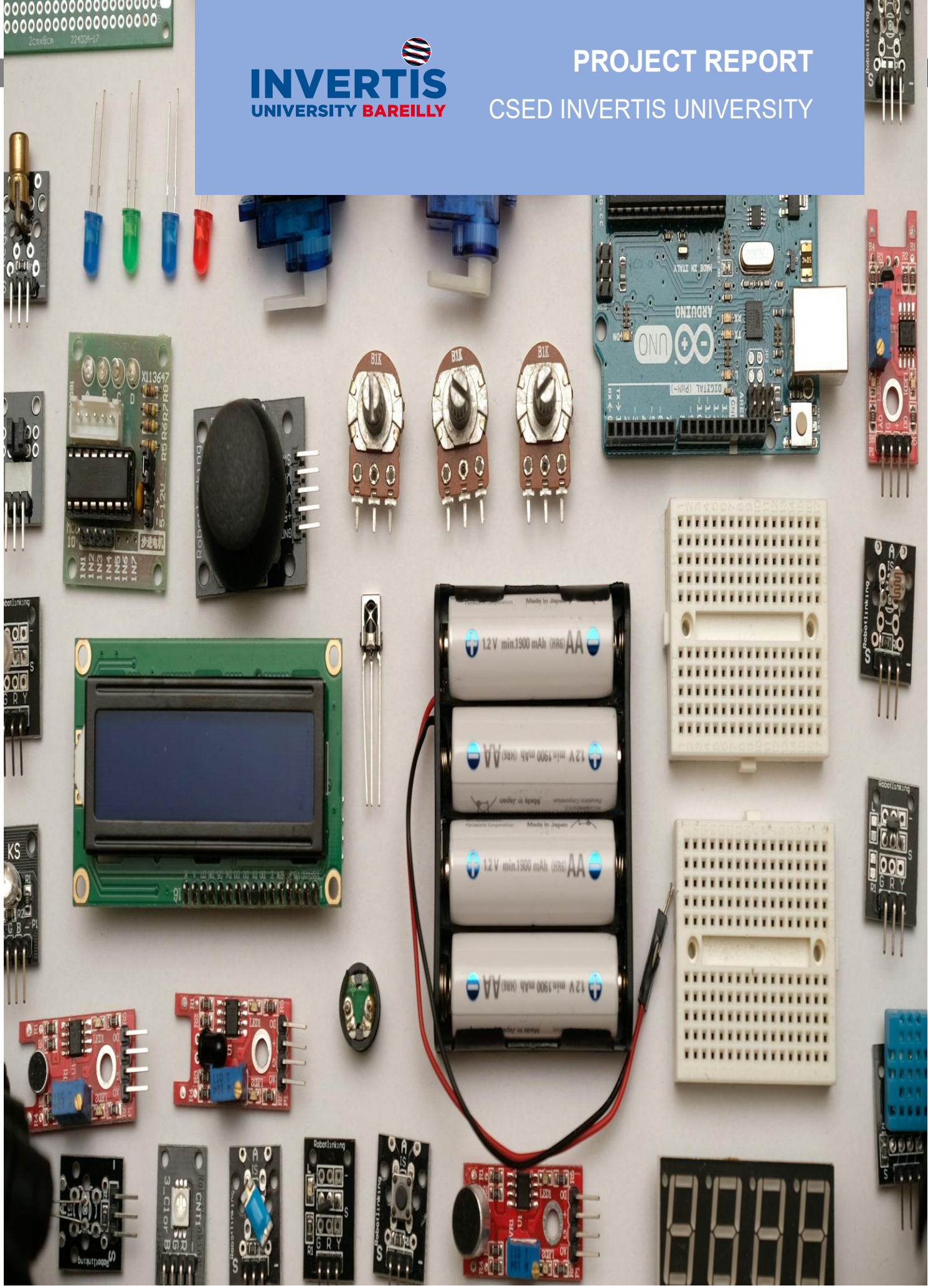
INVERTIS UNIVERSITY BAREILLY

PROJECT REPORT

CSED INVERTIS UNIVERSITY



DCS
Power To Design The Desired

CENTER FOR SKILL AND ENTREPRENEURSHIP DEVELOPMENT (CSED)

INDUSTRIAL INTERNET OF THINGS

PROGRAM CODE: **IIOT-2**

PROGRAM NAME: **FUNDAMENTALS OF IOT**

PROJECT NAME: **SMART INDOOR ENERGY OPTIMIZATION**

MENTOR NAME: **MR. RAHUL CHAPLE**

DCS
Power To Design The Desired

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

COURSE NAME: **B.TECH(ARTIFICAL INTELLIGENCE)**

TEAM NAME: **2AICCT3**

TEAM MEMBERS:	STUDENT ID:
1.Dhruv Maheshwari(LEAD)	BTAI2024048
2.Monty Gaurav	BTAI2024045
3.Ayush	BTAI2024054
4.Saksham Shrivastav	BTAI2024053
5.Adnan Ansari	BTAI2024055
6.Abdullah Khan	BTAI2024043
7.Anjali Singh	BTAI2024059
8.Dhriti Dixit	BTAI2024049
9.Ankit	BTAI2024042
10.Nikhil Sahu	BTAI2024046
11.Ankur Pratap Singh	BTAI2024050
12.Naitik Gupta	BTAI2024047
13.Arpit Singh	BTAI2024044
14.Zaid Ahmad	BTAI2024058
15.Anubhav Yadav	BTAI2024057
16.Dakash Yadav	BTAI2024056

INDEX

Introduction:

In this project, we design a smart, energy-efficient system that automates the control of a relay (e.g., for lighting or appliances) based on real-time environmental conditions and device connectivity status. The system is built around an **ESP32 microcontroller**, integrating a **PIR sensor** for motion detection and an **LDR (Light Dependent Resistor)** for light intensity sensing.

The ESP32 connects to a Wi-Fi network and communicates with the **ThingsBoard IoT platform**, which serves as the cloud interface for remote monitoring and control. The system's behavior is visually indicated using onboard LEDs:

- **Red LED**: ESP32 is not connected to Wi-Fi.
- **Blue LED**: ESP32 connected to Wi-Fi but not to ThingsBoard cloud.
- **Green LED**: ESP32 connected successfully to both Wi-Fi and ThingsBoard.

