RV Educational Institutions ®

RV College of Engineering ®

Autonomous Institution
Affiliated to Visvesvaraya
Technological University,
Belagavi

Approved by AICTE,
New Delhi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS - CS235AI

REPORT

Title- Home Automation Using RTOS.

Submitted by

<Rishabh Kumar Lal>	**<1RV22CS159>**
<Risheek S>	**<1RV22CS160>**
<Sankalpa BR>	**<1RV22CS178>**

Computer Science and Engineering

2023-2024

INDEX

Introduction

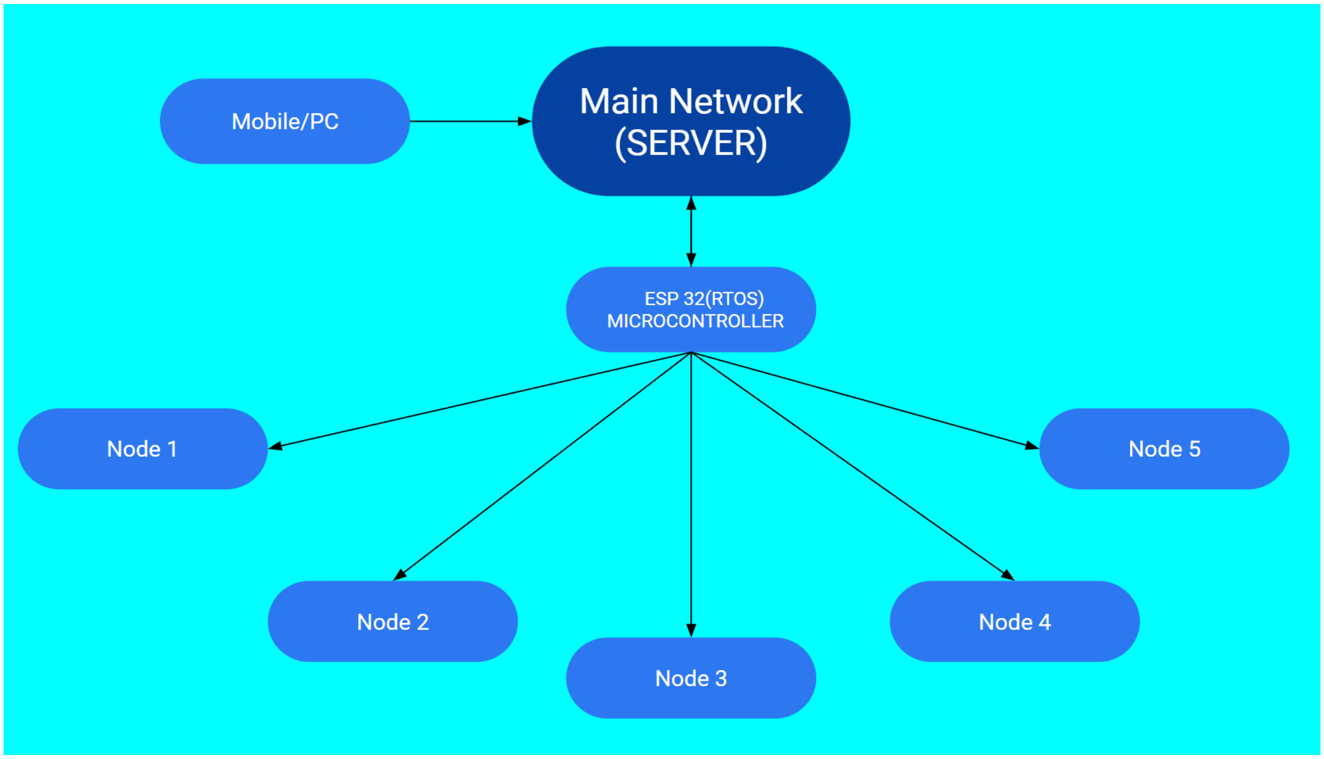
The increasing demand for smart home solutions has catalyzed the evolution of home automation systems, offering users unparalleled convenience, efficiency, and control over household devices. As homes become more interconnected and technologically advanced, there arises a need for robust frameworks to efficiently manage the myriad tasks, resources, and communication channels inherent in such systems. Real-Time Operating Systems (RTOS) emerge as a promising solution in this landscape, offering precise control over timing and prioritization to ensure seamless operation of critical functions within the home automation ecosystem.

