



Professional Engineering Practice and Industrial Management

2nd Year, 2nd Semester

Final Report

SMART ENERGY MANAGEMENT SYSTEM

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

18/06/2024





Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

Project Details

Project Title	SMART ENERGY MANAGEMENT SYSTEM
Project ID	PEP_33

Group Members

Reg. No	Name	Signature
IT22115584	W K T Duranga (Group leader)	
IT22549518	P R M Chaminda	
IT22555830	N K D Narangaspitiya	
IT22052810	K L H D Jayalath	

Abstract

The current energy management system is inefficient due to its inability to monitor energy consumption in real-time, leading to excessive and unnecessary energy use. This is particularly problematic with lighting and cooling systems, where energy wastage is common due to lack of specific control mechanisms. To address these issues, we propose the installation of a smart energy monitoring system aimed at optimizing energy consumption and enhancing sustainability. The solution encompasses several innovative components. Firstly, a voice-controlled light system allows users to turn lights on or off through voice commands, ensuring lights are only used when needed. Secondly, an Arduino-based electric meter provides real-time monitoring of energy usage, offering detailed insights into consumption patterns. Thirdly, a smart fan equipped with temperature sensors automatically adjusts its speed based on the ambient temperature, thus optimizing cooling efficiency. Lastly, a motion sensor light system ensures that lights are activated only when movement is detected, significantly reducing energy wastage in unoccupied spaces. Together, these components form a comprehensive smart energy management system that not only reduces unnecessary energy use but also supports better energy management practices. This integrated approach promises substantial improvements in energy efficiency, contributing to sustainability and cost savings.

Keywords - Energy management, PIR sensor, Energy consumption, Real-time monitoring, Lighting systems, Cooling systems, Smart energy monitoring, Motion sensor light system, Voice-controlled light system

Acknowledgement

We would like to express our sincere gratitude to the Sri Lanka Institute of Information Technology for providing us with the opportunity and resources to undertake this project. We are deeply indebted to our supervisors, Mrs. Narmada Gamage, for her invaluable guidance, support, and encouragement throughout the development of this smart energy monitoring system. Her insights and expertise have been instrumental in shaping the direction and success of our work.

We also extend our heartfelt thanks to our group members, Thashmila, Kithmi, Mahesh, and Lakshi, for their dedicated efforts, collaboration, and unwavering commitment to the project. Each member's unique contributions have been vital in bringing this project to fruition.

Finally, we would like to thank everyone who supported us, directly or indirectly, throughout this journey. Your encouragement and assistance have been greatly appreciated and have made a significant impact on the successful completion of this project. Thank you all.

Table of Contents

Declaration.....	ii
Abstract.....	iii
Acknowledgement	iv
Table of Contents	v
List of Figures.....	i
List of Tables	vii
List of Acronyms and Abbreviations	vii
1. Introduction.....	1
1.1 Problem Statement	1
1.2 Product Scope	2
1.3 Project Report Structure.....	3
2. Methodology	4
2.1 Requirements and Analysis.....	4
2.1.1 Functional requirements of the project.....	4
2.1.2 Non-Functional requirements of the project.....	5
2.2 Design	8
2.2.1 High level architecture diagram.....	8
2.2.2 Interfaces.....	9
2.2.2.1 User Interfaces.....	9
2.2.2.2 Hardware Interfaces.....	11
2.2.3 Circuit design diagrams.....	12
2.3 Implementation	14
2.3.1 Circuit diagrams.....	14
Function 1 - Voice-Controlled Light System.....	14
Function 2 - Arduino-Based Electric Meter.....	15
Function 3 - Smart Fan with Temperature Sensing.....	16
Function 4 - Motion Sensor Light System.....	17
2.3.2 System circuit diagram.....	18
2.4 Testing.....	19
3. Evaluation.....	23
3.1 Assessment of the Project results.....	23
3.1.1 Overall Assessment.....	24
3.2 Lessons Learned.....	25
3.3 Future Work	27
4. Conclusion	29
4.1 Realization of Objectives.....	29
4.2 Future Work and Enhancements.....	30
4.3 Addressing Weaknesses and Limitations.....	30
4.4 Benefits to the Client Organization.....	30
4.5 Final Thoughts.....	30
5. References.....	31
Appendix A: Selected Code Listings	32

List of Figures

Figure 2.1.1-Use case diagram.....	7
Figure 1.2.1.1- High-level architecture diagram	8
Figure 2.2.2.1.1-The Arduino Blue Control app user interface	9
Figure 2.3.2.1.2- Blynk app user interface.....	10
Figure 1.2.3.1-Voice-Controlled Light System circuit.....	12
Figure 1.2.3.2-Arduino-based electric meter circuit.....	12
Figure 1.2.3.3- Smart Fan with Temperature Sensing circuit.....	13
Figure 1.2.3.4- Motion Sensor Light System circuit.....	13
Figure 1.3.1.1 - Circuit diagram implementation of Voice-Controlled Light System.....	14
Figure 1.3.1.2 - Circuit diagram implementation of Arduino-based electric meter.....	15
Figure 1.3.1.3 - Circuit diagram implementation of Smart fan with temperature sensing circuit.....	16
Figure 1.3.1.4 - Circuit diagram implementation of Motion sensor light system circuit.....	17
Figure 1.3.2.1 – Final system circuit diagram.....	18

List of Tables

Table 1.1: Project Report Structure table.....	3
Table 2.1.1.1 – Function 1 -Voice-Controlled Light System	4
Table 2.1.1.2 – Function 2 - Arduino-Based Electric Meter	4
Table 2.1.1.3 – Function 3 - Smart Fan with Temperature Sensing	5
Table 2.1.1.4 – Function 4 - Motion Sensor Light System.....	5
Table 1.4.1 - This test scenario confirms that the Voice-Controlled Light System functions as intended by responding appropriately to valid voice commands and ignoring invalid commands.....	19
Table 1.4.2 - test scenario confirms that the Arduino-Based Electric Meter accurately measures voltage and current, providing information on energy consumption correctly.....	20
Table 1.4.3 - This test scenario confirms that the Smart Fan with Temperature Sensing system operates correctly across different temperature ranges.....	21
Table 2.4.4 - This test scenario confirms that the Motion Sensor Light System operates correctly by responding to motion detection signals.....	22

List of Acronyms and Abbreviations

AI-driven: Advance Intelligence driven

DBMS : Database Management System

1. Introduction

1.1 Problem Statement

The current energy management systems in many buildings are inefficient, primarily due to the lack of real-time monitoring and control. This inefficiency leads to excessive energy consumption, increased utility bills, and a larger environmental footprint. Traditional systems often result in lights and cooling devices being left on unnecessarily, contributing to significant energy wastage. To address these challenges, we propose the development of a smart energy monitoring system focused on optimizing energy use in lighting and cooling systems.

This project integrates several innovative components to form a comprehensive solution. The voice-controlled light system allows users to manage lighting through simple voice commands, ensuring lights are used only when needed. The Arduino-based electric meter provides real-time monitoring and detailed insights into energy consumption patterns, enabling users to make informed decisions about their energy use. A smart fan equipped with temperature sensors automatically adjusts its speed based on the ambient temperature, optimizing cooling efficiency and reducing energy waste. Additionally, the motion sensor light system ensures that lights are activated only when movement is detected, significantly minimizing energy usage in unoccupied spaces.

By combining these technologies, the smart energy management system offers a holistic approach to energy optimization. It enhances energy efficiency, reduces unnecessary consumption, and supports sustainable energy practices. The real-time data provided by the system allows for better energy management and informed decision-making, ultimately leading to cost savings and a reduced environmental impact. This project demonstrates the potential for intelligent, automated solutions to significantly improve energy management in buildings, paving the way for more sustainable and efficient energy use in the future.

1.2 Product Scope

Overall Scope: The smart energy monitoring system aims to enhance energy efficiency in buildings by integrating real-time monitoring and automated control of lighting and cooling systems. It combines voice control, an Arduino-based electric meter, a smart temperature-sensing fan, and motion sensor lights to reduce energy consumption and support sustainable energy management practices.

In scope: The in-scope elements include integrating voice control for lighting, real-time energy consumption monitoring via an Arduino-based electric meter, automatic fan speed adjustment based on temperature, and motion-activated lighting. These components aim to optimize energy use, enhance efficiency, and provide users with actionable insights into their energy consumption

Out scope: Out of scope elements include integration with comprehensive smart home systems, advanced AI-driven analytics for predictive maintenance, control of non-lighting and non-cooling appliances, extensive retrofitting of existing infrastructure, and development of custom hardware beyond the Arduino-based meter and basic sensors. The focus remains on core lighting and cooling optimizations.

1.3 Project Report Structure

Table 1.1: Project Report Structure table

Chapter Number	Organization
Chapter 1	Introduction: Introduction to the project, including Problem Statement, Product Scope, and Project Report Structure (we have already discussed this chapter.)
Chapter 2	Methodology: Requirements and Analysis- Detailed and specific project requirements, including use case and activity diagrams illustrating system requirements. Design- High-level architecture diagram, Explanation of software design using UML diagrams (e.g., class diagram, sequence diagram, state charts, activity diagrams), Database design explained using ER diagrams, User interface designs. Implementation- Comprehensive explanation of major module structures, Explanation of any reusable code and development tools used, Description of the chosen DBMS, implementation languages, and key algorithms (with actual code in appendices). Testing- Description of the test plan, Evidence of system-wide testing, Inclusion of at least 16 test cases (2 per function/module).
Chapter 3	Conclusion: This structured approach ensures a comprehensive evaluation of project outcomes, facilitates knowledge transfer through lessons learned, and outlines actionable recommendations for future enhancements and advancements in smart energy management technology.
Chapter 4	References: List of references in IEEE referencing style. <ul style="list-style-type: none">• Appendix A: Selected Code Listings<ul style="list-style-type: none">○ Code of special algorithms implemented (details without copying all code).

