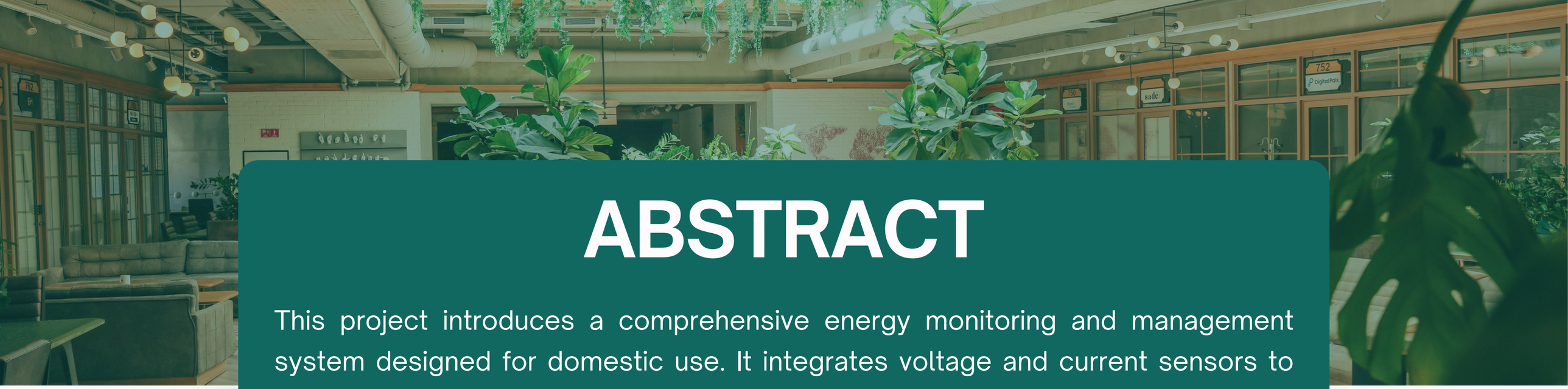
PROJECT PRESENTATION

Smart Energy Monitoring System with Environmental Adaptation

Description:
Real - Time Energy Consumption Monitoring

Team Members :

Santha Lakshmi S, Megashree M, Priyadharshini S S,
Shifa Anjum S, Siddavattam Deekshitha



ABSTRACT

This project introduces a comprehensive energy monitoring and management system designed for domestic use. It integrates voltage and current sensors to track key electrical parameters, displayed in real-time on the Blynk IoT platform and Arduino's serial monitor. Power calculations are computed using a theory formulas. The system controls a bulb and a fan through Arduino, with fan speed regulated by a motor driver based on temperature changes. Additionally, a relay is employed to cut off power supply during excessive consumption. Motion sensors are utilized to intelligently control lights and fans, contributing to energy conservation. Future enhancements include implementing energy consumption limits and weather-based fan speed adjustments. Overall, this project offers an efficient and sustainable approach to household energy management.