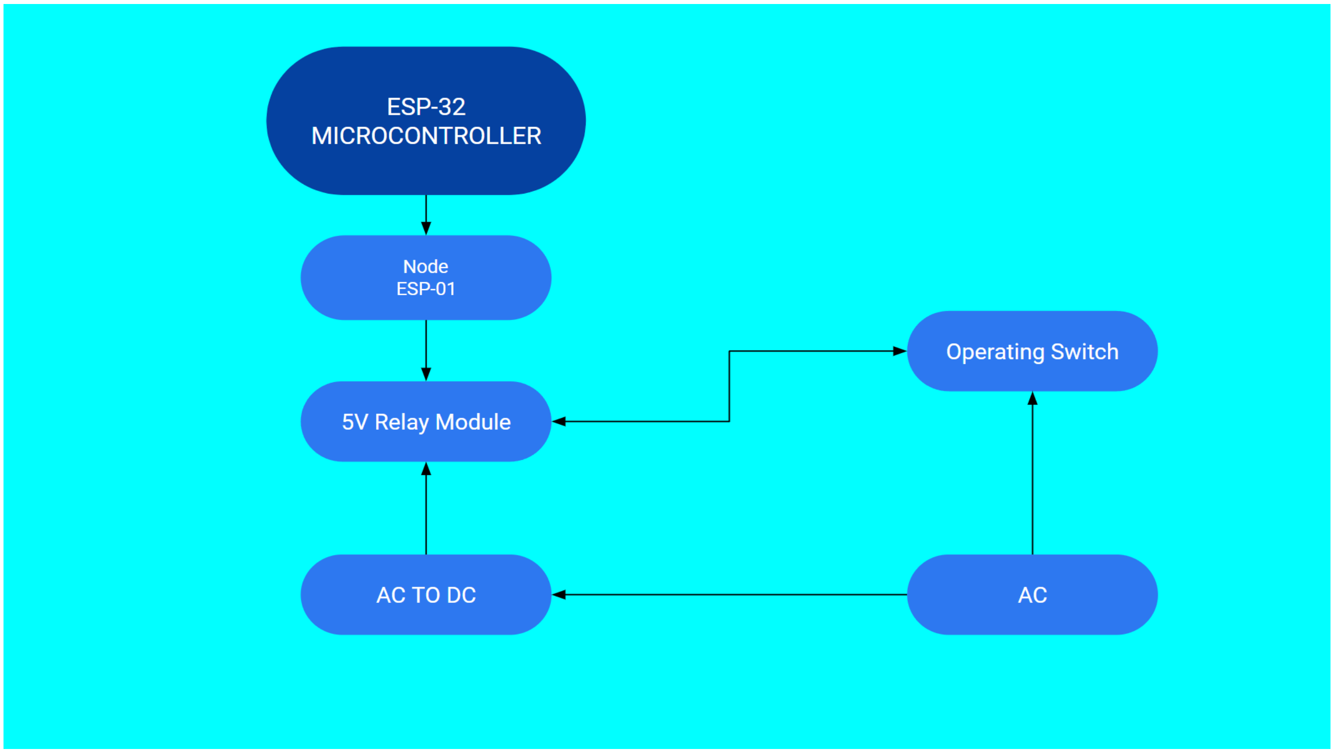
This report delves into the intricacies of designing and implementing a home automation system leveraging the capabilities of an RTOS, with a specific focus on FreeRTOS. By harnessing the power of real-time scheduling, resource management, and inter-task communication, the system orchestrates a harmonious interaction between nodes distributed throughout the home and a central server. The integration of IoT devices and smart home appliances further enriches the user experience, allowing for comprehensive control and monitoring from a centralized interface.

Emphasizing the practical application of OS concepts, this project endeavors to overcome the inherent challenges posed by embedded systems and IoT environments. Through meticulous design and implementation, the system aims to offer scalability and modularity, accommodating the addition of new nodes, sensors, and devices as the requirements evolve. Ultimately, the goal is to provide users with an intuitive and seamless home automation experience that enhances comfort, efficiency, and overall quality of life.

System architecture

pg 2





Methodology

- Task Design: The methodology begins with the meticulous design of tasks, each tailored to control specific home automation functions such as lighting, temperature regulation, security monitoring, and appliance management. These tasks are conceptualized based on the system requirements and user preferences, with careful consideration given to their timing constraints and criticality.
- RTOS Integration: Once the tasks are delineated, they are seamlessly integrated into the FreeRTOS framework. This involves leveraging the real-time scheduling capabilities of FreeRTOS to ensure timely execution of critical operations while effectively managing system resources. Tasks are assigned priority levels based on their importance and urgency, with higher-priority tasks preempting lower-priority ones as needed.
- Inter-Task Communication: Inter-task communication mechanisms are paramount for facilitating seamless data exchange between nodes and the central server. To achieve this, protocols such as the Message Queuing Telemetry Transport (MQTT) are employed, enabling efficient communication and coordination of home automation functions across the network. Messages containing control commands and sensor data are exchanged between tasks to synchronize their actions and maintain system coherence.
- Device Integration: The integration of IoT devices and smart home

appliances is a crucial aspect of the methodology. Each device is interfaced with the microcontrollers using Arduino IDE APIs, allowing for seamless interaction with the RTOS-based home automation system. Sensors provide real-time data on environmental conditions, while actuators enable the execution of control commands, ensuring dynamic responsiveness and adaptability to changing circumstances.

