Edge-IoT Based Smart Energy Management System

GROUP: 05

Prepared By :

Names of Group Members:

. Adithyakrishnan P K
. Abin Abraham
. Abhijith Santhosh

Introduction:

The Edge-IoT Based Smart Energy Management System is designed to monitor and control the power consumption of multiple rooms or sections in a home or building.
Each room is equipped with an end node consisting of voltage and current sensors to measure real-time electrical parameters. These end devices send data to a central Edge Device using MQTT protocol over a local network.

The Edge Device performs data processing tasks such as:

Calculating power (W) and energy (kWh),

Detecting anomalies such as overvoltage or overcurrent, and

Sending control commands (e.g., relay OFF) back to the nodes.

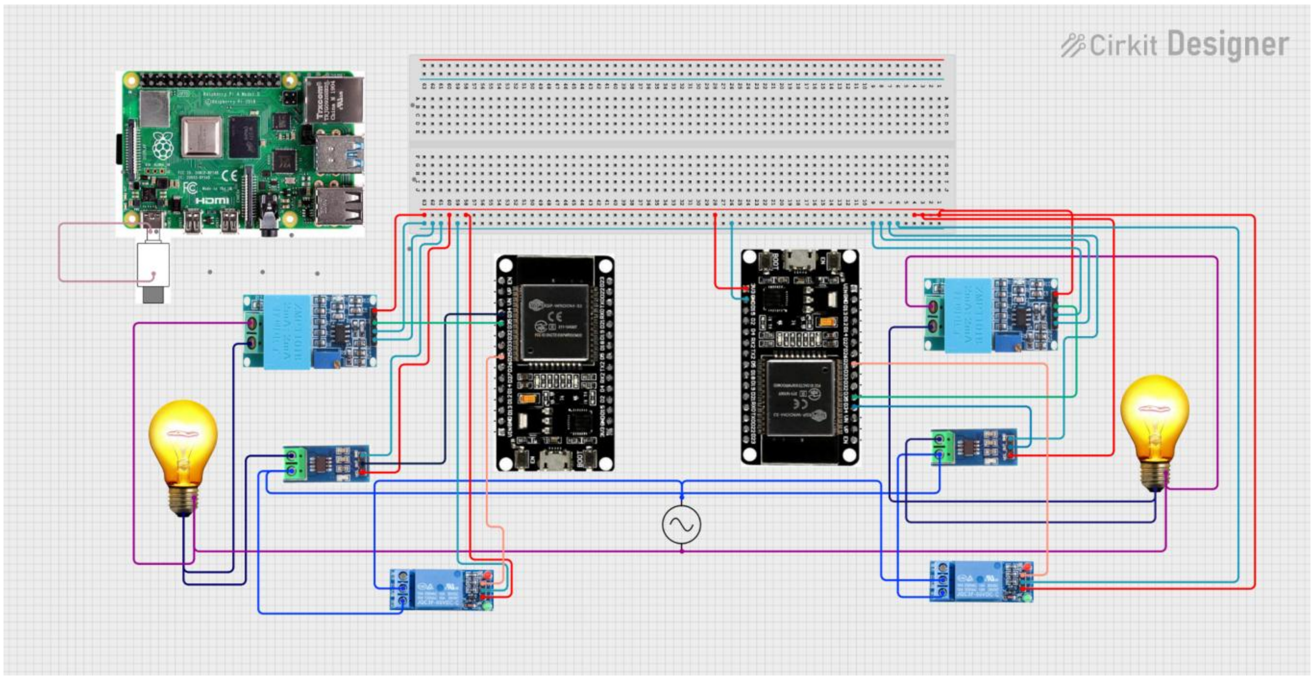
Processed data is then uploaded to the Adafruit IO Cloud Dashboard, where the user can visualize power usage, total energy, and system status.
This setup allows real-time monitoring, efficient power management, and improved safety in home energy systems.