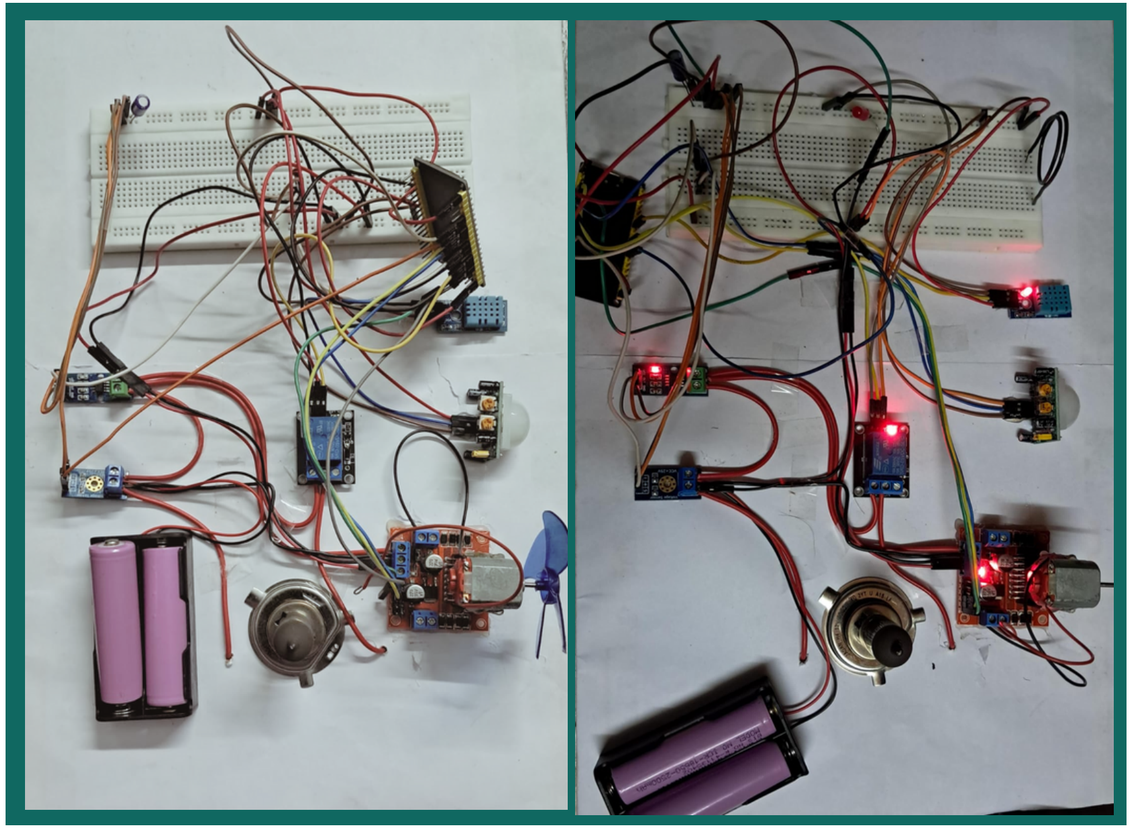
INTRODUCTION

- Smart energy monitoring and management system for domestic environments.
- Utilizes voltage and current sensors to track key electrical parameters: voltage, current, power, and energy usage.
- Real-time display of data on the Blynk IoT platform and Arduino's serial monitor.
- Innovative control mechanisms, including motor drivers and relays, for autonomous management of devices like bulbs and fans.
- Optimization of device operation based on environmental factors such as temperature and occupancy.

Focuses on energy conservation and efficiency, providing a practical solution for households to reduce environmental footprint and utility costs.