2. Methodology

2.1 Requirements and Analysis

2.1.1 Functional requirements of the project

The functional requirements for the product are organized based on system features, illustrating the major services provided. This structure allows for a clear understanding of use cases, user classes, and functional hierarchy, ensuring logical organization and comprehensive coverage of system capabilities.

Table 2.1.1.1 – Function 1 -Voice-Controlled Light System

F1	Voice-Controlled Light System
Input	Voice commands from users to turn lights on/off , Bluetooth signals from the Arduino Blue Control app
Process	gathering requirements for related sensors and designing, coding, and hardware part
Output	when giving a voice command, the bulb is turned on or stays turned off. It is fully automating the system.
Definition	a voice-controlled light system to reduce electricity costs by controlling light bulbs as needed.

Table 2.2.1.2 – Function 2 - Arduino-Based Electric Meter

F2	Arduino-Based Electric Meter
Input	Current and voltage readings from current and voltage sensors
Process	Designing and connecting all sensors and equipment
Output	Per unit cost, efficient energy process, and other important details are displayed on the LED panel.
Definition	The unit cost related to the equipment used can be confirmed efficiently and effectively.

Table 2.3.1.3 – Function 3 - Smart Fan with Temperature Sensing

F3	Smart Fan with Temperature Sensing
Input	Temperature readings from temperature sensors
Process	Designing a circuit diagram using a temperature sensor, an Arduino board, and other necessary things.
Output	Fan speed automatically changes to slow, medium, or high levels according to the temperature.
Definition	Fan speed is controlled according to the current temperature format.

Table 2.4.1.4 – Function 4 - Motion Sensor Light System

F4	Motion Sensor Light System
Input	Motion detection signals from motion sensors
Process	connecting to the motion sensor, Arduino Uno board, and other equipment
Output	If someone is exposed to the sensor, it is turned on; if it is not, the bulb is off.
Definition	The bulb only lights up according to motion.

2.1.2 Non-Functional requirements of the project

Performance

- **Response Time:** The system should respond to voice commands and motion detection within 1 second.
- **Fan Speed Adjustment:** The fan should adjust its speed within 2 seconds of a temperature change.
- **Energy Monitoring Update:** Energy consumption data should be updated on the LED panel every 5 seconds.

Reliability

- **Uptime:** The system should be operational 99.9% of the time, ensuring minimal downtime.
- **Error Handling:** The system should handle sensor failures gracefully, providing alerts if a sensor malfunctions.

Scalability

- **Device Integration:** The system should support the integration of additional smart devices (e.g., more lights, fans, and sensors) without significant reconfiguration.
- **User Load:** The system should efficiently handle multiple user inputs and commands simultaneously.

Security

- **Data Encryption:** All data transmitted between sensors, the Arduino board, and the LED panel should be encrypted to prevent unauthorized access.
- **Access Control:** Only authorized users should be able to send commands to the system.

Usability

- **User Interface:** The Arduino Blue Control app and LED panel interface should be intuitive and easy to navigate for users of all technical levels.
- **Voice Recognition Accuracy:** The voice control system should accurately recognize and execute commands with at least 95% accuracy.

Maintainability

- **Modular Design:** The system should be designed in a modular way, allowing easy updates and replacements of individual components (sensors, Arduino board, etc.).
- **Documentation:** Comprehensive documentation for the system setup, operation, and troubleshooting.

Compatibility

- **Cross-Platform Support:** The Arduino Blue Control app compatible with both Android and iOS devices.
- **Standards Compliance:** The system should comply with relevant electrical and communication standards.

Energy Efficiency

- **Low Power Consumption:** The system should be designed to minimize power usage, particularly for the sensors and Arduino board.
- **Efficient Energy Processing:** The energy monitoring system should accurately track and optimize energy consumption to reduce costs.

Environmental Conditions

- **Operating Temperature:** The system should function correctly within a temperature range of 0°C to 50°C.

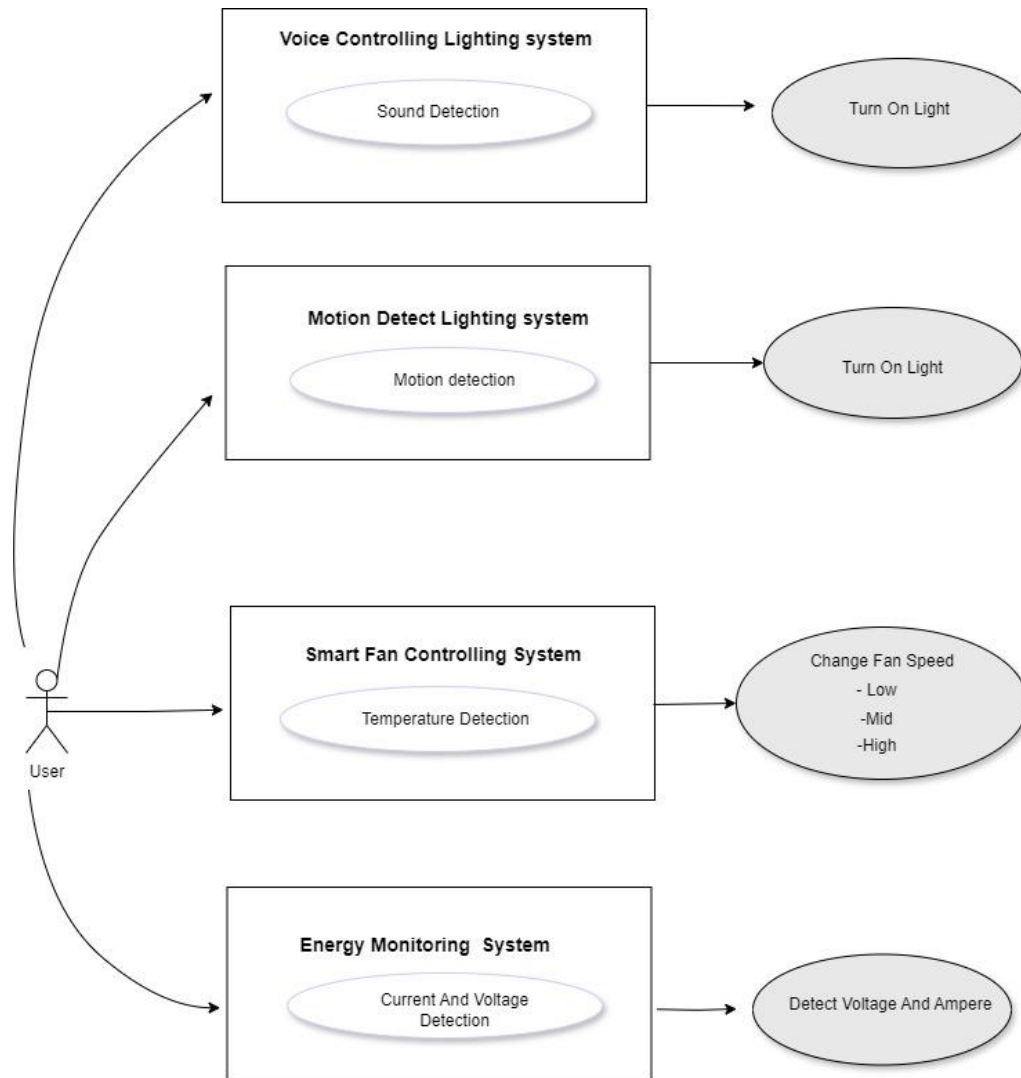


Figure 2.1.1-Use case diagram

The provided use case diagram illustrates a comprehensive smart home system with various functionalities for controlling lighting, fan speed, and energy monitoring. The diagram visually represents these interactions, showing how the smart home system integrates various detection mechanisms (sound, motion, temperature, and electrical parameters) to automate and optimize home utilities for enhanced convenience and energy efficiency.