Components/Modules:

1. ESP32 (End Nodes) – Used to collect sensor data (voltage and current) and control relays.
2. Raspberry Pi (Edge Device) – Acts as the MQTT broker and local data processor; calculates power and energy.
3. ACS712 Current Sensor (30A) – Measures current flow through each load.
4. ZMPT101B Voltage Sensor – Measures AC voltage levels accurately.
5. Relay Module – Controls each connected load based on commands from the edge device.
6. Adafruit IO Cloud Platform – Displays real-time data such as node power and total energy.

7. Wi-Fi Network – Connects all devices (end nodes, edge, and cloud).
8. Python (Edge Script) – Performs MQTT data handling, power calculations, and Adafruit IO data publishing.

Connection Diagram:



Working:

1. Each end node continuously measures voltage and current values using the ACS712 and ZMPT101B sensors.
2. These readings are transmitted to the Edge Device over MQTT.
3. The Edge Device (Raspberry Pi) calculates:
    - ➢ Power (W) = Voltage × Current
    - ➢ Energy (kWh) = ∑(Power × Time) / 1000
4. If any abnormal condition (overvoltage or overcurrent) is detected, the edge device sends a control command to the respective node to turn off its relay.

5. All computed data (node1 power, node2 power, total energy) is published to Adafruit IO, where users can monitor energy usage in real time through interactive gauges and charts.
6. The system supports scaling — more rooms or appliances can be added simply by introducing new end nodes.

Program:

➢ Node1:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "ACS712.h"
#include <ZMPT101B.h>

// WiFi credentials
const char* ssid = "qwerty";
const char* password = "12345678";

// Edge device IP (MQTT broker)
const char* mqtt_server = "192.168.6.213";

WiFiClient espClient;
PubSubClient client(espClient);

#define RELAY_PIN 25

// Sensors
ACS712 ACS(34, 3.3, 4095, 65);
ZMPT101B voltageSensor(35, 50.0);

float current = 0;
float volt = 0;

void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) message += (char)payload[i];
  if (message == "OFF") digitalWrite(RELAY_PIN, LOW);
  if (message == "ON") digitalWrite(RELAY_PIN, HIGH);
}

void reconnect() {
  while (!client.connected()) {
    if (client.connect("node1")) {
      client.subscribe("ems/node1/control");
    } else {
      delay(2000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH); // Relay default ON

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

float readCurrent() {
  float average = 0;
  for (int i = 0; i < 100; i++) {
    average += ACS.mA_AC();
    delay(2);
  }
  float mA = average / 100.0;
  return (mA > 10) ? mA : 0; // filter small noise
```

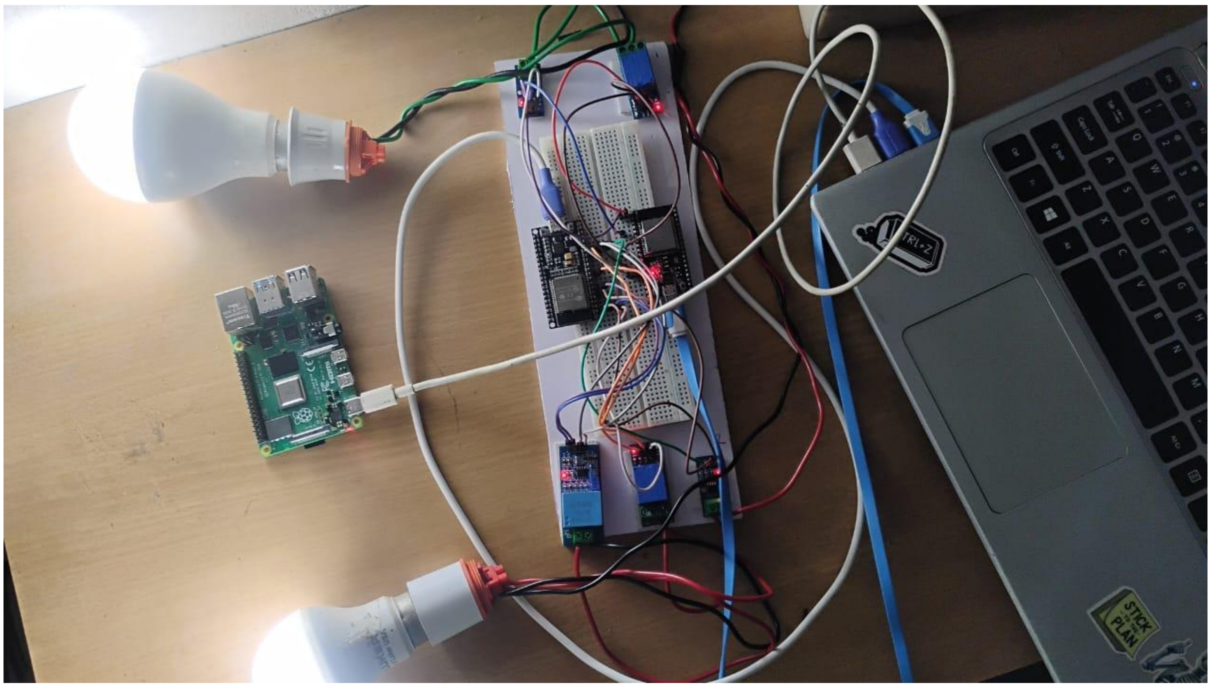
```
}