The core working logic ensures that:

- The relay is **activated (ON)** only when the environment is dark (**LDR detects low light**) **and** motion is detected (**PIR sensor is triggered**).
- If no motion is detected or the environment is sufficiently lit, the **relay remains OFF**, saving energy.
- All device states and sensor readings are continuously monitored and can be visualized through the ThingsBoard dashboard.

This project emphasizes **energy optimization**, **IoT connectivity**, and **automation**, making it ideal for smart homes, offices, or industrial applications where intelligent energy usage is essential.

Need of Project:

In today's world, the rising demand for energy and increasing environmental concerns make **energy efficiency** a top priority. A significant amount of electricity is wasted daily due to human negligence, such as leaving lights and appliances ON unnecessarily when they are not needed. Manual control is unreliable and often forgotten, especially in public or shared spaces.

This project addresses these challenges by providing an **automated, intelligent energy management system** that optimizes electricity usage based on real-time environmental conditions and human presence. By using a combination of **motion detection (PIR sensor)** and **ambient light measurement (LDR sensor)**, the system ensures that energy is only consumed when absolutely necessary.

Furthermore, integrating **IoT connectivity** through **Wi-Fi** and **ThingsBoard** enables remote monitoring, real-time status visualization, and smart energy-saving calculations. This enhances both **user awareness** and **system efficiency**, supporting the broader goals of **sustainability**, **cost reduction**, and **smart infrastructure development**.

Thus, this project is essential for:

- Reducing unnecessary power consumption
- Promoting smarter and automated building management
- Supporting energy conservation efforts
- Lowering operational costs in homes, offices, and industries

Problem Statement:

In indoor areas, energy consumption is often inefficient due to manual control of lighting, air conditioning, and other electrical devices. This leads to unnecessary energy wastage when indoors are unoccupied or when environmental conditions do not require full energy usage.

This project aims to develop an Industrial Internet of Things (IIoT)-based Smart Indoor Energy Optimization System for university classrooms. The system will utilize smart sensors, real-time data analytics, and automation to optimize energy consumption.

By integrating occupancy detection, ambient light sensing, and temperature control, the system will dynamically adjust energy usage, reducing wastage while maintaining a comfortable learning environment. This solution will help universities reduce energy costs, lower their carbon footprint, and promote sustainable energy use through smart, data-driven decision-making.

Objective of the Project:

The primary objectives of the Smart Indoor Energy Optimization System are:

Automate Lighting Control:

Design a system that automatically turns lights ON or OFF based on human presence(PIR sensor) and ambient light conditions(LDR sensor), minimizing unnescessary energy usage.

1. **Real-Time System Status Monitoring:**

Use an RGB LED to indicate the current system status:

- Red LED: ESP32 not connected to Wi-Fi
- Blue LED: Connected to Wi-Fi but not ThingsBoard
- Green LED: Connected to both Wi-Fi and ThingsBoard

2. **IoT-Based Remote Monitoring:**

Send live data such as relay status and active time to the **ThingsBoard IoT platform** for real-time visualization and tracking.

3. **Energy Usage and Savings Calculation:**

Calculate total energy consumed and energy saved daily, comparing automated control to traditional always-ON lighting systems.

4. **Daily Reset and Data Accuracy:**

Ensure that energy consumption and savings statistics reset every 24 hours for accurate daily reporting.

5. **Cloud Dashboard Visualization:**

Develop an interactive dashboard on ThingsBoard to display:

- Relay ON/OFF status
- Total energy used and saved (in kWh)
- Historical motion detection patterns
- Comparative graphs of automated vs non-automated energy consumption

6. Promote Energy Conservation and Smart Building Initiatives:

Encourage sustainable energy practices through intelligent automation, contributing to smart homes, offices, and environmentally friendly infrastructures.

Methodology:

1. Hardware Components and Configuration

The system is built around the **ESP32 microcontroller**, which integrates both Wi-Fi and GPIO capabilities. The following sensors and modules are connected:

- **PIR Motion Sensor** (PIR_PIN = GPIO14): Detects human motion.
- **Light Sensor (LDR)** (LIGHT_SENSOR_PIN = GPIO26): Measures ambient light level to determine whether it's dark.
- **Relay Module** (RELAY_PIN = GPIO13): Controls the connected load (e.g., light or appliance) based on sensor inputs.
- **RGB LED** (RED_PIN = GPIO25, GREEN_PIN = GPIO33, BLUE_PIN = GPIO32): Provides visual feedback of the system's network and MQTT status.

2. Network and MQTT Communication

The ESP32 connects to a Wi-Fi network using SSID "Wokwi-GUEST", and publishes data to the **ThingsBoard** cloud IoT platform via the MQTT protocol. The MQTT broker details are:

- Server: demo.thingsboard.io
- Port: 1883
- Access Token: Used for device authentication

3. System Logic Flow

The logic of the system operates in the following steps:

a. Sensor Monitoring

- The **PIR sensor** continuously checks for motion.
- The **light sensor** reads ambient light intensity.
- If **motion is detected** and **light level is below a set threshold**, the relay is activated to turn on a light or load.

b. Relay Control Logic

IF motion is detected AND light intensity < threshold:

Turn ON the relay

ELSE:

Turn OFF the relay

The threshold for light intensity is set to a value (e.g., 100) determined through calibration to define "dark" conditions.

c. Time Tracking

Two time counters are maintained:

- onTimeSeconds: Tracks how long the relay remains ON.
- offTimeSeconds: Tracks how long the relay remains OFF.

This is done using millis() to compute the elapsed time in each loop iteration without blocking the program execution.

d. Data Telemetry

Every second, the ESP32 constructs a JSON payload and sends it to ThingsBoard with the following telemetry:

- relay_status: Boolean value indicating if the relay is ON/OFF.
- on_time: Total time (in seconds) the relay has been ON.
- off_time: Total time the relay has been OFF.