2.2 Design

2.2.1 High level architecture diagram

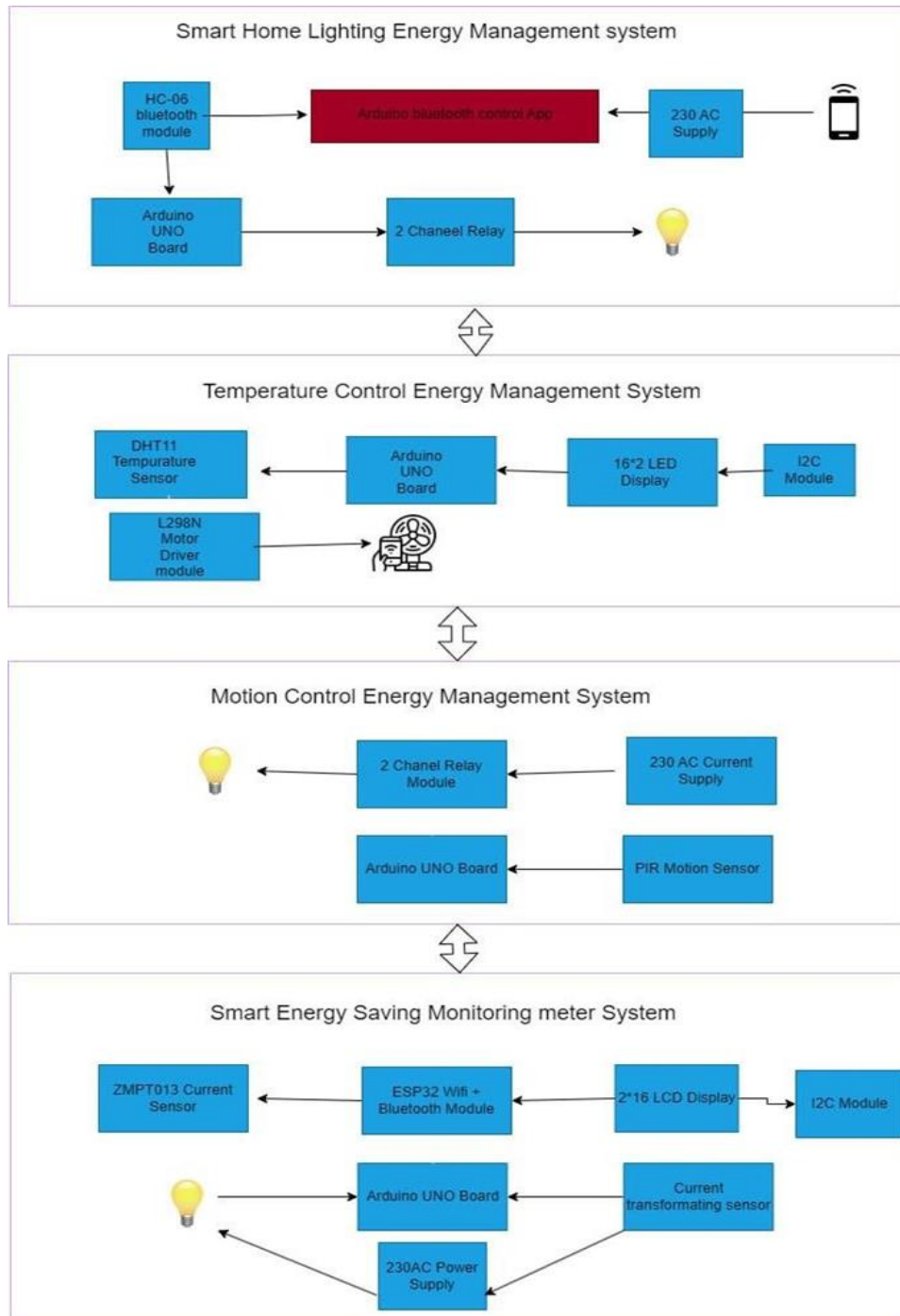


Figure 2.2.1.1- High-level architecture diagram

The high-level architecture diagram showcases a comprehensive smart home energy management system with four interconnected subsystems: voice-controlled lighting, temperature-based fan control, motion-activated lighting, and energy consumption monitoring. Each subsystem utilizes an Arduino UNO board alongside various sensors and modules to automate home utilities. The diagram illustrates how these components interact to enhance convenience, efficiency, and energy savings within a smart home environment.

2.2.2 Interfaces

2.2.2.1 User Interfaces

The Arduino Blue Control app:

The app facilitates a voice-controlled light system, enabling users to effortlessly control bulbs with voice commands. Integrated with Arduino hardware, users can issue verbal instructions via the Arduino Blue Control app, connecting to the system via Bluetooth. This seamless integration allows for easy management of lighting environments, whether it's turning lights on or off or adjusting illumination levels. With this innovative system, users experience hands-free operation and enhanced convenience in home automation.



Figure 2.2.2.1.1-The Arduino Blue Control app user interface

Blynk app:

The Arduino-Based Electric Meter's user interface uses the Blynk mobile application for intuitive monitoring of electricity usage. It features customizable widgets and real-time readings from sensors connected to the Arduino board. The interface adheres to Blynk's GUI standards, displays real-time readings of current and voltage from sensors connected to the Arduino board, offering standard buttons for navigation and interaction. Error messages are displayed concisely, guiding users to troubleshoot issues. This enhances user engagement and empowers informed decisions about energy usage and cost efficiency.

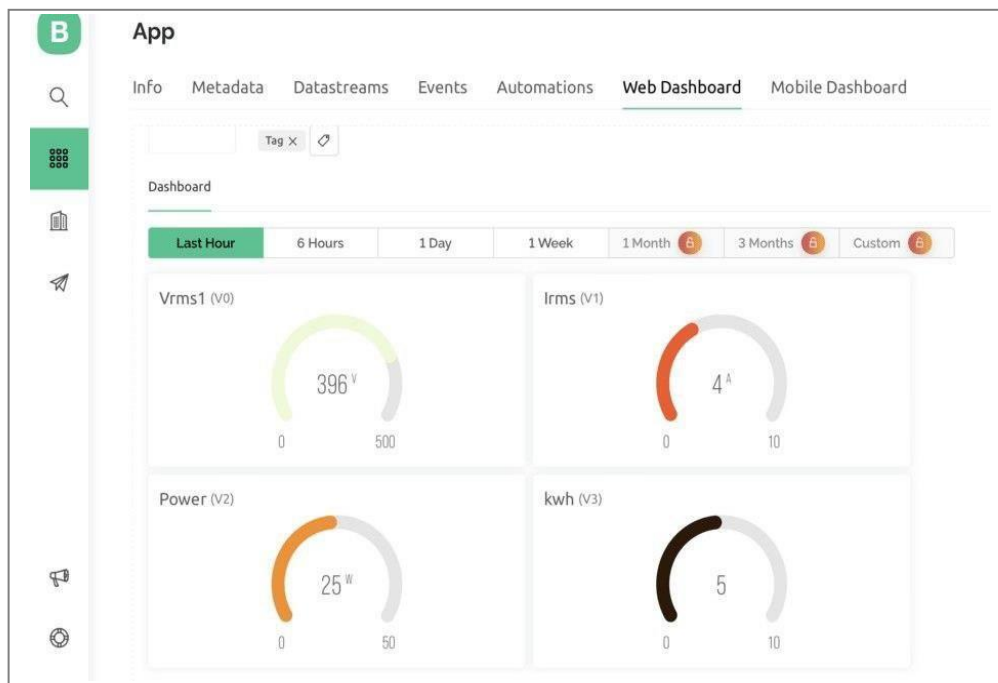


Figure 2.3.2.1.2- Blynk app user interface

2.2.2.2 Hardware Interfaces

For the Smart Fan with Temperature Sensing, the interface between the software and hardware involves communication with temperature sensors and fan control mechanisms. Data interactions entail reading temperature values from the sensor and sending control signals to the fan motor for speed adjustment. The logical characteristics involve interpreting temperature readings, determining appropriate fan speed levels (slow, medium, or high), and displaying them on an LCD display.

Regarding the Arduino-Based Electric Meter, the interface involves communication with current and voltage sensors to measure load parameters. Data interactions entail reading current and voltage values from the sensors and displaying them on an LCD display. The logical characteristics involve processing sensor readings to calculate current consumption and voltage levels.

