University of Moratuwa

Department of Electronics and Telecommunication

EN3251 – Internet Of Things

SMART Lighting System

Project Report

200356A, 200118X, 200179H

Problem Statement

Our project is driven by a multi-faceted goal: to design and implement smart lighting system that are not only beneficial for the elderly and individuals facing health challenges but also cater to the fast-paced lives of busy individuals. These devices offer the unique advantage of controlling lights from anywhere, removing the need for physical presence within the house and enhancing convenience for all.

The first key objective is to improve the quality of life for the elderly and those dealing with illness. These individuals often face mobility issues and daily challenges that can affect their independence. By providing remote control over core household lights, our smart lighting system offer them an opportunity to maintain their comfort and freedom, regardless of their location. This results in a more comfortable and manageable life, reducing the need for constant supervision and assistance.

Furthermore, our project recognizes that busy individuals also stand to benefit greatly from this technology. In today's fast-paced world, time is a precious commodity, and convenience is paramount. Our smart lighting system allow busy individuals to control their home environment while on the go, making it easier to switch on and off lights. The system will automatically switch on the bulbs when someone enters the room. Moreover, switches can be on and off by voice commands.

Solution and Design Details

Mobile Application: The focal point of our solution is an intuitively designed mobile application, acting as the primary control hub for users to manage and oversee their home environment remotely. The application offers a user-friendly interface, ensuring accessibility for diverse user groups, including the elderly, individuals with health concerns, and those leading busy lives. Users can remotely operate lights with this downloadable application, compatible with multiple devices. The application's flexibility allows customization to suit individual user preferences. Additionally, integration with the Node-Red platform expands the user interface options, enabling further customization and control.

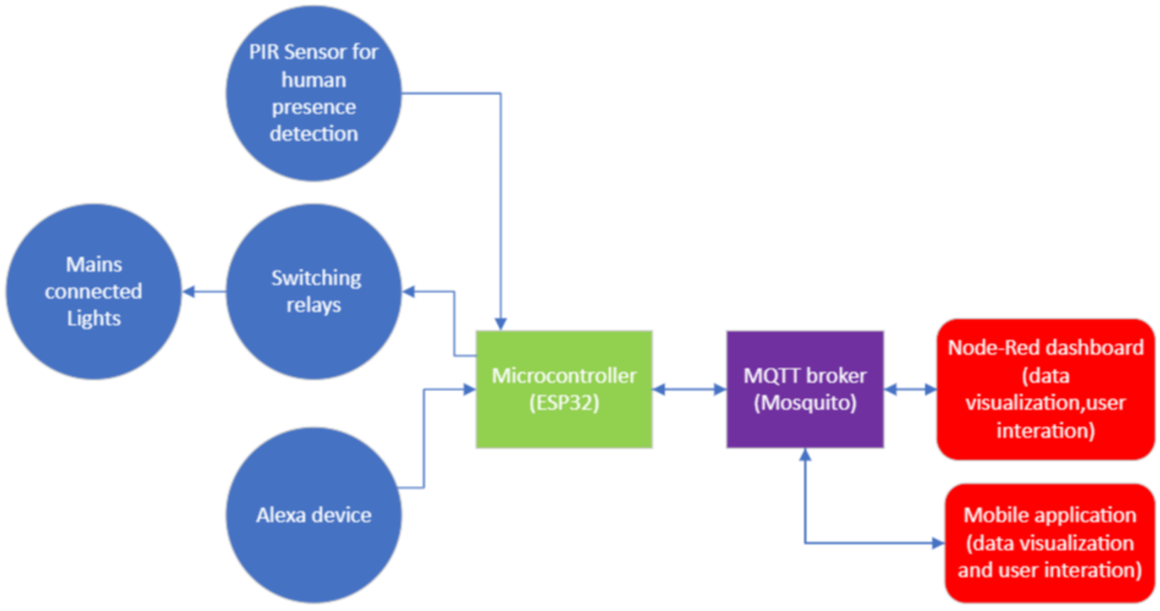
Scheduling: The smart lighting system is seamlessly integrated with Alexa, facilitating effortless pre-setting of lighting preferences. Users can customize lighting conditions for different periods throughout the day or night, enabling adjustments to ambiance, brightness, and color temperature according to individual preferences. This feature is particularly beneficial for individuals requiring a consistent and adaptable environment, such as the elderly or those with health challenges. Simultaneously, for busy individuals, the system simplifies daily routines by automating and adjusting lighting settings through voice commands to Alexa-enabled devices.

Sensing (Lighting Control, Alexa Integration): The smart lighting system utilizes sensors to detect ambient light levels and harmoniously integrates with Alexa-enabled devices. These sensors enable the system to regulate lighting based on real-time data and user preferences. Users have the convenience of adjusting brightness, and scheduling using voice commands through Alexa, delivering a personalized and convenient lighting experience.

Power Unit: This essential component requires power to drive the ESP32 board and other indicators. While it can be configured to function with battery power, owing to its direct connection to AC current, a rectification circuit is employed to obtain the necessary 5V voltage. The circuit reduces the initial 230V AC voltage to 10V and then utilizes a 7805-voltage regulator to stabilize the output at 5V, ensuring the proper functioning of the device.

Main Unit: This unit serves as the nerve center of the system, receiving and processing data transmitted from the mobile application via Wi-Fi. MQTT protocol is used for data transmission, with information published and subscribed to in JSON format. The ESP32, programmed in C++, manages the temperature sensor and various relays, including devices like the Adjustable Delay Timer Switch 5-30V MicroUSB Power. The relays facilitate the control of electrical equipment by using a small DC voltage, ensuring efficient and precise management.