float readVoltage() {
  float voltage = voltageSensor.getRmsVoltage();
  return (voltage > 50) ? voltage : 0;
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();

  current = readCurrent();
  volt = readVoltage();

  // Build JSON payload
  String payload = "{\"voltage\":" + String(volt,2) +
                   ",\"current\":" + String(current,2) +"}";

  client.publish("ems/node1/data", payload.c_str());

  delay(2000); // publish every 2s
}
```

Node 2:

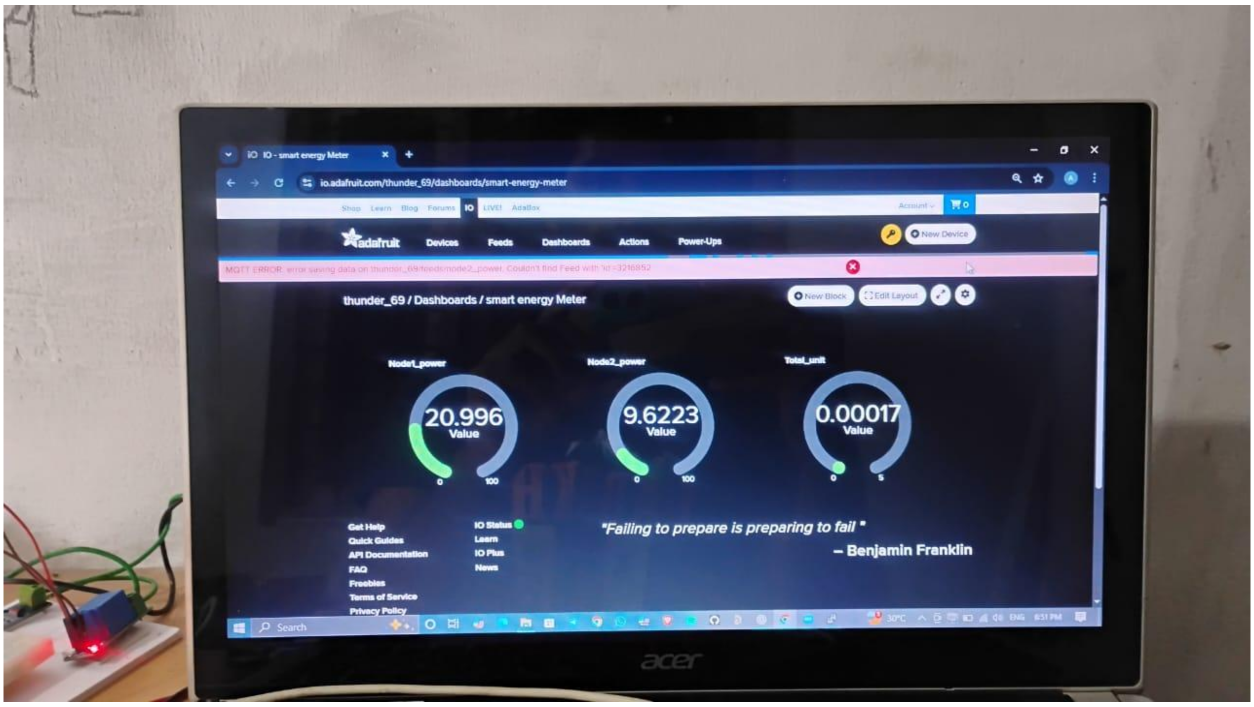
```
#include <WiFi.h>
#include <PubSubClient.h>
#include "ACS712.h"
#include <ZMPT101B.h>

