CODE

Arduino Mega: -

```
#include <Wire.h>          // Library for I2C communication
#include <RTClib.h>        // Library for Real-Time Clock (RTC)
#include <DHT.h>           // Library for DHT sensor
#include <PZEM004Tv30.h>   // Library for PZEM-004T energy meter
// Initialize RTC and DHT Sensor
RTC_DS3231 rtc;
#define DHTPIN 2           // DHT sensor connected to pin 2
#define DHTTYPE DHT11      // Define sensor type as DHT11
DHT dht(DHTPIN, DHTTYPE);
// Define Relay and Sensor Pins
const int solarRelay = 4;
const int gridRelay = 8;
const int batteryChargeRelay = 5;
const int voltagePin = A2;         // Solar panel voltage sensor
const int batteryVoltagePin = A1;  // Battery voltage sensor
const int watSensPower = 3;        // Water sensor power pin
const int watSensPin = A0;         // Water level sensor pin
const int ldrPin = A3;             // Light Dependent Resistor (LDR) pin
const int HL1 = 13;                // Light relay control
const int R1A = 11;                // Room 1 AC relay control
const int R2A = 9;                 // Room 2 AC relay control
const int Pump = 7;                // Water pump relay control
// Voltage Divider Parameters
const float adcResolution = 1023.0;     // ADC resolution for analogRead()
const float dividerVoltageMax = 5.0;    // Maximum voltage read by the ADC
const float scaleFactor = 10.0;         // Scale factor for solar voltage measurement
const float scaleFactor1 = 3.0;         // Scale factor for battery voltage measurement
const float batteryFullVoltage = 13.8;  // Battery full charge voltage
```

```
// Define Hardware Serial Ports for PZEM & ESP32
#define PZEM_SERIAL Serial1   // Serial1 for PZEM energy meter
#define ESP_SERIAL Serial2    // Serial2 for ESP32 communication
// Initialize PZEM-004T Energy Meter
PZEM004Tv30 pzem(&PZEM_SERIAL);
void setup() {
   Serial.begin(115200);      // Start serial communication for debugging
   ESP_SERIAL.begin(115200);  // Start serial communication with ESP32
   PZEM_SERIAL.begin(9600);   // Start serial communication with PZEM energy meter
   Wire.begin();              // Start I2C communication
   dht.begin();               // Start DHT sensor
   // Initialize RTC
   if (!rtc.begin()) {
      Serial.println("Couldn't find RTC");
      while (1);  // Stop execution if RTC is not found
   }
   if (rtc.lostPower()) {  // Reset RTC if power was lost
      Serial.println("RTC lost power, setting the time...");
      rtc.adjust(DateTime(_DATE, __TIME_));
   }
   // Set pin modes
   pinMode(solarRelay, OUTPUT);
   pinMode(gridRelay, OUTPUT);
   pinMode(batteryChargeRelay, OUTPUT);
   pinMode(watSensPower, OUTPUT);
   pinMode(HL1, OUTPUT);
   pinMode(R1A, OUTPUT);
   pinMode(R2A, OUTPUT);
   pinMode(Pump, OUTPUT);
   // Initialize all relays and sensors to LOW (OFF state)
```

```
    digitalWrite(solarRelay, LOW);
    digitalWrite(gridRelay, LOW);
    digitalWrite(batteryChargeRelay, LOW);
    digitalWrite(watSensPower, LOW);
    digitalWrite(HL1, LOW);
    digitalWrite(R1A, LOW);
    digitalWrite(R2A, LOW);
    digitalWrite(Pump, LOW);
     Serial.println("System Initialized");
}
void loop() {
    // Read current time from RTC
    DateTime now = rtc.now();
    // Read and calculate solar voltage
    int solarAnalogValue = analogRead(voltagePin);
    float solarVoltage = (solarAnalogValue / adcResolution) * dividerVoltageMax *
scaleFactor;
    // Read and calculate battery voltage
    int batteryAnalogValue = analogRead(batteryVoltagePin);
    float batteryVoltage = (batteryAnalogValue / adcResolution) * dividerVoltageMax *
scaleFactor1;
    // Read and map water level (0-100%)
    int waterLevel = map(analogRead(watSensPin), 0, 490, 0, 100);
    waterLevel = constrain(waterLevel, 0, 100);
    // Read temperature and humidity from DHT11 sensor
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();
    if (isnan(temperature) || isnan(humidity)) temperature = 26;  // Default temperature if
sensor fails
    // Read light intensity from LDR sensor
    int ldrValue = analogRead(ldrPin);
```

```