This allows remote monitoring and analytics via the ThingsBoard dashboard.

e. RGB LED Status Indicator

The RGB LED provides visual feedback of the system's connectivity status:

- **Red**: Wi-Fi not connected
- **Blue**: Wi-Fi connected, but MQTT not connected
- **Green**: Both Wi-Fi and MQTT connected

4. Loop Operation

The main loop executes the following tasks every second:

- Check MQTT connection and reconnect if needed.
- Read PIR and light sensor values.
- Apply control logic to relay.
- Update ON/OFF timers.
- Send telemetry data to ThingsBoard.
- Update RGB LED based on network status.

Formulas:

- Energy = Power * Time.
- Total energy(kwh) = Power(kw) * 24(hrs).
- Consumed energy(kwh) = Power(kw) * Active relay time(hrs).

Tools and Technologies Used

- **Microcontroller**: ESP32
- **Platform**: Arduino IDE
- **Cloud IoT Platform**: ThingsBoard
- **Communication Protocol**: MQTT
- **Programming Language**: C/C++ (Arduino framework)

Literature Review:

The development of smart energy systems has gained considerable momentum with the rise of IoT-based technologies.

1. **M. Fathi et al., "Smart Energy Management for Smart Homes," *IEEE Access*, 2021.**

Explored **smart home energy management** using IoT devices. Their work highlights the importance of integrating real-time data from motion and light sensors to optimize energy consumption automatically without human intervention.

2. **D. Georgievski et al., "Energy-Efficient Smart Lighting Systems Using IoT," *Future Generation Computer Systems*, 2017.**

Focused on the application of **smart lighting systems**, utilizing **PIR** (Passive Infrared) sensors and **LDRs** (Light Dependent Resistors). They demonstrated that automatic lighting control based on human presence and ambient brightness significantly reduces electricity usage in indoor environments.

3. **S. Madakam et al., "Internet of Things (IoT): A Literature Review," *Journal of Computer and Communications*, 2015.**

Presented a comprehensive **IoT framework** for energy systems. Their review emphasized how **cloud platforms** like ThingsBoard facilitate remote monitoring, control, and real-time decision-making, which are crucial for energy optimization projects.

4. **M. Elamari et al., "A Smart Motion-Sensing Lighting System for Energy Saving," *Journal of Building Engineering*, 2020.**

Introduced a **motion-sensing lighting system** aimed at energy saving, further validating the role of PIR sensors in dynamic environments such as offices and

homes. Their study provided experimental results showing measurable energy savings compared to traditional manual systems.

Lastly, ThingsBoard's official documentation elaborates on how the **open-source IoT platform** supports device communication, telemetry data management, and dashboard visualization, enabling seamless integration between field devices and cloud-based analytics essential for smart energy solutions.

Together, these studies provide a strong foundation for the design and implementation of a **Smart Indoor Energy Optimization System**, combining sensor-based automation with real-time cloud monitoring.

Explanation of Project (Working):

System Functionality

1. Motion and Light Detection

- A **PIR sensor** detects motion within its field of view.
- An **LDR (light sensor)** monitors the surrounding light intensity.
- If **motion is detected and the environment is dark**, the system turns ON the relay, activating the connected device (e.g., a light bulb).

2. Relay Control

- The relay is activated only when both conditions are met:
  - Motion is detected.
  - Ambient light is below a threshold.
- This ensures the system doesn't waste energy by turning on lights during the daytime or when no one is present.

3. Time Tracking

- The system tracks how long the relay remains ON and OFF.
- This data is useful for energy usage analysis and automation tuning.
- Timing is calculated using the millis() function to keep the system responsive and non-blocking.

4. Cloud Connectivity and Telemetry

- The ESP32 connects to the **ThingsBoard IoT platform** using MQTT.
- Every second, it sends a JSON-formatted message containing:
  - Current relay status (true or false)
  - Total time the relay has been ON
  - Total time it has been OFF
- These values can be visualized in real-time on the ThingsBoard dashboard, allowing remote monitoring and analysis.

5. Visual Feedback with RGB LED

- The built-in RGB LED gives users quick insight into the system status:
    - **Red**: Wi-Fi not connected
    - **Blue**: Wi-Fi connected, but MQTT disconnected
    - **Green**: System is fully connected and functional

Block Diagram :