// WiFi credentials
const char* ssid = "qwerty";
const char* password = "12345678";

// Edge device IP (MQTT broker)
const char* mqtt_server = "192.168.6.213";

WiFiClient espClient;
PubSubClient client(espClient);

#define RELAY_PIN 25 // different relay pin for Node2

// Sensors
ACS712 ACS(34, 3.3, 4095, 185);
ZMPT101B voltageSensor(35, 50.0);

float current = 0;
float volt = 0;


void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) message += (char)payload[i];

  if (message == "OFF") digitalWrite(RELAY_PIN, LOW);
  if (message == "ON") digitalWrite(RELAY_PIN, HIGH);
}

void reconnect() {
  while (!client.connected()) {
    if (client.connect("node2")) {
      client.subscribe("ems/node2/control");
    } else {
      delay(2000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH); // Relay default ON

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) delay(500);

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

float readCurrent() {
  int noise = ACS.getNoisemV();
  float average = 0;
  for (int i = 0; i < 200; i++) average += ACS.mA_AC();
  float mA = average / 200.0;
  return (mA > 1) ? mA : 0;
}

float readVoltage() {
  float voltage = voltageSensor.getRmsVoltage();
  return (voltage > 50) ? voltage : 0;
}

void loop() {
  if (!client.connected()) reconnect();
  client.loop();
  current = readCurrent();
  volt = readVoltage();



  // Build JSON payload
  String payload = "{\"voltage\":" + String(volt,2) +
                   ",\"current\":" + String(current,2) + "}";

  client.publish("ems/node2/data", payload.c_str());

  delay(2000); // publish every 2s
}
```

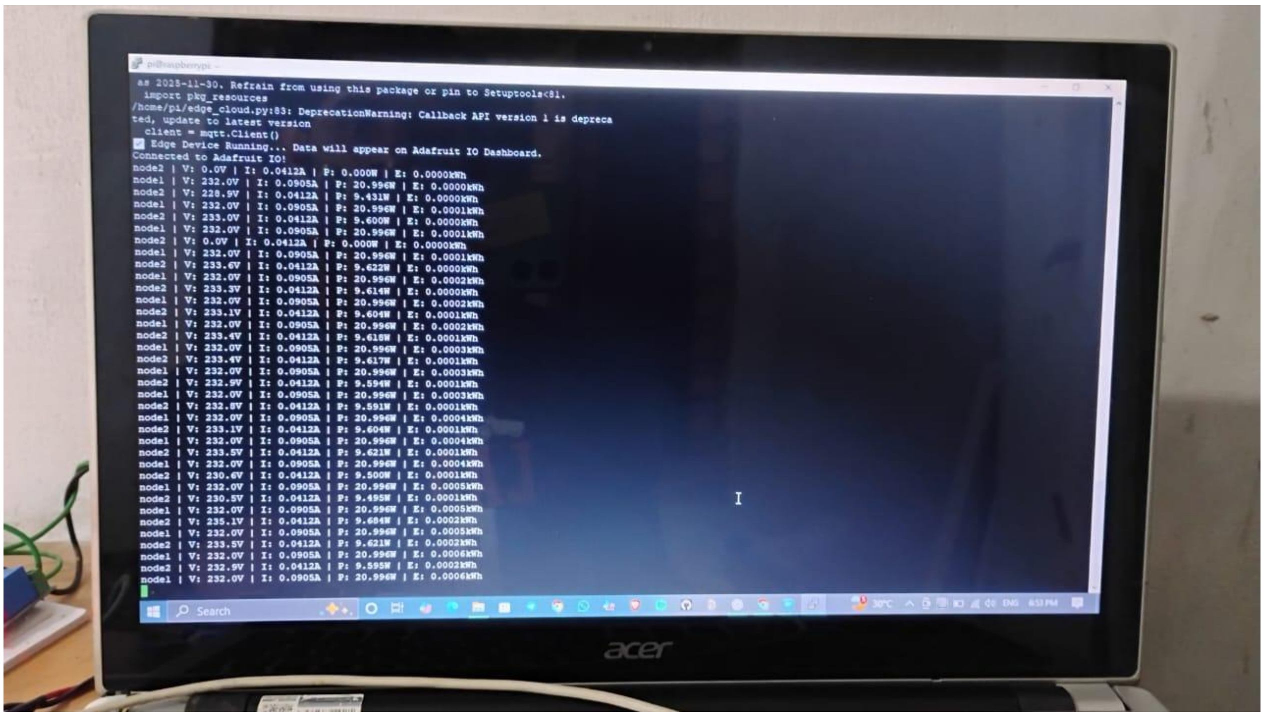
Results:

- Real-time readings of node1 power, node2 power, and total energy are displayed on the Adafruit IO dashboard.
- The system successfully detects anomalies and automatically controls the relays.
- Demonstrated accurate energy monitoring for multiple nodes within a local IoT network.



[redacted]


IO - smart energy Meter
io.adafruit.com/thunder_69/dashboards/smart-energy-meter
Shop Learn Blog Forums IO LIVE! AdaBox
adafruit
Devices Feeds Dashboards Actions Power-Ups
New Device
MQTT ERROR: error saving data on thunder_69/feeds/node2_power. Couldn't find Feed with 'id'=3216852
thunder_69 / Dashboards / smart energy Meter
New Block
Edit Layout
Node1_power
20.996
Value
0
100
Node2_power
9.6223
Value
0
100
Total_unit
0.00017
Value
0
5
"Failing to prepare is preparing to fail "
– Benjamin Franklin
Get Help
Quick Guides
API Documentation
FAQ
Freebies
Terms of Service
Privacy Policy
IO Status
Learn
IO Plus
News
Search
30°C
6:51 PM
acer