PROPOSED MODEL

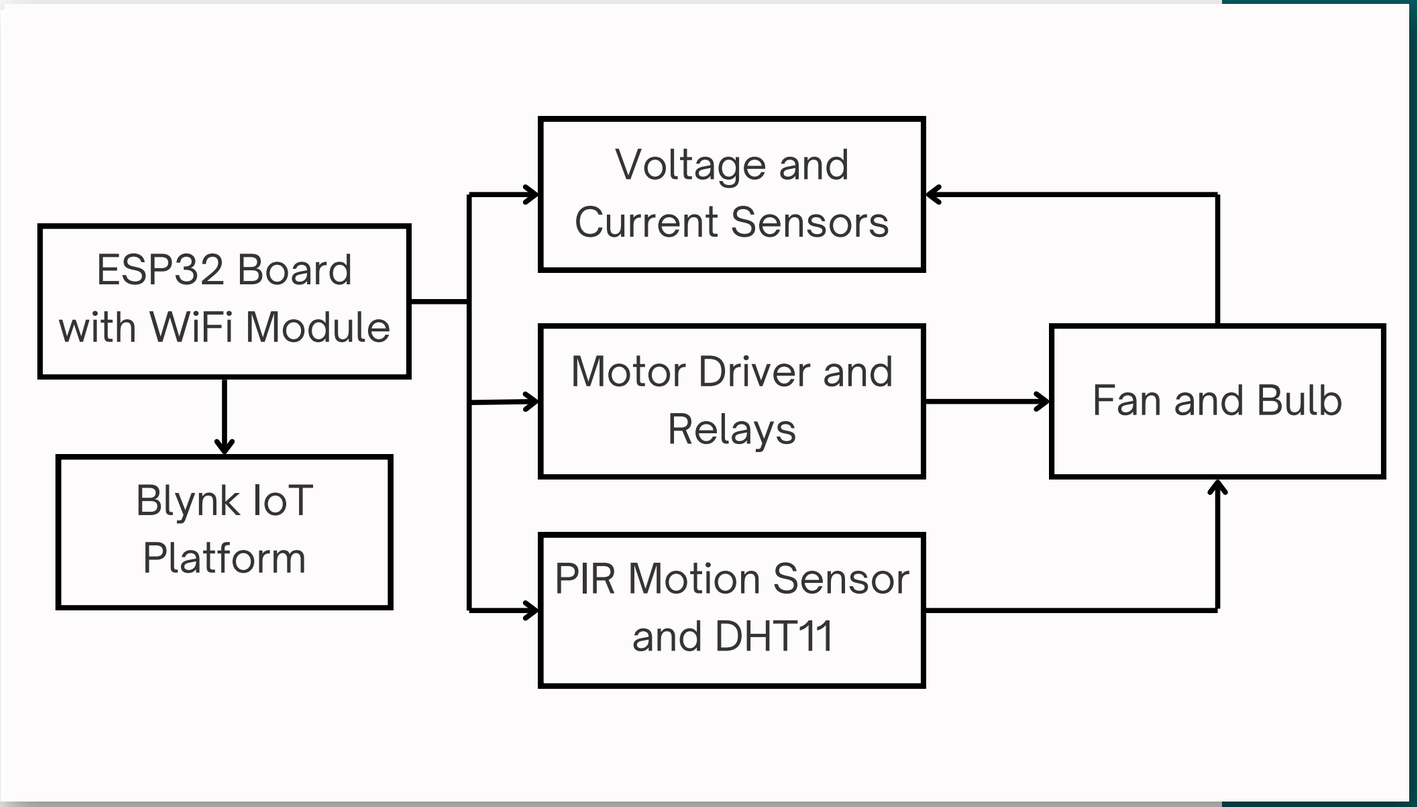
- Integration of voltage and current sensors for real-time electrical parameter monitoring.
- Utilization of Blynk IoT platform and Arduino's serial monitor for data display and analysis.
- Innovative control mechanisms like motor drivers and relays for autonomous device management.
- Integration of energy consumption limits for resource-efficient operation.
- Introduction of weather-based fan speed adjustments for optimized energy usage.
- Implementation of motion-sensing technology for intelligent control of lights and fans.
- Aim to provide households with a practical and sustainable energy management solution.