    // Read electrical parameters from PZEM-004T energy meter
    float voltage = pzem.voltage();
    float current = pzem.current();
    float power = pzem.power();
    float energy = pzem.energy();
    // Logic to control relays based on sensor values and time
    bool solarActive = (now.hour() >= 10 && now.hour() < 14 && solarVoltage > 12.0);  // Solar active between 10AM-2PM
    bool gridActive = !solarActive;  // Grid active when solar is inactive
    bool batteryCharging = (now.hour() >= 10 && now.hour() < 18 && batteryVoltage < batteryFullVoltage);  // Charge battery if below full voltage
    bool pumpOn = (now.hour() == 18 && now.minute() < 45) || (waterLevel <= 16);  // Turn on pump at 6PM or if water level is low
    bool acRoom1 = (now.hour() >= 10 && now.hour() < 18 && temperature > 25);  // Room 1 AC ON if temp > 25°C (10AM-6PM)
    bool acRoom2 = (now.hour() >= 10 && now.hour() < 18 && temperature > 25);  // Room 2 AC ON if temp > 25°C (10AM-6PM)
    bool lightOn = (now.hour() >= 10 && now.hour() < 18 && ldrValue < 600);  // Lights ON in low light conditions (10AM-6PM)
    // Control relays based on logic
    digitalWrite(solarRelay, solarActive);
    digitalWrite(gridRelay, gridActive);
    digitalWrite(batteryChargeRelay, batteryCharging);
    digitalWrite(Pump, pumpOn);
    digitalWrite(R1A, acRoom1);
    digitalWrite(R2A, acRoom2);
    digitalWrite(HL1, lightOn);
    // Print values for debugging
    Serial.print("Time: "); Serial.print(now.hour()); Serial.print(":"); Serial.println(now.minute());
    Serial.print("Solar Voltage: "); Serial.println(solarVoltage);
    Serial.print("Battery Voltage: "); Serial.println(batteryVoltage);
    Serial.print("Water Level: "); Serial.println(waterLevel);
```

```
  Serial.print("Temperature: "); Serial.println(temperature);
  Serial.print("Humidity: "); Serial.println(humidity);
  Serial.print("Light Intensity: "); Serial.println(ldrValue);
  Serial.print("Voltage: "); Serial.println(voltage);
  Serial.print("Current: "); Serial.println(current);
  Serial.print("Power: "); Serial.println(power);
  Serial.print("Energy: "); Serial.println(energy);
  Serial.print("Solar Relay: "); Serial.println(solarActive);
  Serial.print("Grid Relay: "); Serial.println(gridActive);
  Serial.print("Battery Charging: "); Serial.println(batteryCharging);
  Serial.print("Pump: "); Serial.println(pumpOn);
  Serial.print("AC Room 1: "); Serial.println(acRoom1);
  Serial.print("AC Room 2: "); Serial.println(acRoom2);
  Serial.print("Light: "); Serial.println(lightOn);
  Serial.println("-----------------------------");
  // Send sensor data to ESP32 via Serial2
  String data = String(batteryVoltage) + "," + String(waterLevel) + "," +
                String(temperature) + "," +
                String(voltage) + "," + String(current) + "," + String(energy) + "," +
                String(gridActive) + "," + String(batteryCharging) + "," +
                String(pumpOn) + "," + String(acRoom1) + "," + String(lightOn);
  ESP_SERIAL.println(data);  // Send formatted data to ESP32
  delay(5000);  // Wait 5 seconds before next loop iteration
}
```

For ESP32: -

```
#define BLYNK_TEMPLATE_ID "TMPL3edcDWsYN"
#define BLYNK_TEMPLATE_NAME "relayControl"
#define BLYNK_AUTH_TOKEN "61i86q5iCfXwIWOTWxZiOavPxK2bcWpd"
#include <WiFi.h>
```

```
#include <BlynkSimpleEsp32.h>
// WiFi Credentials
char ssid[] = "Anju";
char pass[] = "12345677";
// Relay Pins (ESP32 GPIOs)
#define RELAY_1 5
#define RELAY_2 18
#define RELAY_3 19
#define RELAY_4 21
#define RELAY_5 22
#define RELAY_6 23
#define RELAY_7 25
#define RELAY_8 26
#define RELAY_9 27
// Serial Communication with Arduino Mega
#define ARDUINO_SERIAL Serial2
// Blynk Timer to handle periodic tasks
BlynkTimer timer;
// Function to read sensor data from Arduino Mega and send it to Blynk
void readAndSendSensorData() {
  if (ARDUINO_SERIAL.available() > 0) {
    String data = ARDUINO_SERIAL.readStringUntil('\n');
    Serial.println("\n--- Received Data ---");
    Serial.println("Raw Data: " + data);
    // Print HEX values of received data (for debugging)