Functional Block Diagram


PIR Sensor for human presence detection
Mains connected Lights
Switching relays
Microcontroller (ESP32)
MQTT broker (Mosquito)
Node-Red dashboard (data visualization,user interation)
Alexa device
Mobile application (data visualization and user interation)

ESP32 NodeMCU Dev Board is a versatile and compact microcontroller board that integrates the ESP32, a powerful Wi-Fi and Bluetooth module, with the NodeMCU development platform. It is widely used for IoT (Internet of Things) projects and offers a user-friendly programming environment.

Alexa Echo Dot ,a smart speaker developed by Amazon, equipped with Amazon's virtual assistant, Alexa. Its far-field voice recognition technology enables it to understand and respond to voice commands from across the room, making it convenient for our application.

A PIR (Passive Infrared) sensor is a motion-detection sensor used for human presence detection. It works by detecting changes in heat emissions in its surroundings. When a warm-blooded object, like a person, moves within its field of view, the PIR sensor detects the change in infrared radiation and triggers an output signal. This makes it a valuable component for applications like home security, lighting control, and energy-saving systems.

When you combine these components, you can create a system that uses the ESP32 NodeMCU Dev Board to interface with a PIR sensor for human presence detection. When the PIR sensor detects motion, it can send a signal to the ESP32, which can then turn the lights in each room ON and OFF accordingly. Additionally voice commands through Alexa echo dot device can also switch the lights.

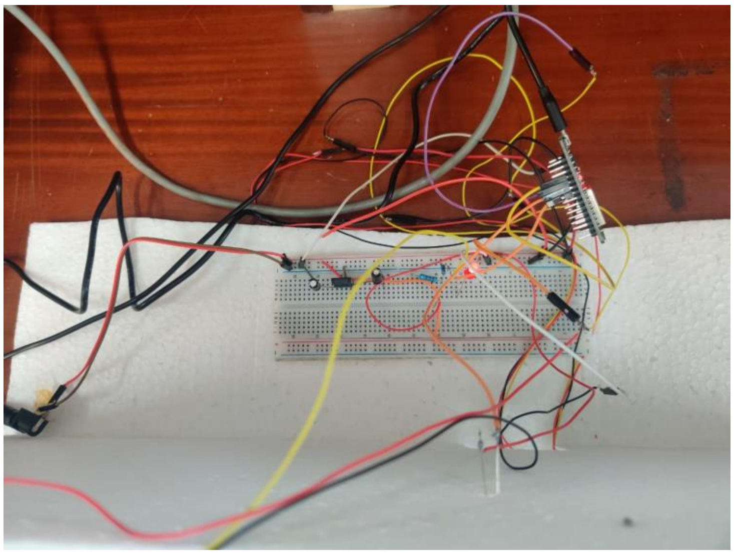
Hardware

Power circuit

We have used a 5V DC power adapter to get 5V to PIR sensor.

To get power for the ESP32 we reduced it to 3.3V using a voltage regulator circuit.(BA033T)

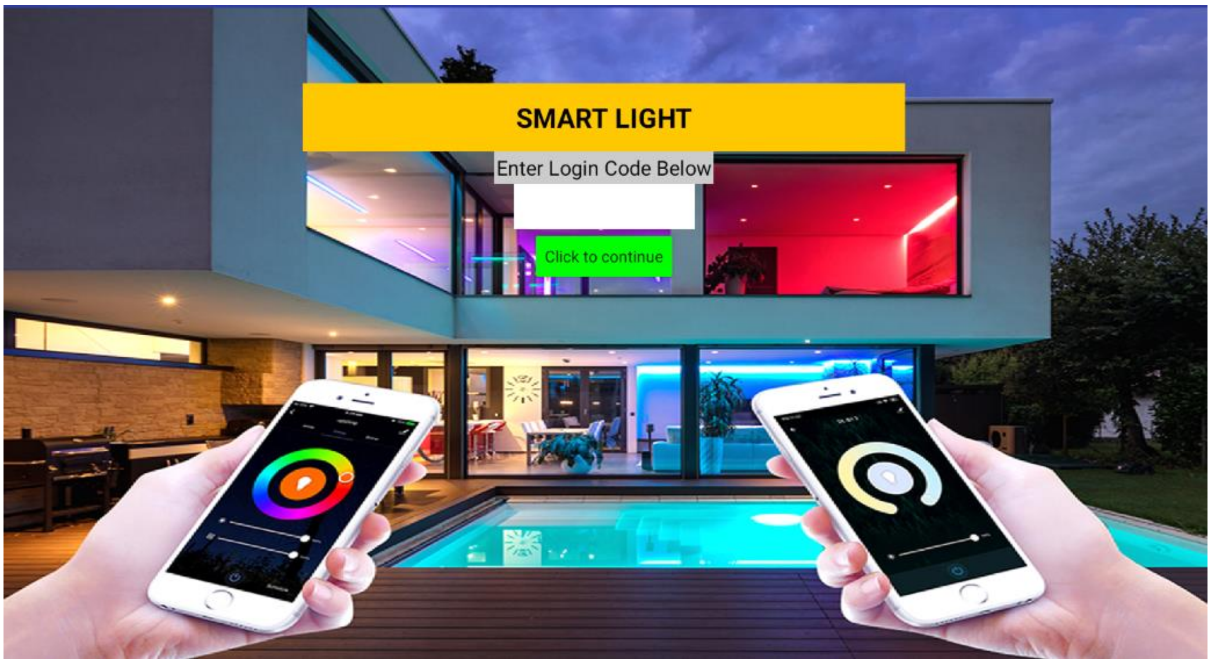
Additional LED to indicate power is on.