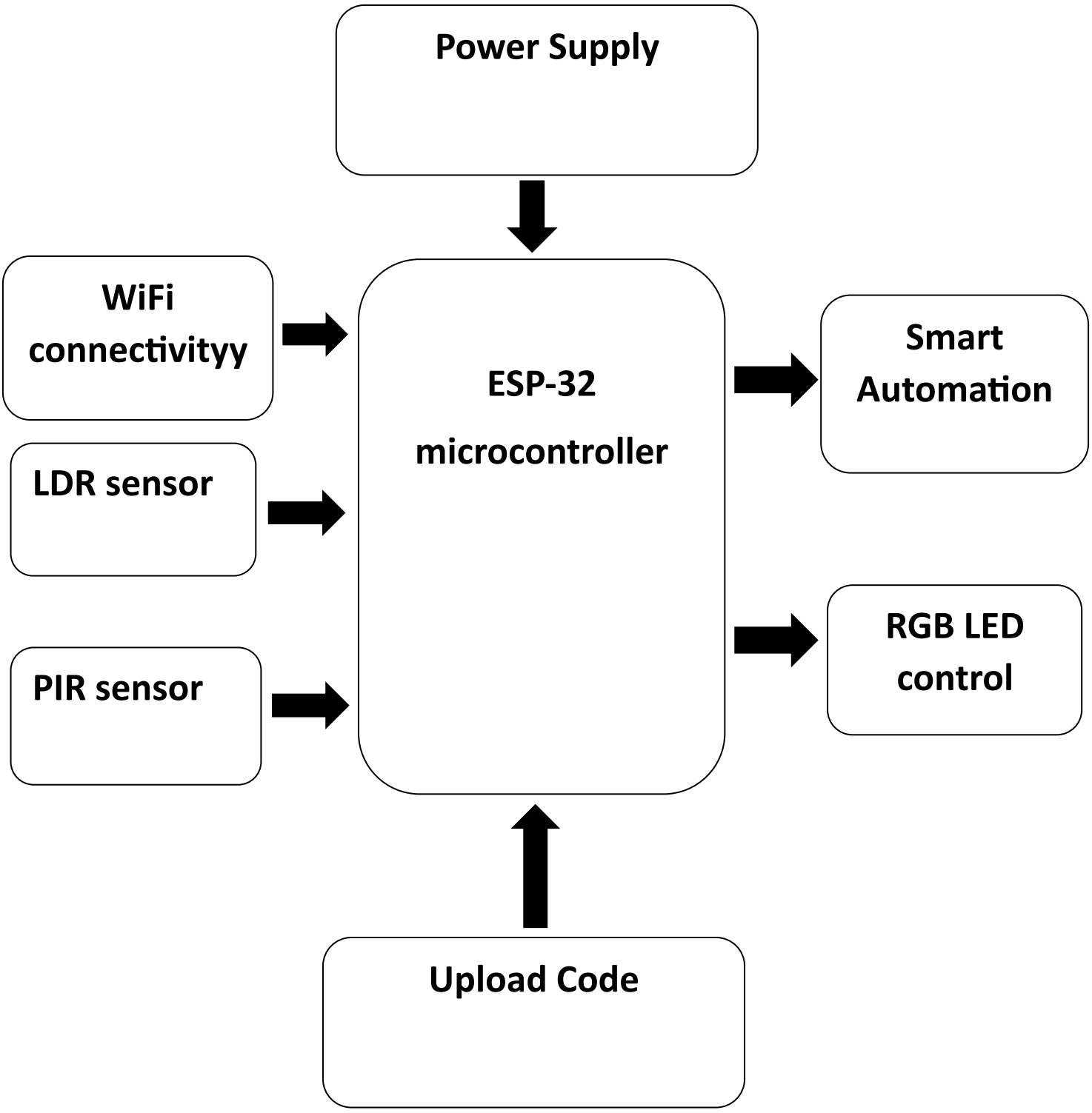


Figure No. 01

Flowchart:

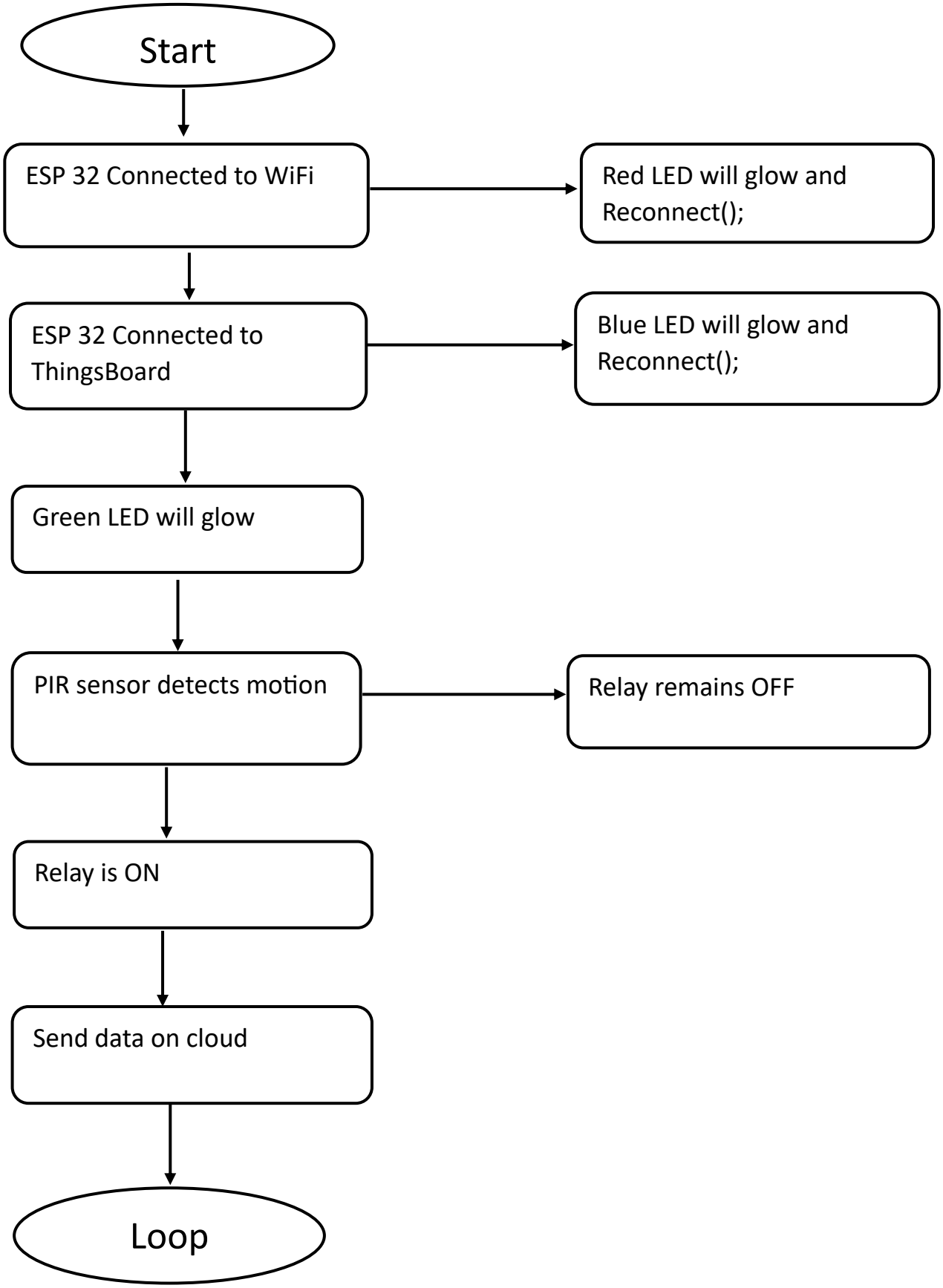


Figure no. 02

Circuit Diagram:

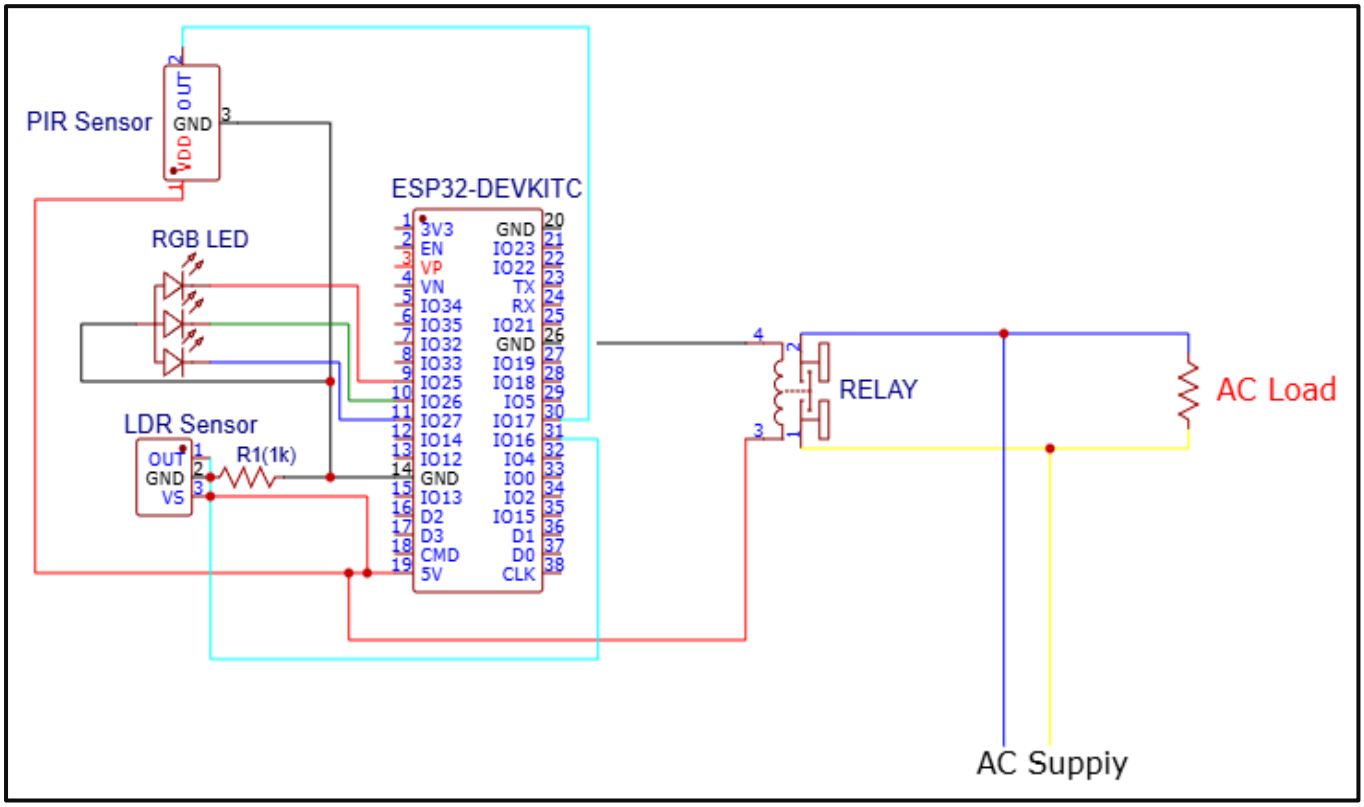


Figure no. 03

Program :

1.Code for microcontroller(ESP32) in Embbed c:

```
#include <WiFi.h>
#include <PubSubClient.h>

// === Pin Definitions ===
#define PIR_PIN 14            // PIR motion sensor input pin
#define LIGHT_SENSOR_PIN 26   // Analog light sensor input pin
#define RELAY_PIN 13          // Relay control pin

#define RED_PIN 25            // RGB LED Red pin
#define GREEN_PIN 33          // RGB LED Green pin
#define BLUE_PIN 32           // RGB LED Blue pin

// === WiFi and MQTT Configuration ===
const char* ssid = "Wokwi-GUEST";                // WiFi SSID
const char* password = "";                       // WiFi Password
const char* mqtt_server = "demo.thingsboard.io";    // ThingsBoard MQTT server
const int mqtt_port = 1883;                      // MQTT port
const char* access_token = "boIQk98csjl1ujKEVOlL";   // Device access token

// === Clients ===
WiFiClient espClient;            // WiFi client
PubSubClient client(espClient);   // MQTT client
```

```
// === Time Tracking Variables ===
float onTimeSeconds = 0.0;       // Time relay has been ON
float offTimeSeconds = 0.0;      // Time relay has been OFF
unsigned long lastUpdateTime = 0; // Last time the status was updated

// === Set RGB LED Color ===
// Parameters: r, g, b (255 = HIGH/on, 0 = LOW/off)
void setRGB(int r, int g, int b) {
  digitalWrite(RED_PIN, r);
  digitalWrite(GREEN_PIN, g);
  digitalWrite(BLUE_PIN, b);
}

// === Update RGB LED to Show Connection Status ===
void updateConnectionLED() {
  if (WiFi.status() != WL_CONNECTED) {
    setRGB(255, 0, 0);  // RED = WiFi not connected
  } else if (!client.connected()) {
    setRGB(0, 0, 255);  // BLUE = MQTT not connected
  } else {
    setRGB(0, 255, 0);  // GREEN = Everything connected
  }
}

// === Send Telemetry Data to ThingsBoard ===
```

```
void sendRelayStatus(bool status,  float time, float falseTime) {
  String payload = "{";
  payload += "\"relay_status\":";
  payload += status ? "true" : "false";
  payload += ",";
  payload += "\"on_time\":";
  payload += String(time, 3);
  payload += ",";
  payload += "\"false_time\":";
  payload += String(falseTime, 3);
  payload += "}";

  Serial.println("Sending payload: " + payload);

  if (client.publish("v1/devices/me/telemetry", payload.c_str())) {
    Serial.println("Data sent to ThingsBoard");
  } else {
    Serial.println("Failed to send data");
  }
}
// === Connect to MQTT Server ===
void connectToMQTT() {
  while (!client.connected()) {
    updateConnectionLED();  // Update LED to show status
    Serial.println("Connecting to ThingsBoard...");
```

```
    if (client.connect("ESP32Client", access_token, NULL)) {
      Serial.println("Connected to ThingsBoard!");
    } else {
      Serial.print("MQTT connect failed, rc=");
      Serial.print(client.state());
      Serial.println(". Trying again in 2s...");
      delay(2000);
    }
  }
}

// === Setup Function ===
void setup() {
  Serial.begin(115200);

  // Configure pin modes
  pinMode(PIR_PIN, INPUT);
  pinMode(LIGHT_SENSOR_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);  // Start with relay OFF

  // Configure RGB LED pins
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  setRGB(0, 0, 0);  // Turn off LED initially
```

```
  // Connect to WiFi
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
    updateConnectionLED();  // Show RED while waiting
  }
  Serial.println("\nConnected to WiFi");

  // Connect to MQTT
  client.setServer(mqtt_server, mqtt_port);
  connectToMQTT();

  lastUpdateTime = millis();  // Initialize timing
}

// === Main Loop ===
void loop() {
  // Reconnect to MQTT if disconnected
  if (!client.connected()) {
    connectToMQTT();
  }
  client.loop();  // Maintain MQTT connection
```

```
updateConnectionLED();  // Update LED based on connection status

// Get current time and calculate elapsed time
unsigned long now = millis();
float timeDiffSec = (now - lastUpdateTime) / 1000.0;

// Read sensor values
bool pirState = digitalRead(PIR_PIN);
int lightIntensity = analogRead(LIGHT_SENSOR_PIN);
bool relayState = false;

// Relay logic: turn on only if motion detected and it's dark
if (lightIntensity < 100 && pirState == HIGH) {
  digitalWrite(RELAY_PIN, HIGH);
  relayState = true;
} else {
  digitalWrite(RELAY_PIN, LOW);
  relayState = false;
}

// Update ON/OFF time tracking
if (relayState) {
  onTimeSeconds += timeDiffSec;
} else {
  offTimeSeconds += timeDiffSec;
}
```

```
  lastUpdateTime = now;  // Update timestamp


  // Send current status to ThingsBoard
  sendRelayStatus(relayState, onTimeSeconds, offTimeSeconds);


  delay(1000);  // Wait 1 second before next loop (can be replaced with non-blocking logic)
}
```