    Serial.print("HEX Data: ");
    for (size_t i = 0; i < data.length(); i++) {
      Serial.print("0x");
      Serial.print(data[i], HEX);
      Serial.print(" ");
```

```
}
Serial.println("\n----------------------");
// Variables to store parsed data
float batteryVoltage, waterLevel, temperature, voltage, current, energy;
int gridStatus, batteryCharging, pumpStatus, acControl, lightStatus;
// Parsing the data received from Arduino Mega
int parsed = sscanf(data.c_str(), "%f,%f,%f,%f,%f,%f,%d,%d,%d,%d,%d",
    &batteryVoltage, &waterLevel, &temperature, &voltage, &current, &energy,
    &gridStatus, &batteryCharging, &pumpStatus, &acControl, &lightStatus);
// Check if parsing was successful
if (parsed == 11) {
    Serial.println("✅ Parsed successfully!");
    // Print parsed values
    Serial.println("Battery Voltage: " + String(batteryVoltage) + "V");
    Serial.println("Water Level: " + String(waterLevel) + "%");
    Serial.println("Temperature: " + String(temperature) + "°C");
    Serial.println("Voltage: " + String(voltage) + "V");
    Serial.println("Current: " + String(current) + "A");
    Serial.println("Energy: " + String(energy) + "kWh");
    Serial.println("Grid Status: " + String(gridStatus));
    Serial.println("Battery Charging: " + String(batteryCharging));
    Serial.println("Pump Status: " + String(pumpStatus));
    Serial.println("AC Control: " + String(acControl));
    Serial.println("Light Status: " + String(lightStatus));
    // Send data to Blynk Virtual Pins
    Blynk.virtualWrite(V0, batteryVoltage);
    Blynk.virtualWrite(V1, waterLevel);
    Blynk.virtualWrite(V2, temperature);
    Blynk.virtualWrite(V3, voltage);
    Blynk.virtualWrite(V4, current);
```

```
            Blynk.virtualWrite(V5, energy);
            Blynk.virtualWrite(V6, gridStatus);
            Blynk.virtualWrite(V7, batteryCharging);
            Blynk.virtualWrite(V8, pumpStatus);
            Blynk.virtualWrite(V9, acControl);
            Blynk.virtualWrite(V10, lightStatus);
        } else {
            Serial.println("❌ Parsing failed! Check format. Received: " + data);
        }
    }
}
// Relay Control from Blynk
BLYNK_WRITE(V10) { digitalWrite(RELAY_1, param.asInt()); }
BLYNK_WRITE(V11) { digitalWrite(RELAY_2, param.asInt()); }
BLYNK_WRITE(V12) { digitalWrite(RELAY_3, param.asInt()); }
BLYNK_WRITE(V13) { digitalWrite(RELAY_4, param.asInt()); }
BLYNK_WRITE(V14) { digitalWrite(RELAY_5, param.asInt()); }
BLYNK_WRITE(V15) { digitalWrite(RELAY_6, param.asInt()); }
BLYNK_WRITE(V16) { digitalWrite(RELAY_7, param.asInt()); }
BLYNK_WRITE(V17) { digitalWrite(RELAY_8, param.asInt()); }
BLYNK_WRITE(V18) { digitalWrite(RELAY_9, param.asInt()); }
void setup() {
    Serial.begin(115200);
    Serial.println("\nESP32 is starting...");
    // Initialize Serial Communication with Arduino Mega
    ARDUINO_SERIAL.begin(115200, SERIAL_8N1, 16, 17); // RX = GPIO16, TX = GPIO17
    Serial.println("Serial2 (Arduino Mega) initialized at 115200 baud.");
    // Set Relay Pins as Outputs & Turn OFF initially
    pinMode(RELAY_1, OUTPUT); digitalWrite(RELAY_1, LOW);
    pinMode(RELAY_2, OUTPUT); digitalWrite(RELAY_2, LOW);
```

```
    pinMode(RELAY_3, OUTPUT); digitalWrite(RELAY_3, LOW);
    pinMode(RELAY_4, OUTPUT); digitalWrite(RELAY_4, LOW);
    pinMode(RELAY_5, OUTPUT); digitalWrite(RELAY_5, LOW);
    pinMode(RELAY_6, OUTPUT); digitalWrite(RELAY_6, LOW);
    pinMode(RELAY_7, OUTPUT); digitalWrite(RELAY_7, LOW);
    pinMode(RELAY_8, OUTPUT); digitalWrite(RELAY_8, LOW);
    pinMode(RELAY_9, OUTPUT); digitalWrite(RELAY_9, LOW)
    // Connect to WiFi and initialize Blynk
    Serial.println("Connecting to WiFi...");
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
    Serial.println("Connected to WiFi!");
    // Set a timer to read sensor data every 5 seconds
    timer.setInterval(5000L, readAndSendSensorData);
    Serial.println("Data read interval set to 5 seconds.");
}
void loop() {
    Blynk.run(); // Run Blynk service
    timer.run(); // Run the timer
    delay(5000);  // Prevents watchdog reset
}
```