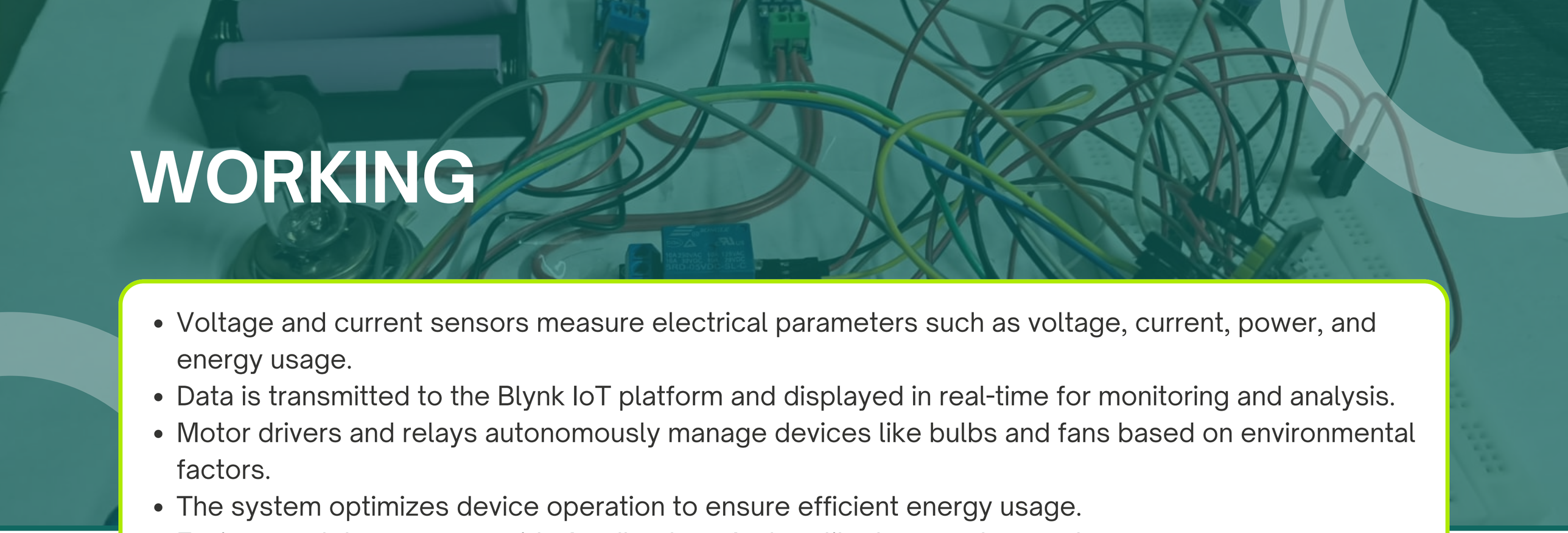


CONSTRUCTION

- Assemble the ESP32 board and connect it to the voltage and current sensors.
- Configure the Blynk IoT platform and integrate it with the ESP32 board for data visualization.
- Set up the Arduino IDE and establish communication with the ESP32 board.
- Install the necessary libraries for sensor interfacing and data processing.
- Connect the motor driver and relay modules to the ESP32 board for device control.
- Integrate environmental sensors such as temperature and occupancy sensors for environmental feedback.
- Using formulas for power calculation, energy consumption limits, and weather-based fan speed adjustments.
- Implement motion-sensing technology for intelligent control of lights and fans using PIR Motion Sensor.