2.Code for Root Rule Chain in ThingsBoard Cloud:

```
// Rated power of the connected device in kilowatts (e.g., 4000W = 4 kW)
var power_kw = 4;


// Total number of seconds in a day
var total_seconds_in_day = 86400;


// --- Parse Relay ON Time from telemetry ---
var relay_seconds = parseFloat(msg.on_time);
// Check for invalid or missing data
if (relay_seconds == null || relay_seconds != relay_seconds) {
   relay_seconds = 0;
}


// --- Parse Relay OFF Time from telemetry ---
var relayoff_seconds = parseFloat(msg.false_time);
if (relayoff_seconds == null || relayoff_seconds != relayoff_seconds) {
```

```
    relayoff_seconds = 0;
}

// --- Retrieve Previous Metadata Safely ---

// Total energy used so far (in kWh)
var prevEnergyUsed = parseFloat(metadata.prevEnergyUsed);
if (prevEnergyUsed == null || prevEnergyUsed != prevEnergyUsed) {
    prevEnergyUsed = 0.0;
}

// Total active (ON) time so far (in minutes)
var prevActiveTime = parseFloat(metadata.prevActiveTime);
if (prevActiveTime == null || prevActiveTime != prevActiveTime) {
    prevActiveTime = 0.0;
}

// Total inactive (OFF) time so far (in minutes)
var prevInactiveTime = parseFloat(metadata.prevInactiveTime);
if (prevInactiveTime == null || prevInactiveTime != prevInactiveTime) {
    prevInactiveTime = 0.0;
}

// Total energy saved so far (in kWh)
var prevEnergySaved = parseFloat(metadata.prevEnergySaved);
if (prevEnergySaved == null || prevEnergySaved != prevEnergySaved) {
```

```
    prevEnergySaved = 0.0;
}

// --- Energy and Time Calculations ---

// Energy consumed = (relay on time in hours) * power rating + previously stored usage
var energy_used = ((relay_seconds / 3600) * power_kw) + prevEnergyUsed;

// Full-day energy usage (if relay stayed ON all day)
var energy_full_day = (total_seconds_in_day / 3600) * power_kw;

// Energy saved = full-day usage - actual usage + previously saved value
var energy_saved = (power_kw * ((total_seconds_in_day - relay_seconds) / 3600)) + prevEnergySaved;

// Convert ON and OFF times to minutes and add to previous values
var active_time = (relay_seconds / 60) + prevActiveTime;
var inactive_time = (relayoff_seconds / 60) + prevInactiveTime;

// Calculate real-time energy cost and savings (assume ₹6.50/kWh or $6.50/kWh depending on context)
var real_cost = energy_used * 6.500;
var saved_cost = energy_saved * 6.500;

// --- Accumulate values only when relay has been ON for at least 1 second ---
if (relay_seconds > 1) {
```

```
    prevEnergyUsed += energy_used;
    prevActiveTime += active_time;
    prevEnergySaved += energy_saved;
}

// --- Daily reset after 1440 minutes (24 hours) ---
var totalTime = active_time + inactive_time;
if (totalTime == 1440) {
    prevEnergyUsed = 0.0;
    prevActiveTime = 0.0;
    prevEnergySaved = 0.0;
    prevInactiveTime = 0.0;
}

// --- Utility: Rounding function to keep results readable ---
function round(val, digits) {
    var factor = Math.pow(10, digits);
    return Math.round(val * factor) / factor;
}

// --- Prepare the output telemetry message ---
var result = {
    energy_used_kwh: round(energy_used, 3),
    energy_saved_kwh: round(energy_saved, 3),
    energy_full_day_kwh: round(energy_full_day, 3),
    active_time_min: round(active_time, 3),
```

```
    real_time_cost: round(real_cost, 3),
    real_time_saved_cost: round(saved_cost, 3),
};

// --- Update metadata for persistent storage ---
metadata.prevEnergyUsed = prevEnergyUsed.toString();
metadata.prevActiveTime = prevActiveTime.toString();
metadata.prevEnergySaved = prevEnergySaved.toString();
metadata.prevInactiveTime = prevInactiveTime.toString();

// --- Return the final processed message, metadata, and type ---
return {
    msg: result,
    metadata: metadata,
    msgType: msgType
};
```

Result & Discussions :

The IoT-based Smart Relay Control system was successfully implemented and tested using the ESP32 microcontroller. The results demonstrate that the system behaves as intended under various environmental conditions. The following outcomes were observed:

1. Relay Control Accuracy

- The relay was activated **only when motion was detected** and **ambient light levels were low**.
- During testing, the system consistently avoided false triggers in daylight and activated the relay promptly during darkness upon motion detection.
- The relay responded **within ~1 second**, offering near real-time control.

