMART GRID
Sheng Chai & Gus Chadney

3. AutodeskInstructables, Simple Arduino Energy Meter AutodeskInstsructables

https://www.instructables.com/Simple-Arduino-Home-Energy-Meter/

9. Conclusion -

The development of the Smart Energy Meter marks a significant step towards efficient energy management and conservation. By leveraging an Arduino-based platform, the project successfully demonstrates the capability to monitor electrical parameters like current and calculate power consumption in real time. The implementation of the ACS712 sensor, along with an LCD display, ensures that energy usage data is readily accessible to users, empowering them to make informed decisions about their consumption patterns.

This project not only addresses the immediate need for local energy monitoring but also lays the foundation for future enhancements, such as IoT integration for remote data transmission and advanced analytics. These features will enable greater scalability and applicability in both residential and industrial settings.

The project aligns with global sustainability goals by promoting responsible energy usage and reducing wastage. While the current system achieves its primary objectives, it offers immense potential for growth with features like voltage measurement, smart grid compatibility, and predictive analytics.

In conclusion, the Smart Energy Meter serves as a practical and scalable solution for modern energy challenges, demonstrating how technology can be harnessed to foster sustainable practices and create a more energy-efficient future.