BLOCK DIAGRAM





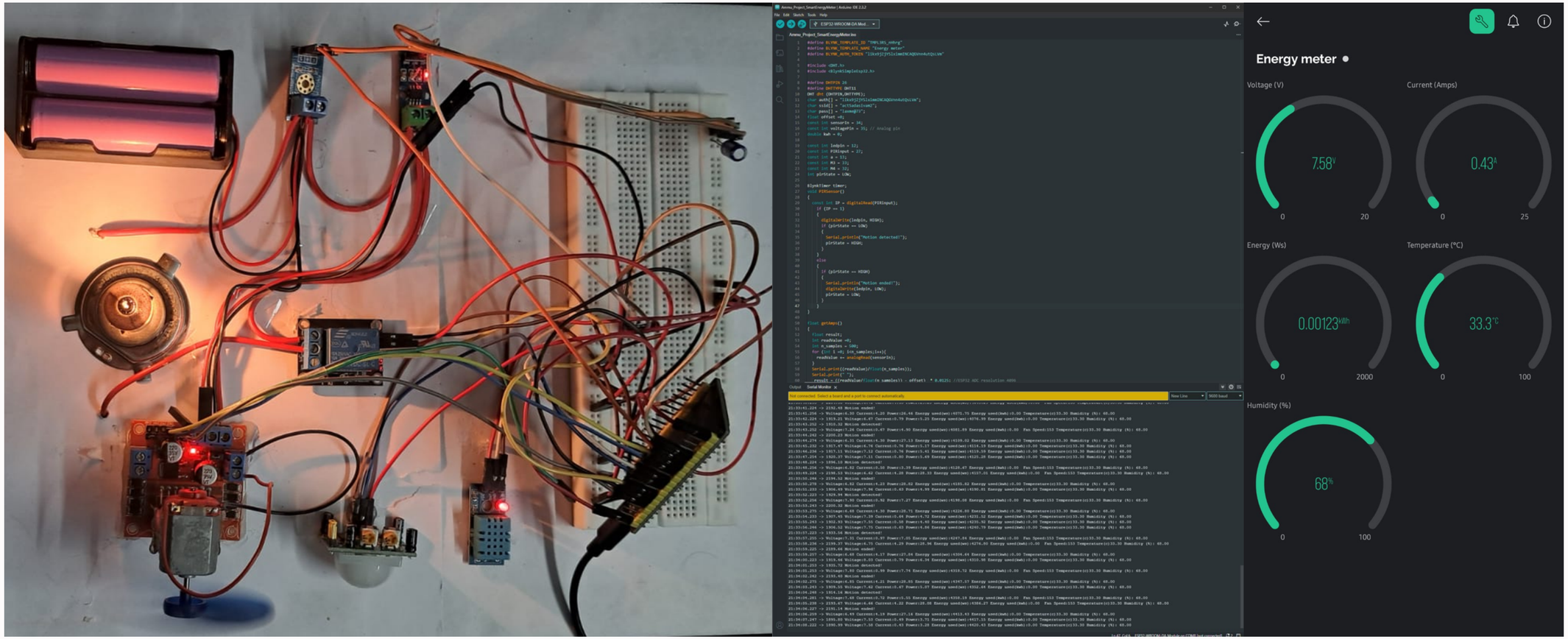
WORKING

- Voltage and current sensors measure electrical parameters such as voltage, current, power, and energy usage.
- Data is transmitted to the Blynk IoT platform and displayed in real-time for monitoring and analysis.
- Motor drivers and relays autonomously manage devices like bulbs and fans based on environmental factors.
- The system optimizes device operation to ensure efficient energy usage.
- Environmental sensors provide feedback on factors like temperature and occupancy.
- Motion-sensing technology controls lights and fans to conserve energy.
- Users can interact with the system through the Blynk app and Arduino's serial monitor.
- Focus on energy conservation and efficiency helps reduce environmental footprint and utility costs.
- Continuous improvement allows for optimization and expansion to meet evolving energy management needs.

CODE:

```
#define BLYNK_TEMPLATE_ID "TMPL3RS_nHhrg"
#define BLYNK_TEMPLATE_NAME "Energy meter"
#define BLYNK_AUTH_TOKEN
"likx9jZjYSlximmINCAQGVnn4utQsLVm"
#include <DHT.h>
#include <BlynkSimpleEsp32.h>
#define DHTPIN 26
#define DHTTYPE DHT11
DHT dht (DHTPIN,DHTTYPE);
char auth[] = "likx9jZjYSlximmINCAQGVnn4utQsLVm";
char ssid[] = "actSadasivam2";
char pass[] = "laxme@73";
float offset =0;
const int sensorIn = 34;
const int voltagePin = 35; // Analog pin
double kwh = 0;
const int relaypin = 12;
const int PIRinput = 27;
const int a = 13;
const int M3 = 33;
const int M4 = 32;
int pirState = LOW;
BlynkTimer timer;
void PIRSensor(){
 const int IP = digitalRead(PIRinput);
 if (IP == 1) {
 digitalWrite(ledpin, HIGH);
 if (pirState == LOW) {
 Serial.println("Motion detected!");
 pirState = HIGH; } }
 else {
if (pirState == HIGH) {
 Serial.println("Motion ended!");
 digitalWrite(ledpin, LOW);
 pirState = LOW; } }}
float getAmps(){
 float result;
 int readValue =0;
 int n_samples = 500;
 for (int i =0; i<n_samples;i++){
 readValue += analogRead(sensorIn); }
 Serial.print((readValue)/float(n_samples));
 Serial.print(" ");
 result = ((readValue/float(n_samples)) - offset) *
0.0125; //ESP32 ADC resolution 4096
 if (result < 0) {
 result *=-1; }
 return result;}
void sendSensorData() {
 float voltage = (analogRead(voltagePin)*(3.3/4095.0))*
((34010.0+7450.0)/7450.0);
 float current = getAmps();
 PIRSensor();
 float power = voltage * current;
 kwh += power;
 Serial.print("Voltage:");
 Serial.print(voltage);
 Serial.print(" Current:");
 Serial.print(current);
 Serial.print(" Power:");
 Serial.print(power);
 Serial.print(" Energy used(ws):");
 Serial.print(kwh);
 Serial.print(" Energy used(kwh):");
 Serial.print(kwh/(3600*1000));
 Blynk.virtualWrite(V0, voltage);
 Blynk.virtualWrite(V1, current);
 Blynk.virtualWrite(V2, kwh/(3600*1000));
 float temperature = dht.readTemperature();
 float humidity = dht.readHumidity();
 if (isnan(temperature)||isnan(humidity)) {
 Serial.println("DHT Sensor is not connected"); }
 else {
 if(pirState == HIGH) {
 digitalWrite(M3,HIGH);
 digitalWrite(M4,LOW);
 int val = map(temperature, 30,35, 0, 255);
 Serial.print(" Fan Speed:");
 Serial.print(val);
 analogWrite (a,val); } }
 Serial.print(" Temperature(c)");
 Serial.print(temperature);
 Serial.print(" Humidity (%): ");
 Serial.println(humidity);
 Blynk.virtualWrite(V3, temperature);
 Blynk.virtualWrite(V4, humidity);}
void setup() {
 Blynk.begin(auth,ssid,pass);
 Serial.begin (9600);
 Serial.println ("ACS712 current sensor");
 dht.begin();
 pinMode(ledpin, OUTPUT);
 pinMode(PIRinput, INPUT);
 pinMode(a,OUTPUT);
 pinMode(M3,OUTPUT);
 pinMode(M4,OUTPUT);
 float result;
 int readValue =0;
 int n_samples = 10000;
 for (int i =0; i<n_samples;i++){
 readValue += analogRead(sensorIn); }
 offset = ((readValue)/float(n_samples));
 timer.setInterval(1000L,sendSensorData);}
void loop() {
 Blynk.run();
 timer.run();
}
```