2. Sensor Integration

- The **PIR motion sensor** accurately detected human movement within its range (~6 meters).
- The **light sensor (LDR)** correctly measured ambient light and determined thresholds for distinguishing day from night.
- Light intensity values around **< 100** (on the ADC scale) were considered "dark" for relay activation.

3. Real-Time Telemetry

- Telemetry data was **successfully transmitted to ThingsBoard** every second via MQTT.
- The following values were updated and visible on the ThingsBoard dashboard:
    - relay_status: Boolean value (true/false).
    - on_time: Duration in seconds the relay was ON.
    - off_time: Duration in seconds the relay was OFF.
- All transmitted data was correctly parsed and visualized in ThingsBoard widgets such as charts and digital indicators.

4. RGB LED Status Feedback

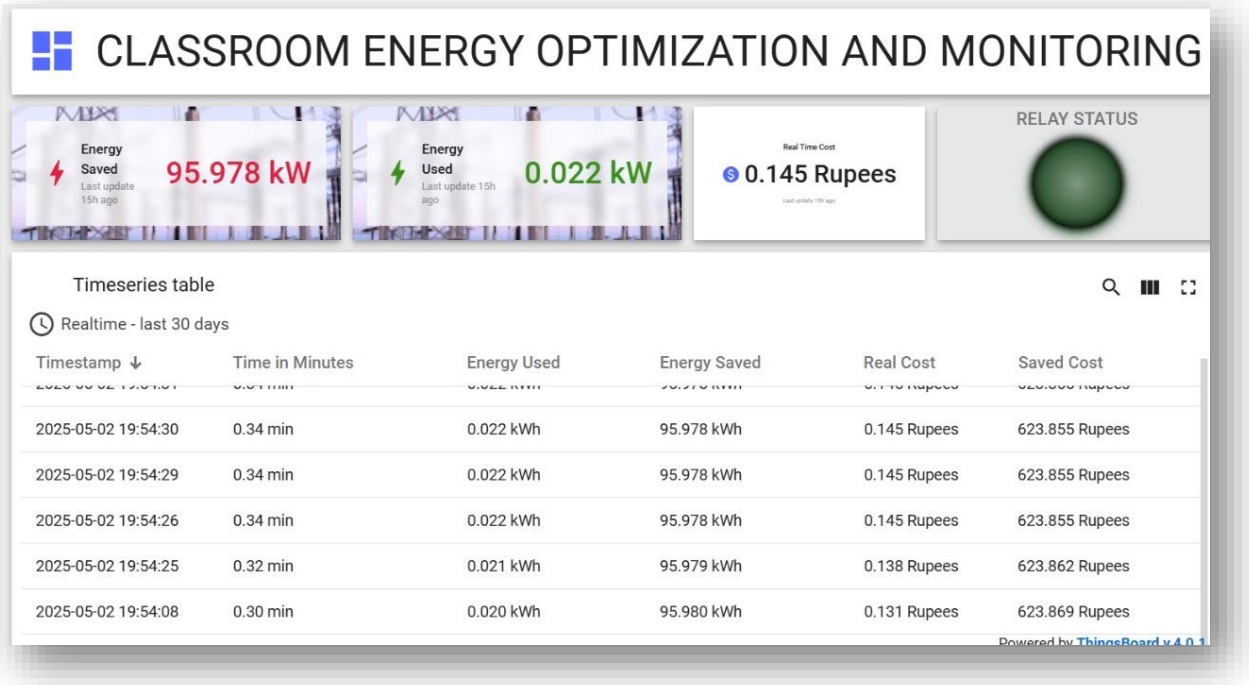
- The RGB LED accurately displayed the system's connectivity status:
  - **Red** when disconnected from Wi-Fi.
  - **Blue** when Wi-Fi was connected but MQTT was not.
  - **Green** when both Wi-Fi and MQTT were connected.
- This provided useful local visual feedback during deployment and debugging.

5. System Stability and Responsiveness

- The system ran **continuously and reliably** during long-duration tests.
- No significant delays, crashes, or missed MQTT transmissions were observed.
- Time tracking (relay ON/OFF duration) was accurate and reset-proof as long as the ESP32 remained powered.