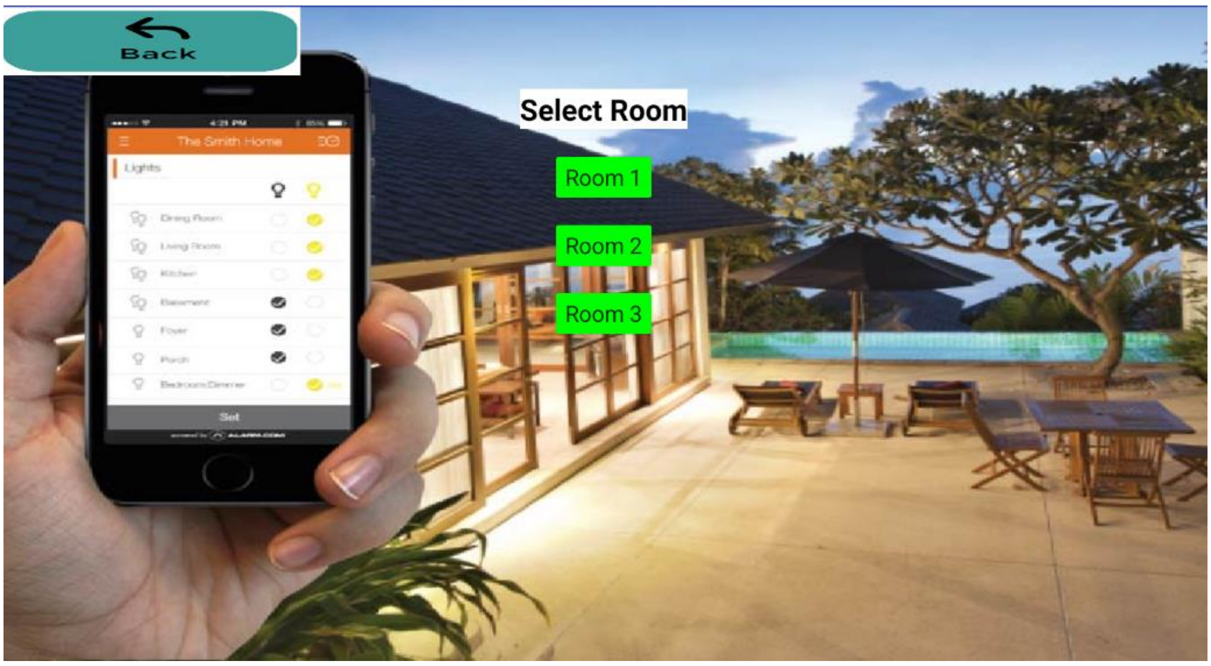
<u>Mobile application</u>

Front End and UI

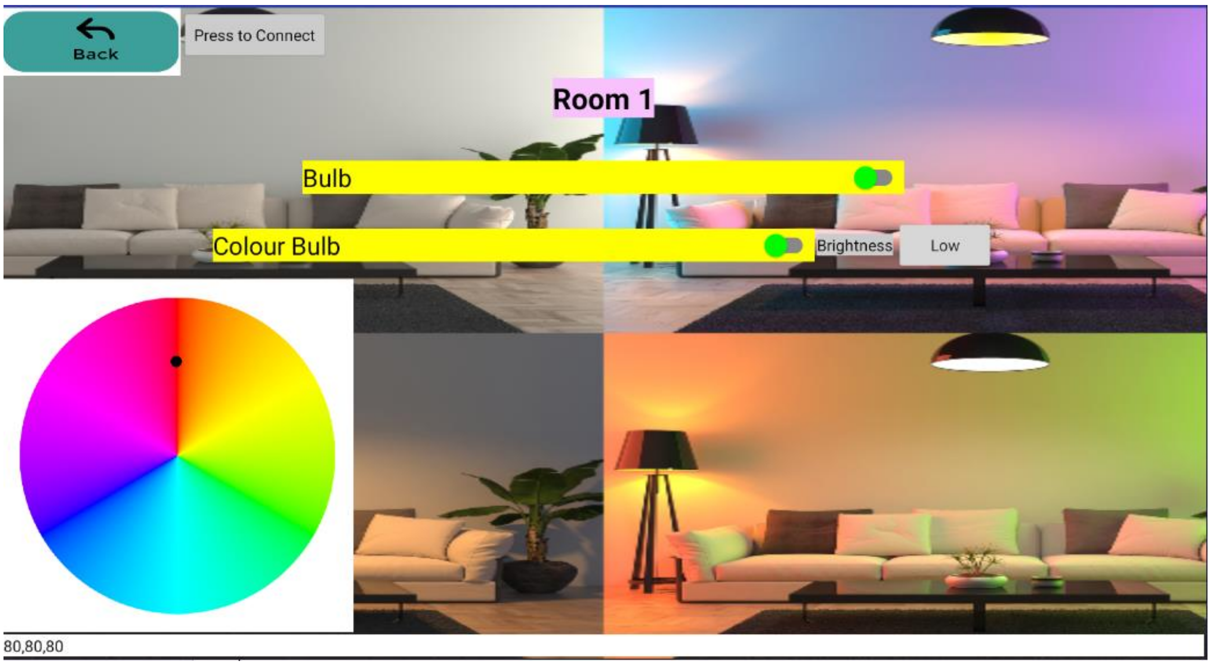
Each house will be provided with a unique identifier, users can use that as the login password. The same word will be used as the beginning part of the topic.



Then in the next screen users can click the room. App is made in more user friendly and simple manner because, this is made for both elder and younger people.



Once the room is selected user will be followed to a page as shown below.



Here user must first click the “press to connect” and check whether the device has the good connection with the server. If the connection is good it will show “Connected”, but if not connected it will show “Not connected”. Then user can switch on or off the normal bulb and colour bulb. In addition, user can select what ever the colour he like by dragging the black dot in the colour circle and choose the brightness level. There are 3 brightness levels called, 50, 150, 250.