OUTPUT


Energy meter
Voltage (V)
7.58V
Current (Amps)
0.43A
Energy (Ws)
0.00123kWh
Temperature (°C)
33.3°C
Humidity (%)
68%

HARDWARE USED

1. Breadboard
2. Jumper wires
3. ESP32 board
4. Voltage sensor
5. Current sensor
6. DHT11 sensor
7. Relay
8. PIR Motion sensor
9. Lithium battery
10. Battery holder
11. Motor driver board (L293D)
12. Fan and motor

SOFTWARE USED

1. Arduino IDE
2. BLYNK IoT

ARDUINO

Blynk

FUTURE ADVANCEMENTS

1. Avoiding Peak Tariffs during Peak Hours
2. Real-Time Monitoring of Equipment Health
3. Implementation of Overvoltage Protection
4. Integration of Overcurrent Protection Measures
5. Enhancing Preventive Maintenance and Profit Maximization for AMC Companies through IoT-Based Data Collection and Cloud Storage
6. Real time monitoring the life of equipment not just in terms of hours but in terms of energy gives much higher accuracy in reliability prediction

REFERENCE

- *“ESP32 Pinout Reference Guide” - https://www.tech-sparks.com/esp32-pinout-reference-gpios/*
- *“ESP32 PWM with Arduino IDE (Analog Output)” - https://randomnerdtutorials.com/esp32-pwm-arduino-ide/*
- *“Arduino L298n Motor Driver control Tutorial, Speed & Direction, PWM” - https://www.electroniclinic.com/arduino-l298n-motor-driver-control-tutorial-speed-direction-pwm/*
- *“L298N Motor Driver – Arduino Interface, How It Works, Codes, Schematics” - https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/*
- *“ESP32 Pin Configuration” - https://lastminuteengineers.com/wp-content/uploads/iot/ESP32-Pinout.png*
- *“ESP32-WROOM-DA-DEVELOPMENT-BOARD - Pin Configuration” - https://www.techtonions.com/wp-content/uploads/2021/11/ESP32-WROOM-DA-DEVELOPMENT-BOARD-scaled.webp*

THANK YOU