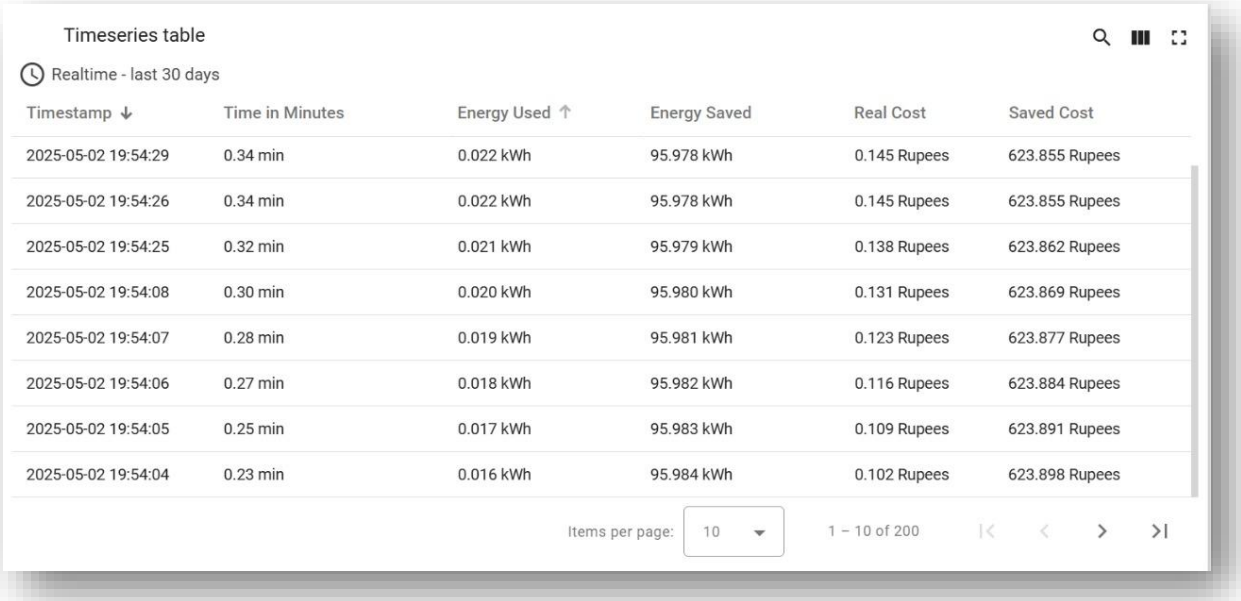
Picture no.1

Visualizing data of difference between before and after automation energy consumption



Picture no.2

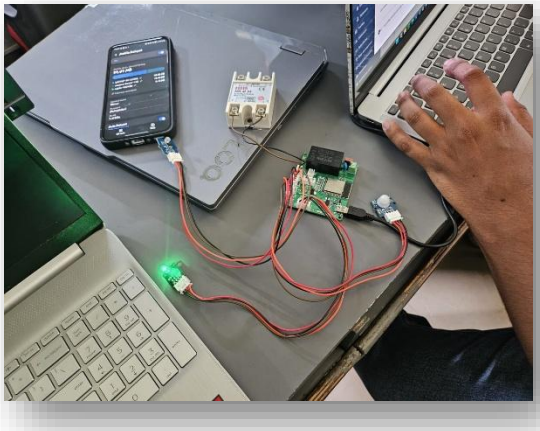
Visualizing data in ThingsBoard cloud dashboard.



Picture no.3

Visualizing data in numeric values using Timeseries Table.

Photographs of Project :



Picture no. 1

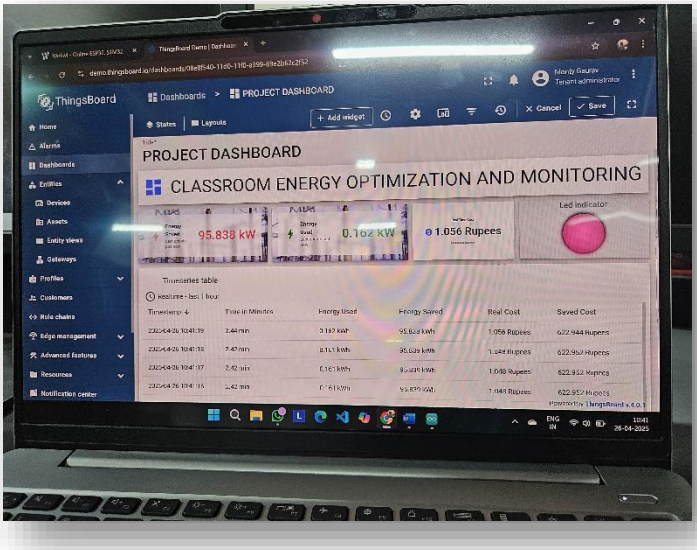
When system is completely online.

Picture no. 2

When system lost MQTT connection.

Picture no. 3

When system lose WiFi and gets offline.



Picture no. 4

Visualizing real time data on thingsBoard Dashboard while testing hardware.

Future Scope:

The Smart Indoor Energy Optimization System has wide potential for future development and scalability:

- **Integration with Smart HVAC Systems:**
  The system can be expanded to control fans and air conditioners based on occupancy and temperature data for greater energy savings.

- **Machine Learning for Predictive Automation:**
  Incorporating ML algorithms can help predict usage patterns and optimize energy use more intelligently over time.

- **Mobile App and Voice Assistant Integration:**
  Future versions can offer remote control and monitoring via mobile apps or smart assistants like Alexa or Google Assistant.

- **Scalability to Smart Campuses/Buildings:**
  The system can be scaled across large facilities, campuses, or office complexes for centralized energy management.

Conclusion:

The Smart Indoor Energy Optimization system effectively combines sensor technology, automation, and IoT to reduce unnecessary energy consumption in indoor environments such as university classrooms. By intelligently managing appliances based on real-time environmental data—like motion, light, and temperature—it ensures energy is used only when needed. The integration with ThingsBoard enables remote control and monitoring, supporting smart campus initiatives. This project highlights a scalable, cost-effective solution for promoting energy efficiency and sustainability in educational and commercial settings.

References:

1. **"Smart Energy Management for Smart Homes"**

Focus: IoT-based energy control systems using sensors and smart controllers.
Reference:
M. Fathi et al., *Smart Energy Management for Smart Homes*, IEEE Access, 2021,Page no.22-25.

2. **"Energy-Efficient Smart Lighting Systems Using IoT"**

Focus: Automatic lighting control with PIR and LDR integration.
Reference:
D. Georgievski et al., *Energy-efficient smart lighting systems using IoT*, Future Generation Computer Systems, 2017,Page no.18-20.

3. **"Internet of Things for Smart Energy Systems"**

Focus: Cloud integration and IoT frameworks for energy optimization.
Reference:
S. Madakam et al., *Internet of Things for Smart Energy Systems*, Procedia Computer Science, 2015,Page no.07-08.

4. **"A Smart Motion-Sensing Lighting System"**

Focus: Using PIR sensors to automatically control lighting to save energy.
Reference:
M. Elamari et al., *A Smart Motion-Sensing Lighting System for Energy Saving*, Journal of Building Engineering, 2020,Page no.09-10.

5. **"ThingsBoard: An Open-source IoT Platform for Data Collection and Visualization"**

Focus: Using ThingsBoard for device management and real-time energy monitoring.
Reference:
Official ThingsBoard Documentation and related case studies from IoT journals.