Back end can be view with .aia file

.aia file link - https://dms.uom.lk/s/LWpR6ZZDk45fcot

.apk file link - https://dms.uom.lk/s/WqRBYDCs7NyXnyT

Further developments

This app will be further develop to show whether there is a person in the room or not. This is done by the PIR sensor in the room which will be use to light up the room when someone come inside.

Arduino code

PIR sensor was used to detect motion and turn on a bulb for a predefined amount of time.(5s in our case)

We used "FastLED" arduino library Daniel Garcia to program our RGB LED strips.

Alexa integration was done by emulating a Philips hue bulb on the esp32 and connecting it to an alexa echo dot device. We used "Fauxmoesp" library by Paul Vint to easily implement it. Alexa voice commands are capable of turning the bulb on/off, adjusting the brightness.

Mqtt messages are sent when the user sends a command to change the state(on/off), color and brightness. After the esp has done the adjustment to the light it publishes an mqtt message which is received by the dashboard and the dashboard sets its visual output accordingly.

We used "pubsubclient" library to create our mqtt client and "ArduinoJson" library for manipulating json objects.

MQTT message format

state: either "on" or "off"   color: RGB hexadecimal value   brightness: an integer from 0-255

Eg:

{"state": "on" , "color": "#3F5C76" , "brightness":125}

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <FastLED.h>
#include "fauxmoESP.h"
#include <Arduino.h>
#include <PubSubClient.h>
#include "Mqtt.h"
#include "Led.h"
#include "Wifi_credentials.h"
#include <ArduinoJson.h>

void Wifi_init();
void pinInit();
void callback(char* topic, byte* payload, unsigned int length);
CRGB light1[NUM_LEDS];
CRGB light2[NUM_LEDS];
static unsigned long last = millis();
int pirState=LOW;

WiFiClient espClient;
PubSubClient client(espClient);
fauxmoESP fauxmo;
```

```
void setup() {
  Serial.begin(115200);
  pinInit();
  Wifi_init();

  client.setServer(MQTTSERVER, MQTTPORT);
  client.setCallback(callback); // For Subscription

  while (!client.connected()) {
    Serial.println("Connecting to MQTT..");
    if (client.connect("ESP32SAHRKs")) {
      Serial.print("Connected to MQTT");
      client.subscribe(TOPIC1); // For Subscription
      //client.subscribe(TOPIC2);
    } else {
      Serial.println("MQTT Failed to connect");
      delay(5000);
    }
  }
  fauxmo.createServer(true); // not needed, this is the default value
  fauxmo.setPort(80); // This is required for gen3 devices
  fauxmo.enable(true);
  fauxmo.addDevice("light1");
  fauxmo.addDevice("light2");

  FastLED.addLeds<WS2812B, PIN_ROOM1, GRB>(light1, NUM_LEDS);
  FastLED.addLeds<WS2812B, PIN_ROOM2, GRB>(light2, NUM_LEDS);
  fauxmo.onSetState([](unsigned char device_id, const char * device_name, bool
state, unsigned char value) {

         Serial.printf("[MAIN] Device #%d (%s) state: %s value: %d\n",
device_id, device_name, state ? "ON" : "OFF", value);

        StaticJsonDocument<200> doc;
        const char* output;
        doc["color"]="#ff0000";
        if (strcmp(device_name, "light1")==0) {
          if (state){
            fill_solid(light1,NUM_LEDS, CRGB::Red);
            FastLED.setBrightness(value);
            doc["state"]="on";
            doc["brightness"]=value;}
          else{
            fill_solid(light1,NUM_LEDS, CRGB::Black);
            doc["state"]="off";}
          FastLED.show();
          //serializeJson(doc,output);
          //client.publish("Sharks/room1pub",output);
        }

        else if (strcmp(device_name, "room2")==0) {
          if (state){
            fill_solid(light2,NUM_LEDS, CRGB::Red);
            FastLED.setBrightness(value);
            doc["state"]="on";
            doc["brightness"]=value;
          }
          else{
            fill_solid(light2,NUM_LEDS, CRGB::Black);
            doc["state"]="off";}
          FastLED.show();
          //serializeJson(doc,output);
          //client.publish("Sharks/room2pub",output);
        }
    });
}

void loop() {
  client.loop();
  fauxmo.handle();


  int val = digitalRead(PIN_PIRSENSOR);
  if (val == HIGH) {
    digitalWrite(PIN_PIRLIGHT, HIGH);
    if (pirState == LOW) {
      Serial.println("Motion detected!");
      pirState = HIGH;
    }
  } else {
    if (millis() - last > 5000) {
        last = millis();
      if (pirState == HIGH){
          digitalWrite(PIN_PIRLIGHT, LOW);
          Serial.println("Motion ended!");
          pirState = LOW;
      }
    }
  }
}