2.2.3 Circuit design diagrams

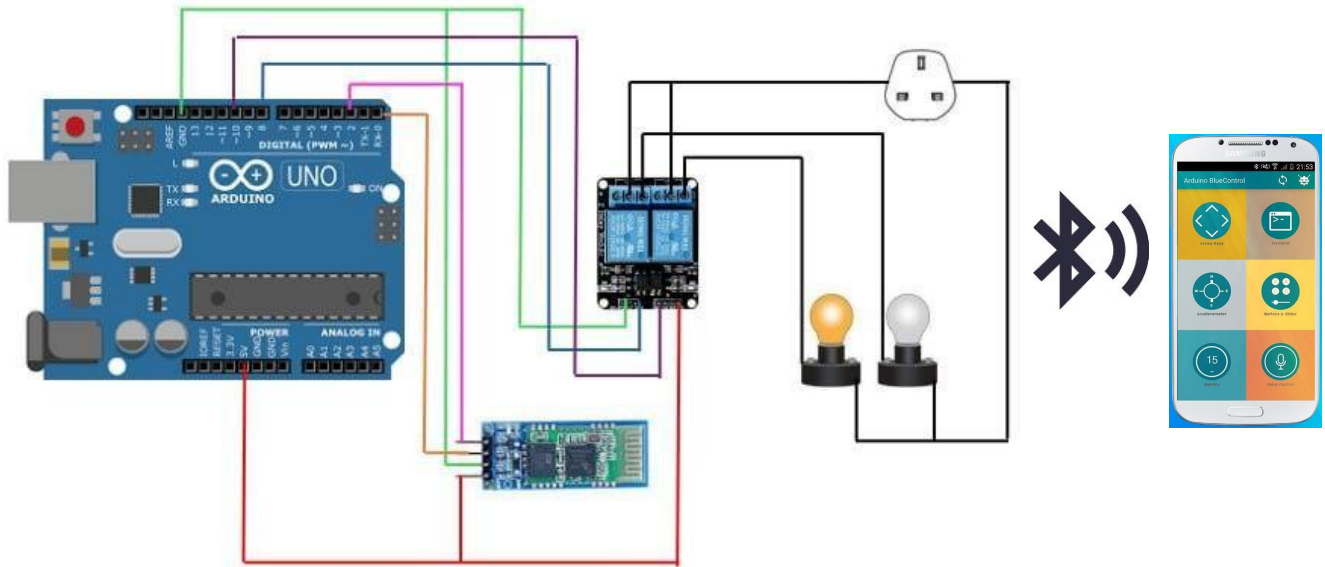


Figure 2.2.3.1-Voice-Controlled Light System circuit

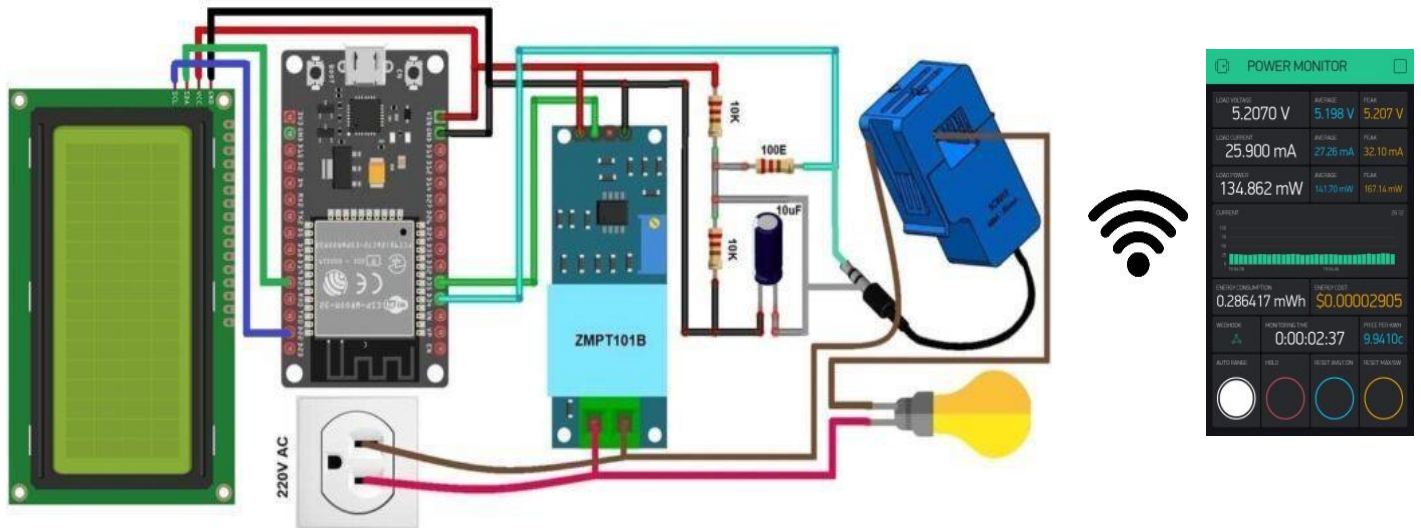


Figure 2.2.3.2-Arduino-based electric meter circuit

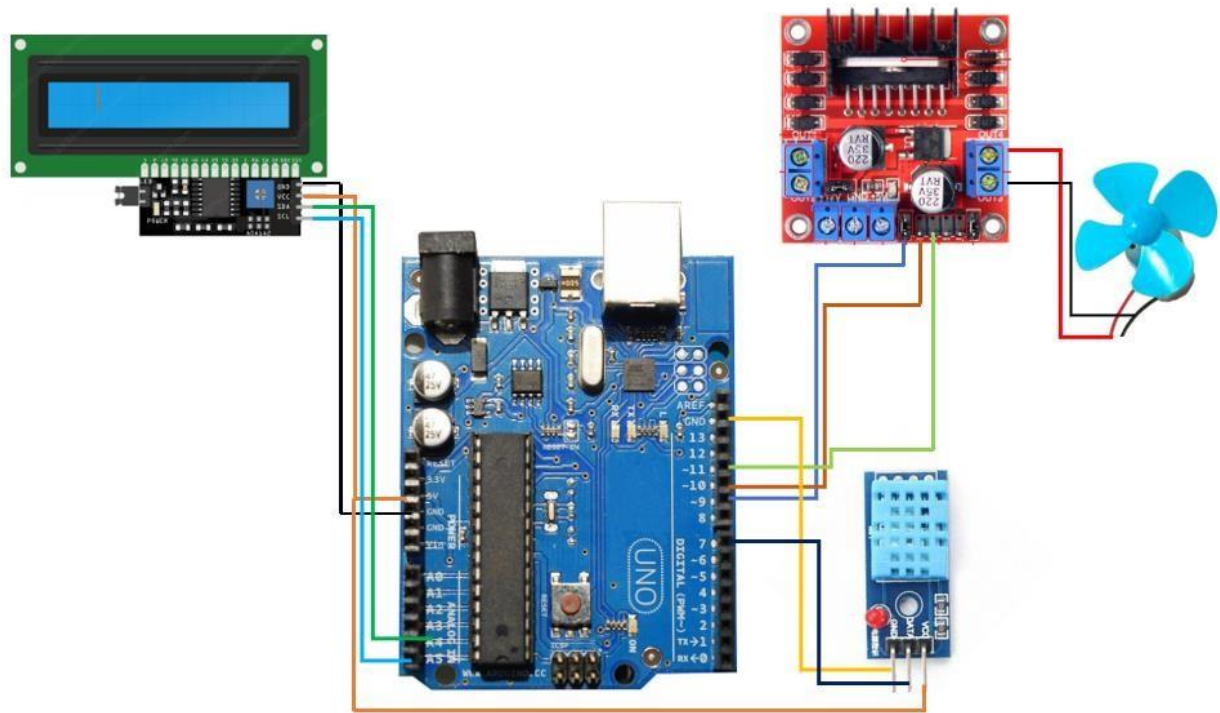


Figure 2.2.3.3- Smart Fan with Temperature Sensing circuit

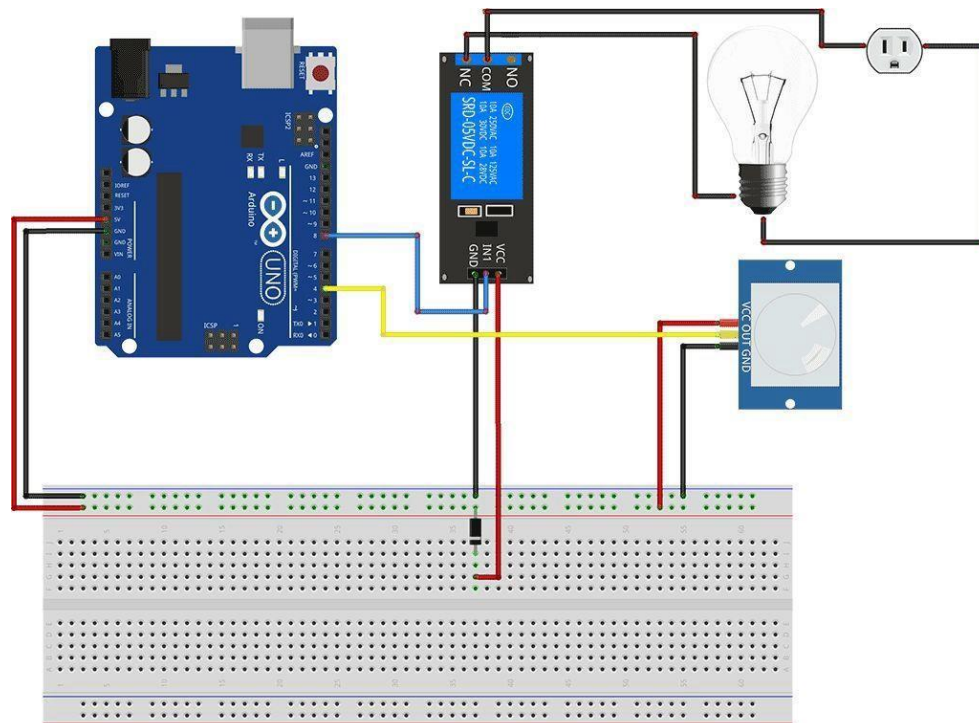


Figure 2.2.3.4- Motion Sensor Light System circuit

2.3 Implementation

2.3.1 Circuit diagrams

I. Function 1 - Voice-Controlled Light System [\[1\]](#)

Arduino UNO Board: The central microcontroller that processes commands received from the Bluetooth module and controls the relays accordingly.

HC-06 Bluetooth Module: Enables wireless communication between the Arduino and the Arduino Blue Control app. **VCC:** Connected to the 5V pin on the Arduino to provide power to the Bluetooth module. **GND:** Connected to the GND pin on the Arduino to complete the circuit. **TX:** Connected to the RX pin on the Arduino (digital pin 0) to send data from the Bluetooth module to the Arduino. **RX:** Connected to the TX pin on the Arduino (digital pin 1) to receive data from the Arduino.

2-Channel Relay Module: Allows the Arduino to control the switching of high-voltage devices like light bulbs. **VCC:** Connected to the 5V pin on the Arduino to power the relay module. **GND:** Connected to the GND pin on the Arduino to complete the circuit. **IN1:** Connected to a digital output pin on the Arduino (PB pin 12) to control the first relay. **IN2:** Connected to another digital output pin on the Arduino (PB pin 13) to control the second relay. **COM (Common) and NO (Open) contacts:** Connected to the light bulbs and the power source. When the relay is activated, it connects the COM and NO contacts, turning the light on.

Light Bulbs: One terminal of each bulb is connected to the NO contact of the respective relay. The other terminal is connected to the neutral wire of the AC supply. The COM contact of each relay is connected to the live wire of the AC supply, completing the circuit when the relay is activated.

The **HC-06 Bluetooth module** receives voice commands from the Arduino Blue Control app via Bluetooth. The **Arduino UNO** processes these commands and sends signals to the **relay module** to switch the light bulbs on or off. Each relay on the module can control one light bulb. When the Arduino sends a HIGH signal to a relay input pin (IN1 or IN2), the relay activates, closing the circuit between the COM and NO contacts, thus powering the light bulb. When the Arduino sends a LOW signal, the relay deactivates, opening the circuit and turning off the light bulb.

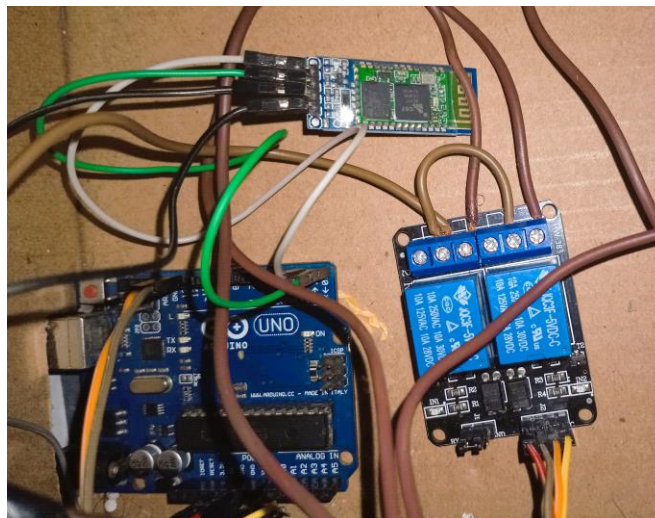


Figure 2.3.1.1 - Circuit diagram implementation of Voice-Controlled Light System

II. Function 2 - Arduino-Based Electric Meter [\[2\]](#)

ESP32 Microcontroller: Acts as the main processing unit, receiving data from sensors and controlling the display. Typically powered via USB or regulated power supply. Data: Connected to sensors and the LCD display via GPIO pins. **Voltage Sensor (ZMPT101B):** Measures the AC voltage from the power supply. Connected to the 220V AC power supply. Signal: Sends voltage data to the ESP32. **Current Sensor (CT Sensor):** Measures the current flowing through the load. Power: Connected in line with the load (bulb). Signal: Sends current data to the ESP32. **LCD Display:** Displays the voltage, current, and power consumption data. Typically powered by the ESP32. Data: Connected to the ESP32 via data lines (usually I2C or other digital pins). **Resistors and Capacitor:** Signal conditioning to ensure accurate readings from the current sensor. In parallel with the current sensor to condition the signal before sending it to the ESP32.