```
as 2025-11-30. Refrain from using this package or pin to Setuptools<81.
  import pkg_resources
/home/pi/edge_cloud.py:83: DeprecationWarning: Callback API version 1 is depreca
ted, update to latest version
  client = mqtt.Client()
✅ Edge Device Running... Data will appear on Adafruit IO Dashboard.
Connected to Adafruit IO!
node2 | V: 0.0V | I: 0.0412A | P: 0.000W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0000kWh
node2 | V: 228.9V | I: 0.0412A | P: 9.431W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0001kWh
node2 | V: 233.0V | I: 0.0412A | P: 9.600W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0001kWh
node2 | V: 0.0V | I: 0.0412A | P: 0.000W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0001kWh
node2 | V: 233.6V | I: 0.0412A | P: 9.622W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0002kWh
node2 | V: 233.3V | I: 0.0412A | P: 9.614W | E: 0.0000kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0002kWh
node2 | V: 233.1V | I: 0.0412A | P: 9.604W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0002kWh
node2 | V: 233.4V | I: 0.0412A | P: 9.618W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0003kWh
node2 | V: 233.4V | I: 0.0412A | P: 9.617W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0003kWh
node2 | V: 232.9V | I: 0.0412A | P: 9.594W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0003kWh
node2 | V: 232.8V | I: 0.0412A | P: 9.591W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0004kWh
node2 | V: 233.1V | I: 0.0412A | P: 9.604W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0004kWh
node2 | V: 233.5V | I: 0.0412A | P: 9.621W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0004kWh
node2 | V: 230.6V | I: 0.0412A | P: 9.500W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0005kWh
node2 | V: 230.5V | I: 0.0412A | P: 9.495W | E: 0.0001kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0005kWh
node2 | V: 235.1V | I: 0.0412A | P: 9.684W | E: 0.0002kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0005kWh
node2 | V: 233.5V | I: 0.0412A | P: 9.621W | E: 0.0002kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0006kWh
node2 | V: 232.9V | I: 0.0412A | P: 9.595W | E: 0.0002kWh
node1 | V: 232.0V | I: 0.0905A | P: 20.996W | E: 0.0006kWh
```