void pinInit(){
  pinMode(PIN_ROOM1,OUTPUT);
  pinMode(PIN_ROOM2,OUTPUT);
  pinMode(PIN_PIRLIGHT,OUTPUT);
}
```

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.println(topic);

  char payloadStr[length + 1];
  memcpy(payloadStr, payload, length);
  payloadStr[length] = '\0';

  Serial.print("Payload: ");
  Serial.println(payloadStr);

  StaticJsonDocument<200> doc;
  DeserializationError error = deserializeJson(doc, payload);

  // Test if parsing succeeds
  if (error) {
    Serial.print(F("deserializeJson() failed: "));
    Serial.println(error.f_str());
    return;
  }

  int brightness=doc["brightness"];

  // Extract color string from JSON
  const char *colorHex = doc["color"];

  // Convert hexadecimal string to RGB values
  long colorValue = strtol(colorHex + 1, NULL, 16); // Skip the '#' character
  uint8_t red = (colorValue >> 16) & 0xFF;
  uint8_t green = (colorValue >> 8) & 0xFF;
  uint8_t blue = colorValue & 0xFF;

  if(strcmp(topic,"Sharks/room1sub")==0){
    if(strcmp(doc["state"],"off")==0) {
      fill_solid(light1,NUM_LEDS, CRGB::Black);
      FastLED.show();
      fauxmo.setState("light1",false,0);
      client.publish("Sharks/room1pub",payloadStr);}
    else if(strcmp(doc["state"],"on")==0){
      fill_solid(light1,NUM_LEDS, CRGB(red,green,blue));
      FastLED.setBrightness(brightness);
      FastLED.show();
      fauxmo.setState("light1",true,brightness);
      client.publish("Sharks/room1pub",payloadStr);}
  }

  if(strcmp(topic,"Sharks/room2sub")==0){
    if(strcmp(doc["state"],"off")==0) {
      fill_solid(light2,NUM_LEDS, CRGB::Black);
      FastLED.show();
      fauxmo.setState("light2",false,0);
      client.publish("Sharks/room2pub",payloadStr);}
    else if(strcmp(doc["state"],"on")==0){
      fill_solid(light2,NUM_LEDS, CRGB(red,green,blue));
      FastLED.setBrightness(brightness);
      FastLED.show();
      fauxmo.setState("light2",true,brightness);
      client.publish("Sharks/room1pub",payloadStr);}
  }

  if(strcmp(topic,"Sharks/room3sub")==0){
    if(strcmp(doc["state"],"off")==0) {
      digitalWrite(PIN_PIRLIGHT,LOW);
      fauxmo.setState("room1",false,0);
      client.publish("Sharks/room3pub",payloadStr);}
    else if(strcmp(doc["state"],"on")==0){
      digitalWrite(PIN_PIRLIGHT,HIGH);
      fauxmo.setState("room1",true,brightness);
      client.publish("Sharks/room3pub",payloadStr);}
  }
}
```

mqtt.h file

```
//json format  {state:on/off , color:0x232325 , brightness: 0-255}
#define MQTTSERVER  "test.mosquitto.org"
#define MQTTPORT  1883
#define TOPIC1   "Sharks/#"
```

Led.h file

```
#define NUM_LEDS 60
#define PIN_ROOM1 2
#define PIN_ROOM2 4
#define PIN_PIRLIGHT 16
#define PIN_PIRSENSOR 17
```

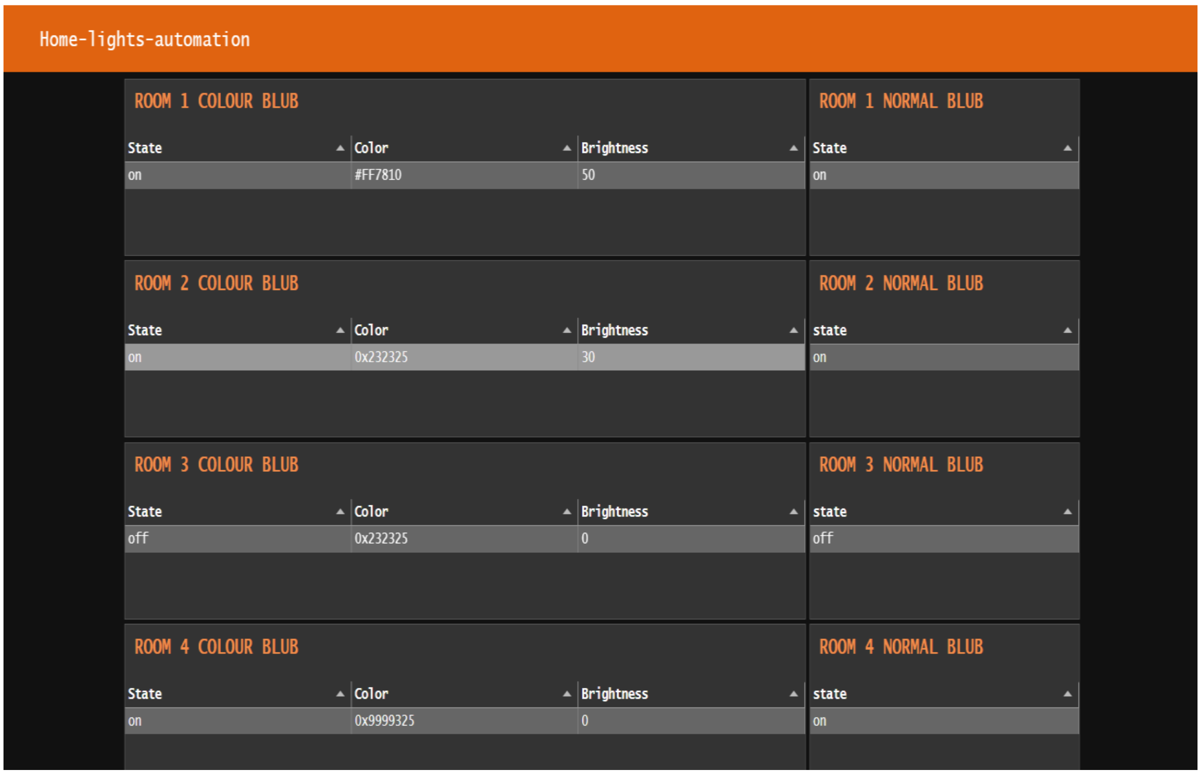
Wifi_credentials.h file

```
#define SSID "OnePlus 7T"
#define PASSWORD "qwertasdf"

void Wifi_init(){
  WiFi.begin(SSID, PASSWORD);  //Initialize WiFi Object
  Serial.print("Connecting to ");
  Serial.println(SSID);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("Connected to WiFi");
  Serial.print("IP Address : ");
  Serial.print(WiFi.localIP());
}
```

Link for the code - https://github.com/DusaraG/Home-lights-automation-IOT-Project.git

Node-red Dashboard

We have successfully implemented our Node-RED home lighting automation system, and as a result, we have constructed an easy-to-use dashboard that allows us to monitor and manage many rooms in our house. Both regular and colour bulbs are supported by the dashboard, which offers features like brightness adjustment, on/off switches, and dynamic colour visualization for colour bulbs, and on/off toggles for standard bulbs. Users may easily adjust their lighting settings in various rooms thanks to this user interface, which facilitates smooth interaction. The technology improves user experience and energy efficiency by providing a practical and effective home lighting solution.