The ESP32 is powered via a USB or a regulated power input. The voltage sensor (ZMPT101B) is connected to the 220V AC supply to measure the voltage. The current sensor (CT sensor) is connected in line with the load (bulb) to measure the current. The ZMPT101B sends voltage data to the ESP32 via appropriate GPIO pins. The CT sensor sends current data to the ESP32 through its output pins, with resistors and a capacitor conditioning the signal. The LCD display receives processed data from the ESP32 and displays the measured voltage, current, and calculated power consumption. Resistors (10Ω and 100Ω) and a $10\mu\text{F}$ capacitor are used to condition the signal from the CT sensor to ensure it is within a readable range for the ESP32.

The voltage sensor (ZMPT101B) and current sensor (CT sensor) measure the AC voltage and current respectively. The ESP32 processes the incoming voltage and current data, calculates power consumption, and other relevant metrics. The LCD display, connected to the ESP32, shows real-time readings of voltage, current, and power consumption.

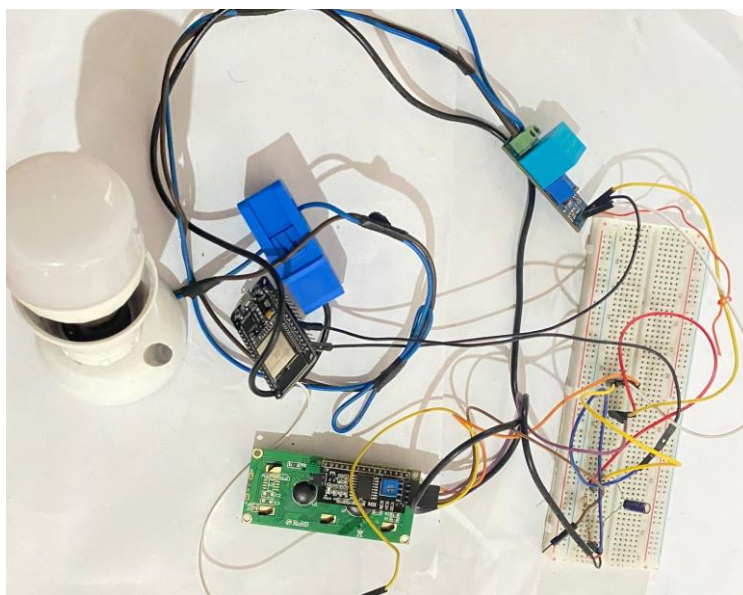


Figure 2.3.1.2 - Circuit diagram implementation of Arduino-based electric meter

III. Function 3 - Smart Fan with Temperature Sensing [\[3\]](#)

The system consists of several components, each playing a crucial role in the overall functionality.

Arduino UNO Board is the central microcontroller that processes inputs from the temperature

sensor and controls the fan speed via the motor driver module. **DHT11 Temperature Sensor:**

VCC: Connected to the 5V pin on the Arduino to provide power. **GND:** Connected to the GND pin on the Arduino to complete the circuit. **Data:** Connected to a digital input pin on the Arduino (pin D7) to send temperature data to the Arduino. **16x2 LCD Display with I2C Module:** Used to

display the current temperature and fan speed status. **VCC:** Connected to the 5V pin on the Arduino to provide power. **GND:** Connected to the GND pin on the Arduino to complete the circuit. **SDA:** Connected to the SDA pin on the Arduino (A4) for I2C communication. **SCL:** Connected to the

SCL pin on the Arduino (A5) for I2C communication. **L298N Motor Driver Module:** Controls the speed and direction of the fan motor based on signals from the Arduino. **IN1 and IN2:** Connected to two digital output pins on the Arduino (pinD10, D11) to control the motor's direction. **ENA:**

Connected to a PWM pin on the Arduino (pin D9) to control the motor's speed. **Motor Output:**

Connected to the fan motor, allowing it to receive power and control signals. **Fan Motor: Positive:** Connected to one of the motor outputs on the L298N motor driver. **Negative:** Connected to the

other motor output on the L298N motor driver.

The **DHT11 sensor** reads the temperature and sends the data to the Arduino. The Arduino processes this data and displays the current temperature on the **16x2 LCD**. Based on the temperature, the Arduino sends control signals to the **L298N motor driver**, which adjusts the speed of the **fan motor** to provide appropriate cooling. The LCD provides real-time feedback on the system's status, enhancing user interaction and monitoring capabilities.

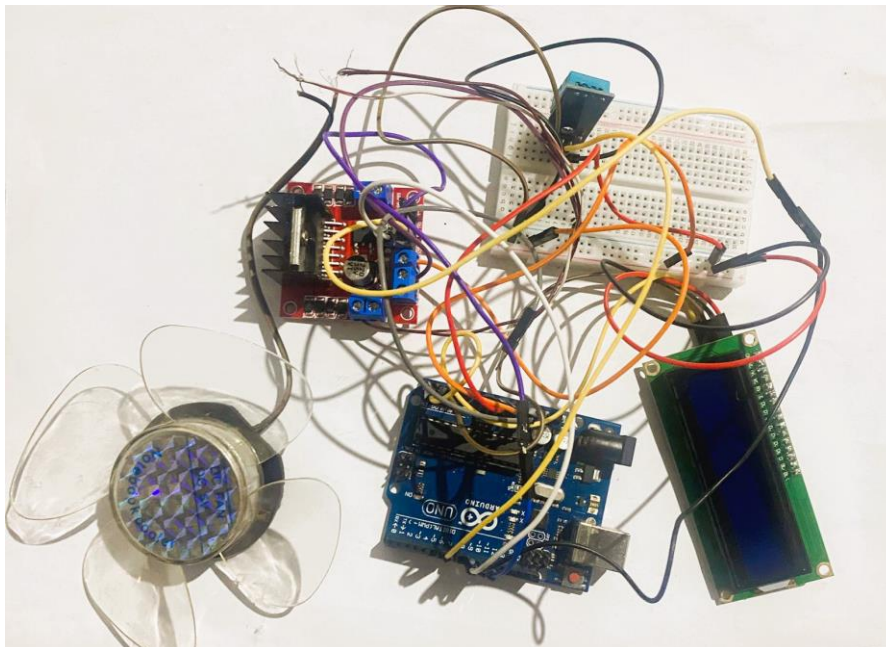


Figure 2.3.1.3 - Circuit diagram implementation of Smart fan with temperature sensing circuit

IV. Function 4 - Motion Sensor Light System [\[4\]](#)

This circuit involves an **Arduino Uno**, a relay module, a light bulb, and a PIR motion sensor. The PIR sensor detects motion and sends a signal to the Arduino, which then activates the relay to turn on the light bulb. The **PIR sensor's** VCC, OUT, and GND pins are connected to the Arduino's 5V, digital input (D2), and GND pins, respectively. **The relay's** VCC, IN, and GND pins are connected to the Arduino's 5V, digital output (D8), and GND pins, respectively. The relay switches the high-voltage circuit for the **light bulb**, completing the system for automatic light control based on motion detection.

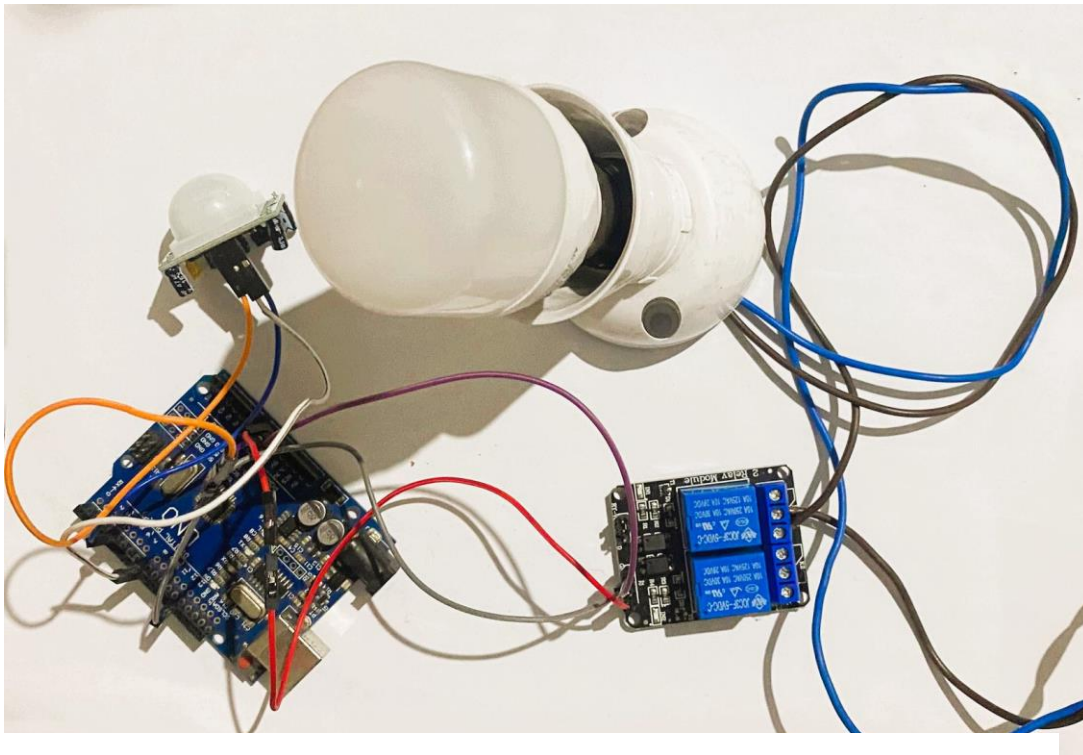


Figure 2.3.1.4 - Circuit diagram implementation of Motion sensor light system circuit

2.3.2 System circuit diagram

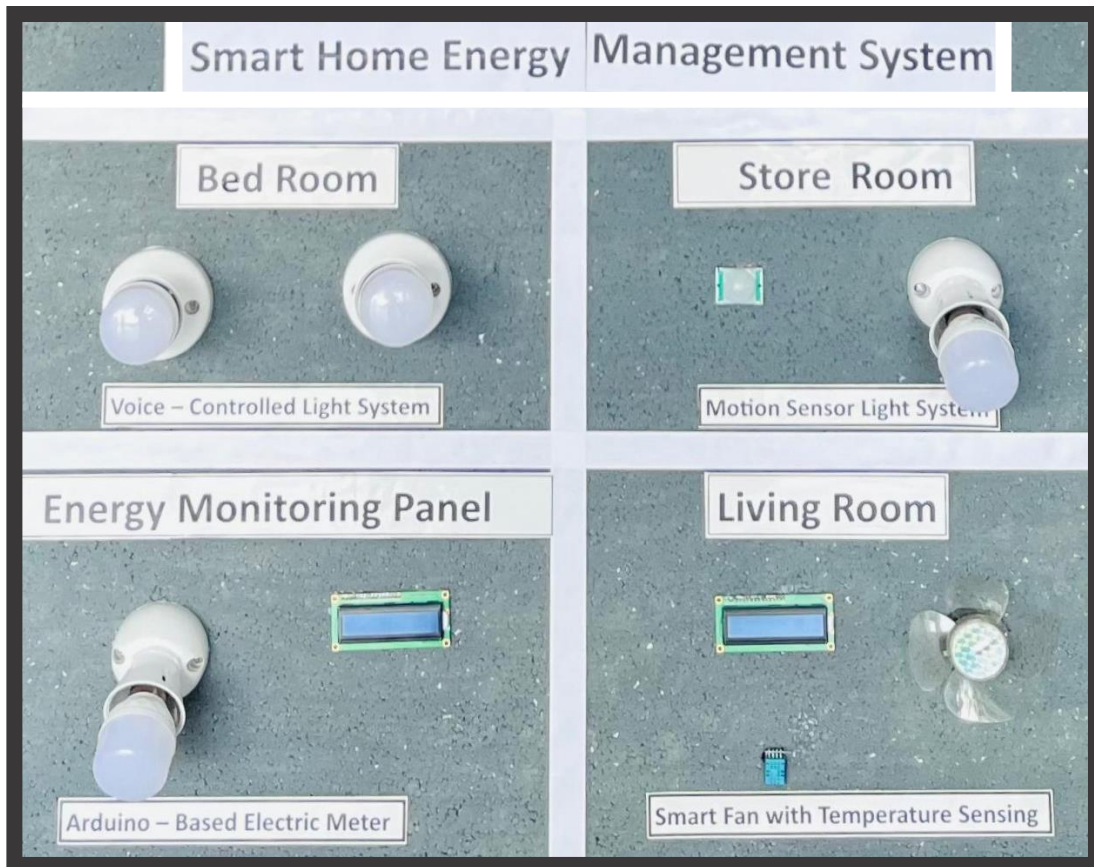


Figure 2.3.2.1 – Final system circuit diagram

Final Output of the System Circuit Diagram and Prototype: The overall system is designed to enhance the energy efficiency and convenience of a smart home. It integrates different technologies to automate lighting and fan control, monitor energy consumption, and respond to environmental changes (like motion and temperature). The voice-controlled lights provide a modern, hands-free way to manage lighting, while the motion sensor and temperature-based controls ensure that energy is used only when necessary. The energy monitoring panel offers valuable feedback on energy usage, allowing users to reduce their consumption and costs. This prototype demonstrates a cohesive smart home energy management solution that combines automation, user control, and monitoring for improved energy efficiency and convenience.

2.4 Testing

Table 2.4.1 - This test scenario confirms that the Voice-Controlled Light System functions as intended by responding appropriately to valid voice commands and ignoring invalid commands.

Test scenario ID		F1				
Test description		Voice-Controlled Light System automates light control based on voice commands received via Bluetooth signals from the Arduino Blue Control app.				
Test execution steps:						
Test Case No.	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Give voice command to turn on the light one	"Light one on"	Light one turns on	Light one turns on	Pass	Light activates correctly when the "light one on" command is given.
2	Give voice command to turn off the light one	"Light one off"	Light one turns off	Light one turns off	Pass	Light deactivates correctly when the " light one off " command is given.
3	Give voice command to turn on all lights	"All on"	All lights turn on	All lights turn on	Pass	Light activates correctly when the "All on" command is given.
4	Give voice command to turn off all lights	"All off"	All lights turn off	All lights turn off	Pass	Light deactivates correctly when the " All off " command is given.
5	Give an invalid voice command	"Invalid command"	No action taken	No action taken	Pass	System correctly ignores an invalid voice command.

Table 2.4.2 - test scenario confirms that the Arduino-Based Electric Meter accurately measures voltage and current, providing information on energy consumption correctly.

Test scenario ID		F2				
Test description		Arduino-Based Electric Meter measures and displays current and voltage readings, providing information on energy consumption and per-unit cost				
Test execution steps:						
Test Case No.	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Measure voltage from a stable source	Voltage input: 230V	Display shows 230V	Display shows 230V	Pass	Voltage reading matches the input source accurately.
2	Measure current from a stable source	Current input: 5A	Display shows 5A	Display shows 5A	Pass	Current reading matches the input source accurately.
3	Simulate energy consumption over time	Voltage: 230V, Current: 5A, Time: 1 hour	Blynk app shows energy consumption and per-unit cost	Blynk app shows accurate values	Pass	Energy consumption and cost calculations are accurate.

Table 2.4.3 - This test scenario confirms that the Smart Fan with Temperature Sensing system operates correctly across different temperature ranges.

Test scenario ID		F3				
Test description		Smart Fan with Temperature Sensing system automatically adjusts fan speed based on real-time temperature readings from the DHT11 sensor.				
Test execution steps:						
Test Case No.	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Set the temperature sensor to low than 29°C	Temperature < 29°C	Fan speed set to Low	Fan speed set to Low	Pass	Fan adjusts to low speed correctly at lower temperature.
2	Set the temperature sensor between 29°C - 32°C	29°C<= Temperature < 32°C	Fan speed set to Medium	Fan speed set to Medium	Pass	Fan adjusts to medium speed correctly at moderate temperature.
3	Set the temperature sensor to 32°C or above	Temperature>=32°C	Fan speed set to High	Fan speed set to High	Pass	Fan adjusts to high speed correctly at higher temperature.

Table 2.4.4 - This test scenario confirms that the Motion Sensor Light System operates correctly by responding to motion detection signals.

Test scenario ID		F4				
Test description		Motion Sensor Light System automatically turns on the light when motion is detected and turns it off when no motion is detected.				
Test execution steps:						
Test Case No.	Action	Inputs	Expected Output	Actual Output	Test Result	Test Comments
1	Simulate motion in sensor range	Motion detected	Light turns on	Light turns on	Pass	Light activates correctly when motion is detected
2	No motion in sensor range for 30s	No motion detected	Light remains off	Light remains off	Pass	Light remains off correctly when no motion is detected.
3	Continuous motion in sensor range	Continuous motion	Light stays on	Light stays on	Pass	Light stays on correctly as long as motion is detected.

3. Evaluation

3.1 Assessment of the project results

The smart home energy management system, as outlined in the high-level architecture diagram, demonstrates a well-integrated and functional approach to automating home utilities. Here's an assessment of the project results based on the four interconnected subsystems:

I. Voice-Controlled Lighting System:

- **Functionality:** Successfully allows users to control lighting via voice commands and Bluetooth signals. The system effectively reduces electricity costs by automating light usage.
- **Performance:** Responds promptly to user commands with minimal delay, ensuring a seamless user experience.
- **Reliability:** The integration of the HC-06 Bluetooth module and Arduino UNO board provides a stable and reliable control mechanism.

II. Temperature-Controlled Fan System:

- **Functionality:** Automatically adjusts fan speed based on real-time temperature readings, enhancing comfort and energy efficiency.
- **Performance:** The DHT11 temperature sensor provides accurate readings, and the fan speed adjusts smoothly, maintaining optimal room temperature.
- **Reliability:** The system's components work together reliably, with clear and accurate temperature displays on the LED screen.

III. Motion-Controlled Lighting System:

- **Functionality:** Efficiently automates lighting based on motion detection, turning lights on only when needed to save energy.
- **Performance:** The PIR motion sensor detects movement effectively, and the relay activates the light promptly.
- **Reliability:** The system is dependable, ensuring lights are only on when motion is detected, thereby preventing unnecessary energy consumption.

IV. Smart Energy Monitoring System:

- **Functionality:** Monitors and displays real-time energy consumption data, helping users understand and manage their energy usage better.

- **Performance:** The current sensors and transformer provide accurate measurements, and the data is displayed clearly on the LCD screen. The ESP32 module ensures easy remote monitoring.
- **Reliability:** The system is robust and consistently provides accurate energy usage data, contributing to overall energy savings.