Home-lights-automation

ROOM 1 COLOUR BLUB

State	Color	Brightness
on	#FF7810	50

ROOM 1 NORMAL BLUB

State
on

ROOM 2 COLOUR BLUB

State	Color	Brightness
on	0x232325	30

ROOM 2 NORMAL BLUB

state
on

ROOM 3 COLOUR BLUB

State	Color	Brightness
off	0x232325	0

ROOM 3 NORMAL BLUB

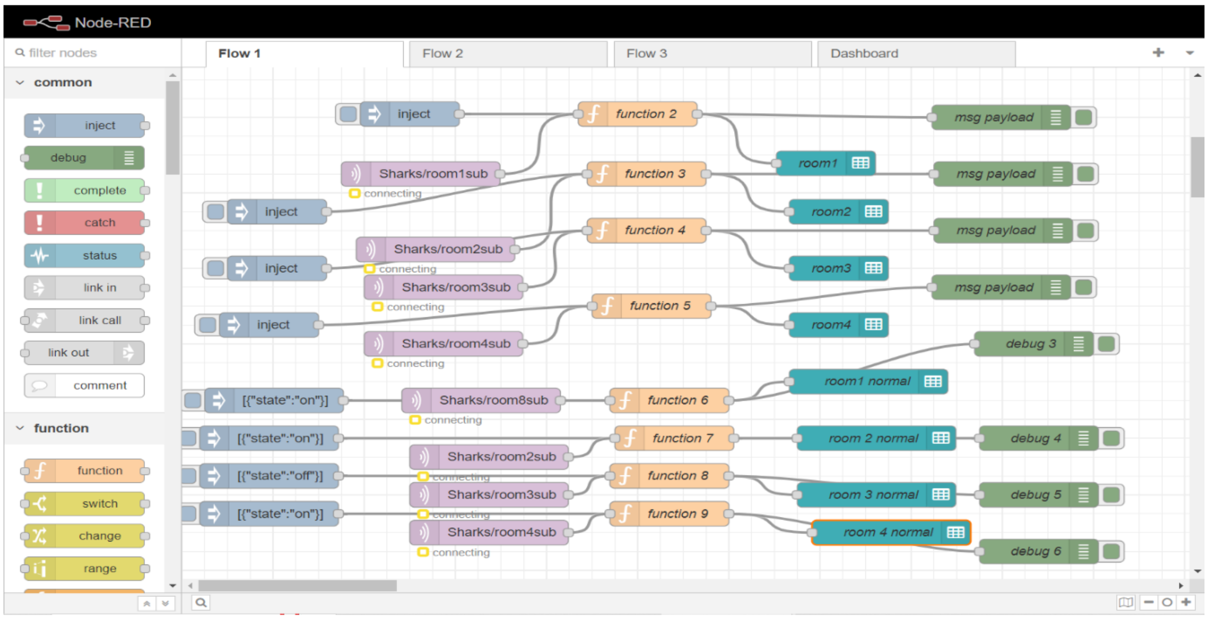
state
off

ROOM 4 COLOUR BLUB

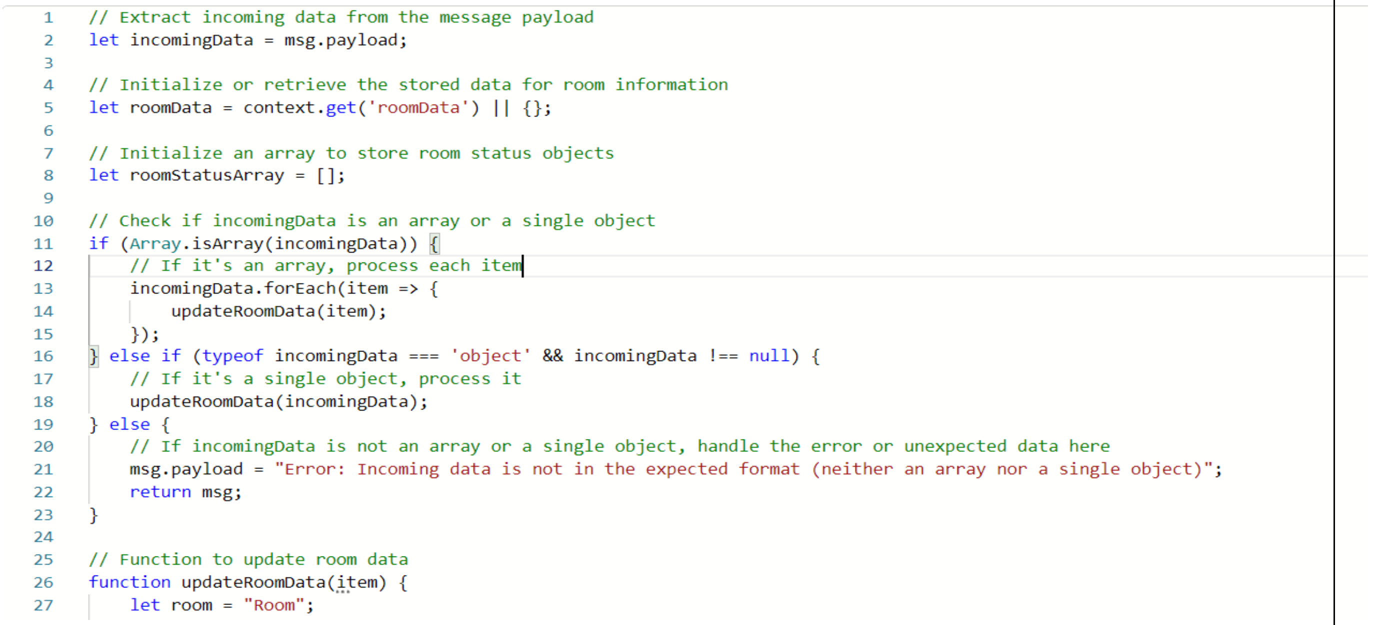
State	Color	Brightness
on	0x9999325	0

ROOM 4 NORMAL BLUB

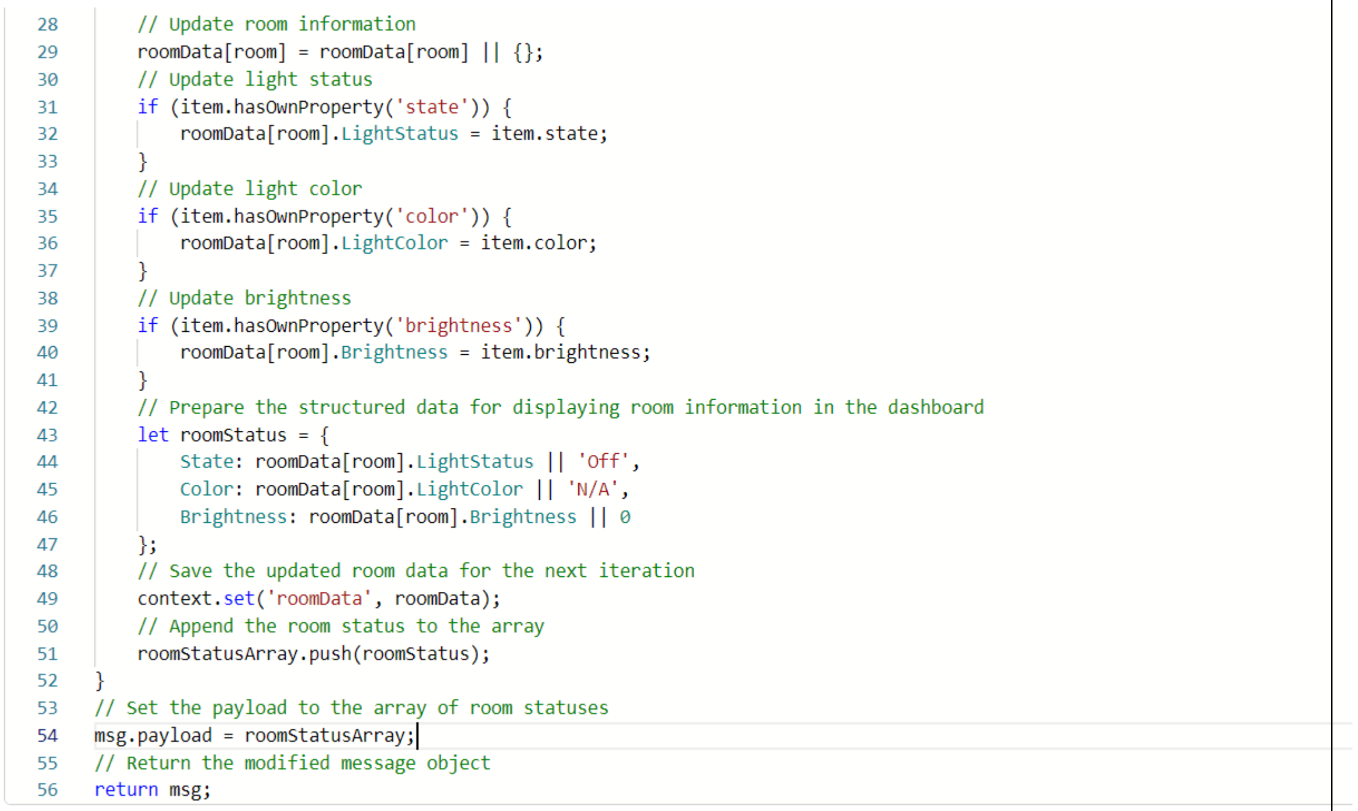
state
on



Colours blub function code.