3.1.1 Overall Assessment

The project demonstrates effective integration of various sensors and modules to create a cohesive smart home energy management system. Each subsystem performs its intended functions reliably and efficiently, contributing to overall energy savings and user convenience. The use of Arduino UNO boards provides a flexible and scalable platform, allowing for future expansions and enhancements. The project successfully achieves its goals of automating home utilities, reducing energy consumption, and enhancing user comfort.

3.2 Lesson learned

Integration of Diverse Technologies:

- **Lesson:** Successfully integrating multiple sensors, modules, and control boards (like the Arduino UNO and ESP32) is key to creating a cohesive smart home system.
- **Insight:** Understanding the compatibility and communication protocols between different components is crucial for seamless operation.

Importance of Real-Time Responsiveness:

- **Lesson:** Real-time responsiveness in systems like voice-controlled lighting and motion-activated lighting is essential for user satisfaction.
- **Insight:** Ensuring minimal latency in response times enhances the user experience and reliability of the system.

Accuracy in Sensor Readings:

- **Lesson:** Accurate sensor readings (e.g., temperature and current sensors) are vital for the correct functioning of automated systems.
- **Insight:** Regular calibration and maintenance of sensors ensure long-term accuracy and efficiency of the system.

Energy Efficiency and Cost Savings:

- **Lesson:** Automated systems can significantly reduce energy consumption and costs by optimizing the use of home utilities.
- **Insight:** Implementing smart control mechanisms based on real-time data (such as motion detection and temperature sensing) can lead to substantial energy savings.

User-Friendly Interfaces:

- **Lesson:** Providing clear and intuitive interfaces (like the Arduino Bluetooth control app and LED displays) is essential for user adoption and interaction.
- **Insight:** Designing user-friendly interfaces that display relevant information clearly helps users to better manage and control their smart home systems.

Reliability and Robustness:

- **Lesson:** The reliability of each subsystem is crucial for the overall dependability of the smart home system.
- **Insight:** Using robust components and ensuring proper installation and integration can minimize system failures and maintenance needs.

Scalability and Flexibility:

- **Lesson:** Designing systems that are easily expandable allows for future upgrades and integration of additional devices.
- **Insight:** Using modular designs and standard communication protocols (like Bluetooth and WiFi) facilitates easy scaling and integration.

Data Security and Privacy:

- **Lesson:** Ensuring data security and user privacy is critical, especially when dealing with remote monitoring and control systems.
- **Insight:** Implementing secure communication methods and access controls helps protect user data and system integrity.

Documentation and Knowledge Sharing:

- **Lesson:** Comprehensive documentation and knowledge sharing are important for troubleshooting and future development.
- **Insight:** Maintaining detailed project documentation, including system designs, code, and user manuals, supports ongoing maintenance and future enhancements.

Collaborative Development:

- **Lesson:** Collaboration among team members with diverse skills (e.g., software development, hardware integration, and user experience design) is essential for project success.
- **Insight:** Encouraging teamwork and regular communication ensures that all aspects of the project are aligned and effectively addressed.

These lessons highlight the importance of careful planning, integration, user-centric design, and ongoing maintenance in developing effective and reliable smart home systems.

3.3 Future work

Enhanced Voice Recognition Capabilities:

- **Improvement:** Integrate advanced natural language processing (NLP) algorithms to improve voice command recognition accuracy and expand the range of voice commands.
- **Benefit:** Enhances user interaction and system responsiveness, making the voice-controlled lighting system more user-friendly and efficient.

Integration with Smart Home Ecosystems:

- **Improvement:** Ensure compatibility with major smart home ecosystems like Google Home, Amazon Alexa, and Apple HomeKit.
- **Benefit:** Provides users with greater flexibility and convenience by allowing them to control the system through various platforms and devices.

AI-Powered Predictive Analytics:

- **Improvement:** Implement machine learning algorithms to predict user behavior and optimize energy usage based on historical data and usage patterns.
- **Benefit:** Increases energy efficiency and cost savings by proactively managing home utilities according to predicted user needs.

Advanced Security Features:

- **Improvement:** Incorporate robust cybersecurity measures, including encryption, secure communication protocols, and multi-factor authentication.
- **Benefit:** Protects user data and system integrity, ensuring a secure and trustworthy smart home environment.

Energy Source Integration:

- **Improvement:** Enable integration with renewable energy sources such as solar panels and wind turbines.
- **Benefit:** Promotes sustainability and reduces reliance on grid electricity, contributing to lower energy costs and environmental impact.

Comprehensive Energy Management Dashboard:

- **Improvement:** Develop a comprehensive dashboard that provides real-time insights into energy consumption, costs, and savings, accessible via mobile and web applications.
- **Benefit:** Empowers users with detailed information and analytics to make informed decisions about their energy usage.

Scalability Enhancements:

- **Improvement:** Design the system architecture to support seamless scaling, allowing for easy addition of new devices and sensors without significant reconfiguration.
- **Benefit:** Facilitates future expansion and customization, accommodating growing user needs and technological advancements.

Enhanced Sensor Accuracy and Reliability:

- **Improvement:** Invest in higher precision sensors and implement regular calibration routines to ensure the accuracy and reliability of data readings.
- **Benefit:** Improves the overall effectiveness and trustworthiness of the system's automated responses and energy management capabilities.

User Customization Options:

- **Improvement:** Provide users with more customization options to set preferences for lighting, temperature, and energy monitoring based on personal needs and schedules.
- **Benefit:** Enhances user satisfaction by allowing tailored control over their smart home environment.

Environmental and Health Monitoring:

- **Improvement:** Integrate additional sensors to monitor air quality, humidity, and other environmental factors.
- **Benefit:** Creates a healthier and more comfortable living environment, alerting users to potential issues and allowing for proactive adjustments.

4. Conclusion

The Smart Energy Management System project successfully achieves its primary objectives of enhancing home automation, optimizing energy consumption, and providing users with greater control over their home utilities. Through the integration of voice-controlled lighting, temperature-based fan control, motion-activated lighting, and comprehensive energy monitoring, the system offers a robust and user-friendly solution for managing energy usage efficiently.

4.1 Realization of Objectives

The original goals of the project—automating home utilities to reduce energy consumption and enhance user convenience—have been effectively realized. Each subsystem functions reliably and performs its intended tasks with high accuracy and responsiveness. The voice-controlled lighting system reduces electricity costs by automating light usage based on voice commands and Bluetooth signals. The temperature-controlled fan system adjusts fan speed according to real-time temperature readings, ensuring optimal comfort and energy efficiency. The motion-activated lighting system conserves energy by turning lights on only when motion is detected. Lastly, the energy monitoring system provides real-time data on energy consumption, helping users make informed decisions to optimize their energy usage.

4.2 Future Work and Enhancements

While the project has met its primary objectives, there are several areas for further improvement and enhancement. Integrating advanced voice recognition capabilities can improve the accuracy and range of voice commands, making the system even more user-friendly. Compatibility with major smart home ecosystems like Google Home, Amazon Alexa, and Apple HomeKit will provide greater flexibility and convenience for users. Implementing AI-powered predictive analytics can optimize energy usage by predicting user behavior based on historical data.

4.3 Addressing Weaknesses and Limitations

One of the main limitations of the current system is its reliance on individual components that may require manual calibration and maintenance. To address this, investing in higher precision sensors and implementing regular calibration routines can ensure long-term accuracy and reliability. Additionally, enhancing the system's security features by incorporating robust cybersecurity measures will protect user data and maintain system integrity.

Another limitation is the potential complexity of scaling the system to accommodate more devices and sensors. Designing the system architecture to support seamless scaling can facilitate future expansions and customizations, ensuring the system can grow with the user's needs.

4.4 Benefits to the Client Organization

The development of this Smart Energy Management System offers significant benefits to the client organization. It provides a comprehensive solution that enhances user convenience, optimizes energy consumption, and reduces electricity costs. By promoting energy efficiency and sustainability, the system aligns with modern environmental goals and can be a valuable addition to the client's product portfolio. Furthermore, the system's scalability and compatibility with other smart home ecosystems position it well for future market demands, ensuring long-term relevance and competitiveness.

4.5 Final Thoughts

In conclusion, the Smart Energy Management System project has successfully developed a functional and efficient solution for modern home automation and energy management. By addressing the identified weaknesses and implementing the suggested enhancements, the system can be further refined and expanded to meet evolving user needs and technological advancements. The project not only achieves its initial objectives but also lays a strong foundation for future developments, offering substantial benefits to both users and the client organization.

5. References

- [1] Hewagamage, N. [@NisalHewagamage]. (2018, August 22). *Arduino Voice Control Room Lighting | Make your home “Smart” [2018 Sinhala]*. Youtube.
<https://www.youtube.com/watch?v=EsfCOY9vDFo>

- [2] Research Tamilan [@researchtamilan4106]. (2023, April 23). *iot smart energy meter using nodemcu esp32 with Blynk app | iot projects using esp32 #iotproject*. Youtube.
<https://www.youtube.com/watch?v=pp6OIvShxIo>

- [3] RoboTechZone [@robotechzone]. (2023, July 15). *Temperature based fan speed controller using arduino UNO || ARDUINO PROJECTS*. Youtube.
<https://www.youtube.com/watch?v=Z1vYp8NE9LI>

- [4] Shameer, T. T. [@TechTrendsShameer]. (2021, July 13). *Motion Sensor Control Electric Bulb | home automation*. Youtube.
<https://www.youtube.com/watch?v=UUIAMvLilb0>

Appendix A: Selected Code Listings

F1 - Voice-Controlled Light System

```
char val;

void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  Serial.begin(9600); // Assuming you want to use Serial communication
}
```

Serial Communication Setup: `Serial.begin(9600);` initializes serial communication at a baud rate of 9600 to receive voice commands.

Pin Setup: `pinMode(13, OUTPUT);` and `pinMode(12, OUTPUT);` set pins 13 and 12 as output pins to control two lights.

```
void loop() {
  if (Serial.available()) {
    val = Serial.read();
  }
```

Reading Voice Commands: `if (Serial.available())` checks if data is available to read from the serial port. `val = Serial.read();` reads the incoming data (assumed to be a voice command).

```
    // ON
    if (val == 'a') {
      digitalWrite(13, LOW);
    } else if (val == 'b') {
      digitalWrite(12, LOW);
    }

    // OFF
    else if (val == 'c') {
      digitalWrite(13, HIGH);
    } else if (val == 'd') {
      digitalWrite(12, HIGH);
    }
  }
```

```
// ALL ON
else if (val == 'e') {
    digitalWrite(13, LOW);
    digitalWrite(12, LOW);
}

// ALL OFF
else if (val == 'f') {
    digitalWrite(13, HIGH);
    digitalWrite(12, HIGH);
}
}
}
```

Different `if` and `else if` statements check the value of `val` to determine the action:

- 'a' turns on the light connected to pin 13.
- 'b' turns on the light connected to pin 12.
- 'c' turns off the light connected to pin 13.
- 'd' turns off the light connected to pin 12.
- 'e' turns on both lights.
- 'f' turns off both lights.

F2 - Arduino-Based Electric Meter

```
#define BLYNK_TEMPLATE_ID "TMPL6KLuZJAvM"
#define BLYNK_TEMPLATE_NAME "Smart Energy Meter"
#include <LiquidCrystal_I2C.h>
#include "EmonLib.h"
#include <EEPROM.h>
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

EnergyMonitor emon;
#define vCalibration 83.3
#define currCalibration 0.50
LiquidCrystal_I2C lcd(0x3F, 20, 4);
BlynkTimer timer;
char auth[] = "0DhUe8f026nFX5D7i7BG4Dpy7QHeKN4G";
char ssid[] = "iPhone";
char pass[] = "12345678";

float kWh = 0;
unsigned long lastmillis = millis();

void myTimerEvent()
{
    emon.calcVI(20, 2000);
    kWh = kWh + emon.apparentPower * (millis() - lastmillis) / 3600000000.0;
    yield();

    float vrms = emon.Vrms;
    float irms = emon.Irms;
    float apparentPower = emon.apparentPower;
```



```

Serial.print(vrms, 2);
Serial.print("V");
EEPROM.put(0, vrms);
delay(100);

Serial.print("\tIrms: ");
Serial.print(irms, 4);
Serial.print("A");
EEPROM.put(4, irms);
delay(100);

Serial.print("\tPower: ");
Serial.print(apparentPower, 4);
Serial.print("W");
EEPROM.put(8, apparentPower);
delay(100);

Serial.print("\tkWh: ");
Serial.print(kWh, 5);
Serial.println("kWh");
EEPROM.put(12, kWh);

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Vrms:");
lcd.print(vrms, 2);
lcd.print("V");
lcd.setCursor(0, 1);
lcd.print("Irms:");
lcd.print(irms, 4);
lcd.print("A");

```

```

    lcd.setCursor(0, 2);
    lcd.print("Power:");
    lcd.print(apparentPower, 4);
    lcd.print("W");
    lcd.setCursor(0, 3);
    lcd.print("kWh:");
    lcd.print(kWh, 4);
    lcd.print("kWh");

    lastmillis = millis();

    // Check if Blynk is connected before sending data
    if (Blynk.connected()) {
        Serial.println("Sending data to Blynk...");
        Serial.print("V0 (Vrms): ");
        Serial.println(vrms);
        Serial.print("V1 (Irms): ");
        Serial.println(irms);
        Serial.print("V2 (Power): ");
        Serial.println(apparentPower);
        Serial.print("V3 (kWh): ");
        Serial.println(kWh);

        Blynk.virtualWrite(V0, vrms);
        Blynk.virtualWrite(V1, irms);
        Blynk.virtualWrite(V2, apparentPower);
        Blynk.virtualWrite(V3, kWh);
    } else {
        Serial.println("Blynk not connected. Data not sent.");
    }
}

void setup()
{
    Serial.begin(9600);
    Blynk.begin(auth, ssid, pass);
    lcd.init();
    lcd.backlight();
    emon.voltage(35, vCalibration, 1.7); // Voltage: input pin, calibration, phase_shift
    emon.current(34, currCalibration);   // Current: input pin, calibration.

    timer.setInterval(5000L, myTimerEvent);
    lcd.setCursor(3, 0);
    lcd.print("IoT Energy");
    lcd.setCursor(5, 1);
    lcd.print("Meter");
    delay(3000);
    lcd.clear();
}

void loop()
{
    Blynk.run();
    timer.run();
}

```

EnergyMonitor emon: Object from EmonLib to perform energy monitoring.vCalibration and currCalibration: Calibration constants for voltage and current.emon.calcVI(20, 2000): Measures voltage and current, calculates power and energy values over 20 cycles and 2000 ms. Energy Calculation: kWh is updated based on the apparent power and elapsed time.Serial Printing and EEPROM Storage: Logs and stores Vrms, Irms, apparent power, and kWh in EEPROM. LCD Display: Updates the LCD with the latest measurements.Blynk Updates: Sends data to Blynk virtual pins if connected. Blynk.begin(auth, ssid, pass): Connects the ESP32 to WiFi and Blynk.lcd.init() and lcd.backlight(): Initializes and turns on the LCD backlight. emon.voltage(35, vCalibration, 1.7): Sets up voltage measurement on pin 35 with calibration and phase shift. emon.current(34, currCalibration): Sets up current measurement on pin 34 with calibration.timer.setInterval(5000L, myTimerEvent): Schedules myTimerEvent to run every 5 seconds. LCD Welcome Message: Displays "IoT Energy Meter" on the LCD during startup.

F3 - Smart Fan with Temperature Sensing

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define DHTPIN 7           // DHT11 sensor data pin
#define DHTTYPE DHT11      // DHT sensor type

#define MOTOR_PIN_ENA 9    // Enable pin for motor driver (PWM pin)
#define MOTOR_PIN_IN1 10   // Input pin 1 for motor driver
#define MOTOR_PIN_IN2 11   // Input pin 2 for motor driver

#define TEMPERATURE_THRESHOLD 29
#define TEMPERATURE_THRESHOLD1 32 // Temperature threshold to adjust motor speed
```

1. Initialization and Setup

- **DHT Sensor Initialization:**

```
DHT dht(DHTPIN, DHTTYPE);
```

Initializes the DHT sensor for temperature readings.

- **LCD Initialization:**

```
LiquidCrystal_I2C lcd(0x3F, 16, 2);

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

  Wire.begin();
  lcd.init();
  lcd.backlight();
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Temperature:");
}
```

Sets up the LCD display to show temperature and motor speed.

2. Temperature Reading

- **Reading Temperature**

```
void loop() {
  delay(2000); // Delay between readings (adjust as needed)

  float temperature = dht.readTemperature(); // Read temperature in Celsius
  if (isnan(temperature)) {
    Serial.println("Failed to read temperature from DHT sensor!");
    return;
  }
}
```

Adds a 2-second delay between temperature readings to allow the sensor to stabilize.

Reads the temperature from the DHT sensor and checks for errors.

```
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" °C");

lcd.setCursor(0, 1); // Set cursor to the second line
lcd.print("          "); // Clear the previous motor speed
```

3. Motor Speed Control Based on Temperature

- **Motor Speed Control Algorithm**

```
if (temperature > TEMPERATURE_THRESHOLD1) {  
    // Increase motor speed  
    analogWrite(MOTOR_PIN_ENA, 255);    // maximum speed  
    digitalWrite(MOTOR_PIN_IN1, HIGH);  
    digitalWrite(MOTOR_PIN_IN2, LOW);  
    lcd.setCursor(0, 1);  
    lcd.print("Motor Speed: Max");  
}  
else if (temperature > TEMPERATURE_THRESHOLD) {  
  
    analogWrite(MOTOR_PIN_ENA, 100);    // Medium speed  
    digitalWrite(MOTOR_PIN_IN1, HIGH);  
    digitalWrite(MOTOR_PIN_IN2, LOW);  
    lcd.setCursor(0, 1);  
    lcd.print("Motor Speed: Med");  
}  
else {  
    // Decrease motor speed  
    analogWrite(MOTOR_PIN_ENA, 45);    // low speed  
    digitalWrite(MOTOR_PIN_IN1, HIGH);  
    digitalWrite(MOTOR_PIN_IN2, LOW);  
    lcd.setCursor(0, 1);  
    lcd.print("Motor Speed: Low");  
}
```

Conditions: Checks temperature against predefined thresholds.

Actions: Sets motor speed using `analogWrite` to different values based on the temperature. Adjusts motor direction using `digitalWrite`.

4. Displaying Information on LCD

- **LCD Display Update**

```
lcd.setCursor(12, 0); // Set cursor to the temperature position on the first line
lcd.print(temperature); // Print temperature
```

F4 - Motion Sensor Light System

This Arduino code sets up a simple motion sensor light system.

```
void setup() {
  pinMode(8, INPUT);
  pinMode(13, OUTPUT);

}

void loop() {
  int val = digitalRead(8);

  if(val == 1){

    digitalWrite(13,LOW);
  }

  else {
    digitalWrite(13,HIGH);
  }
}
```

In the setup() function, pin 8 is configured as an input to read signals from a motion sensor, and pin 13 is configured as an output to control an LED. The loop() function continuously checks the input from the motion sensor. If motion is detected (the sensor outputs a value of 1), the LED connected to pin 13 is turned off by writing a LOW signal to it. Conversely, if no motion is detected (the sensor outputs 0), the LED is turned on by writing a HIGH signal to pin 13. This setup can be used for a basic security or automation system where an LED light is toggled based on motion detection. The logic assumes the LED is active LOW, meaning it turns on with a HIGH signal and off with a LOW signal.