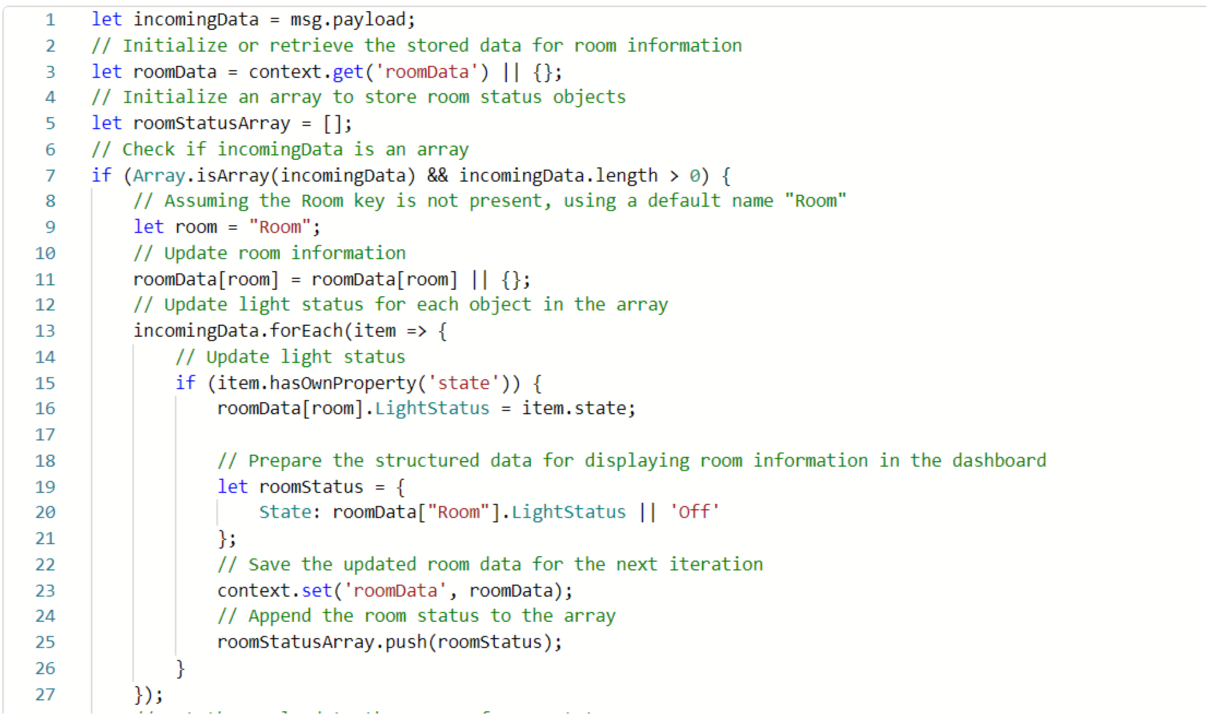


```javascript
// Extract incoming data from the message payload
let incomingData = msg.payload;

// Initialize or retrieve the stored data for room information
let roomData = context.get('roomData') || {};

// Initialize an array to store room status objects
let roomStatusArray = [];

// Check if incomingData is an array or a single object
if (Array.isArray(incomingData)) {
    // If it's an array, process each item
    incomingData.forEach(item => {
        updateRoomData(item);
    });
} else if (typeof incomingData === 'object' && incomingData !== null) {
    // If it's a single object, process it
    updateRoomData(incomingData);
} else {
    // If incomingData is not an array or a single object, handle the error or unexpected data here
    msg.payload = "Error: Incoming data is not in the expected format (neither an array nor a single object)";
    return msg;
}

// Function to update room data
function updateRoomData(item) {
    let room = "Room";
```

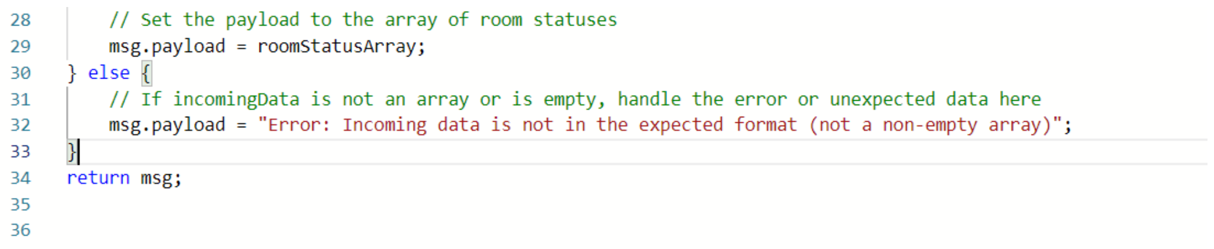


```javascript
    // Update room information
    roomData[room] = roomData[room] || {};
    // Update light status
    if (item.hasOwnProperty('state')) {
        roomData[room].LightStatus = item.state;
    }
    // Update light color
    if (item.hasOwnProperty('color')) {
        roomData[room].LightColor = item.color;
    }
    // Update brightness
    if (item.hasOwnProperty('brightness')) {
        roomData[room].Brightness = item.brightness;
    }
    // Prepare the structured data for displaying room information in the dashboard
    let roomStatus = {
        State: roomData[room].LightStatus || 'Off',
        Color: roomData[room].LightColor || 'N/A',
        Brightness: roomData[room].Brightness || 0
    };
    // Save the updated room data for the next iteration
    context.set('roomData', roomData);
    // Append the room status to the array
    roomStatusArray.push(roomStatus);
}
// Set the payload to the array of room statuses
msg.payload = roomStatusArray;
// Return the modified message object
return msg;
```

Normal Blub function code.



```
let incomingData = msg.payload;
// Initialize or retrieve the stored data for room information
let roomData = context.get('roomData') || {};
// Initialize an array to store room status objects
let roomStatusArray = [];
// Check if incomingData is an array
if (Array.isArray(incomingData) && incomingData.length > 0) {
    // Assuming the Room key is not present, using a default name "Room"
    let room = "Room";
    // Update room information
    roomData[room] = roomData[room] || {};
    // Update light status for each object in the array
    incomingData.forEach(item => {
        // Update light status
        if (item.hasOwnProperty('state')) {
            roomData[room].LightStatus = item.state;

            // Prepare the structured data for displaying room information in the dashboard
            let roomStatus = {
                State: roomData["Room"].LightStatus || 'Off'
            };
            // Save the updated room data for the next iteration
            context.set('roomData', roomData);
            // Append the room status to the array
            roomStatusArray.push(roomStatus);
        }
    });
```



```
    // Set the payload to the array of room statuses
    msg.payload = roomStatusArray;
} else {
    // If incomingData is not an array or is empty, handle the error or unexpected data here
    msg.payload = "Error: Incoming data is not in the expected format (not a non-empty array)";
}
return msg;
```

Further Improvements

Integrating security of home also to the same SMART home platform. Then all the CCTV cameras and fire, smoke sensors will be connected to the application such that user will be able to see them all in the mobile application.