- Scalability and Modularity: The methodology is designed with scalability and modularity in mind, allowing for the seamless addition of new nodes, sensors, and devices as the home automation requirements evolve. This is achieved through a modular architecture that accommodates the integration of new components without disrupting the existing system functionality. By maintaining a flexible and extensible design, the system can easily adapt to the changing needs and preferences of users over time.

Overall, the methodology emphasizes a systematic approach to designing and implementing a home automation system using an RTOS, with a focus on achieving real-time responsiveness, efficient resource management, and seamless communication between nodes and the central server. Through careful planning and execution, the system aims to provide users with a robust and intuitive home automation experience that enhances comfort, convenience, and quality of life.

System Calls Used

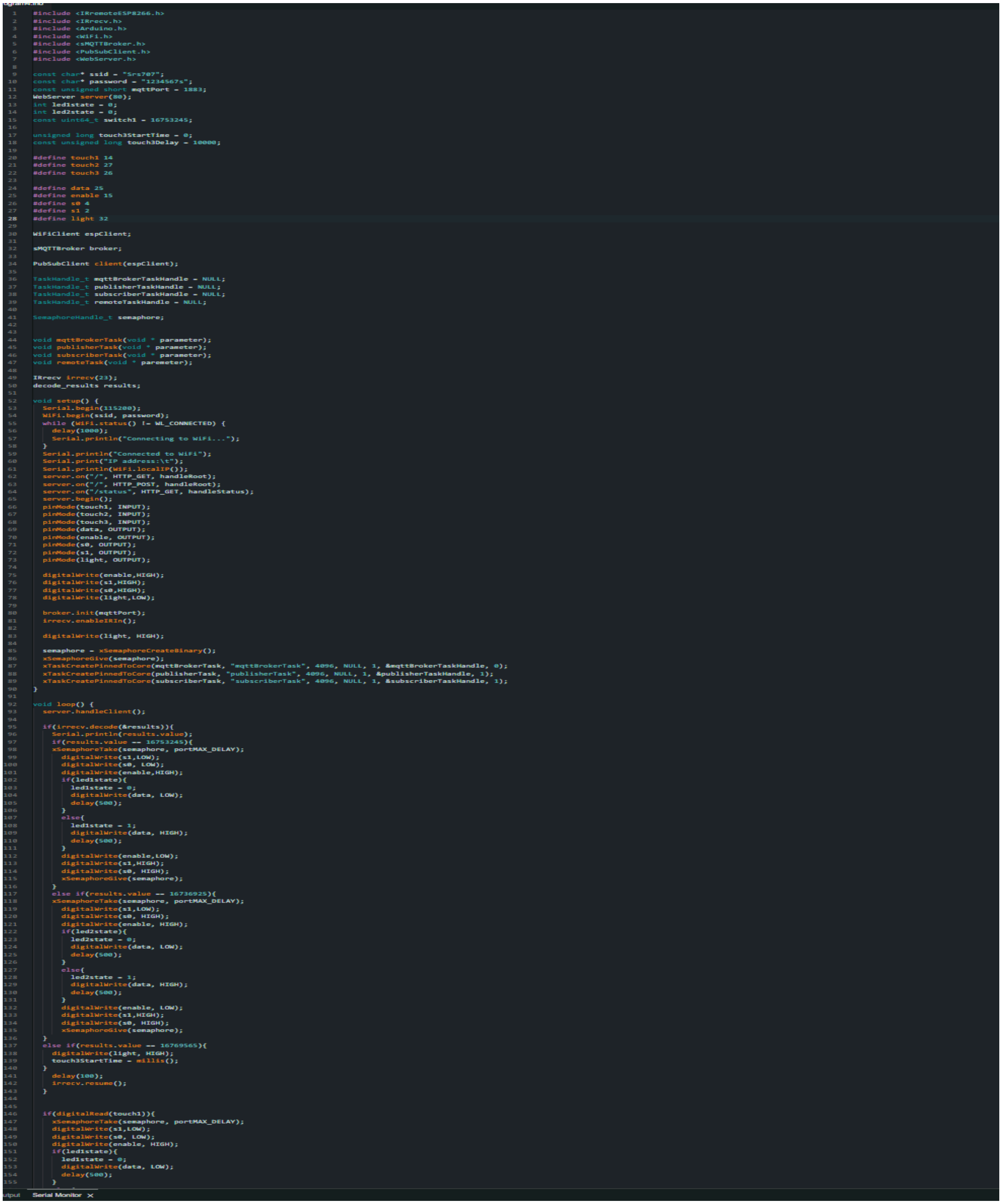
FreeRTOS: A real-time operating system kernel for microcontrollers, providing task scheduling, inter-task communication, and synchronization primitives, enabling developers to create responsive and efficient embedded applications.

Arduino IDE: An integrated development environment tailored for Arduino-compatible boards, simplifying the process of writing, compiling, and uploading code to microcontroller-based projects, fostering rapid prototyping and development.

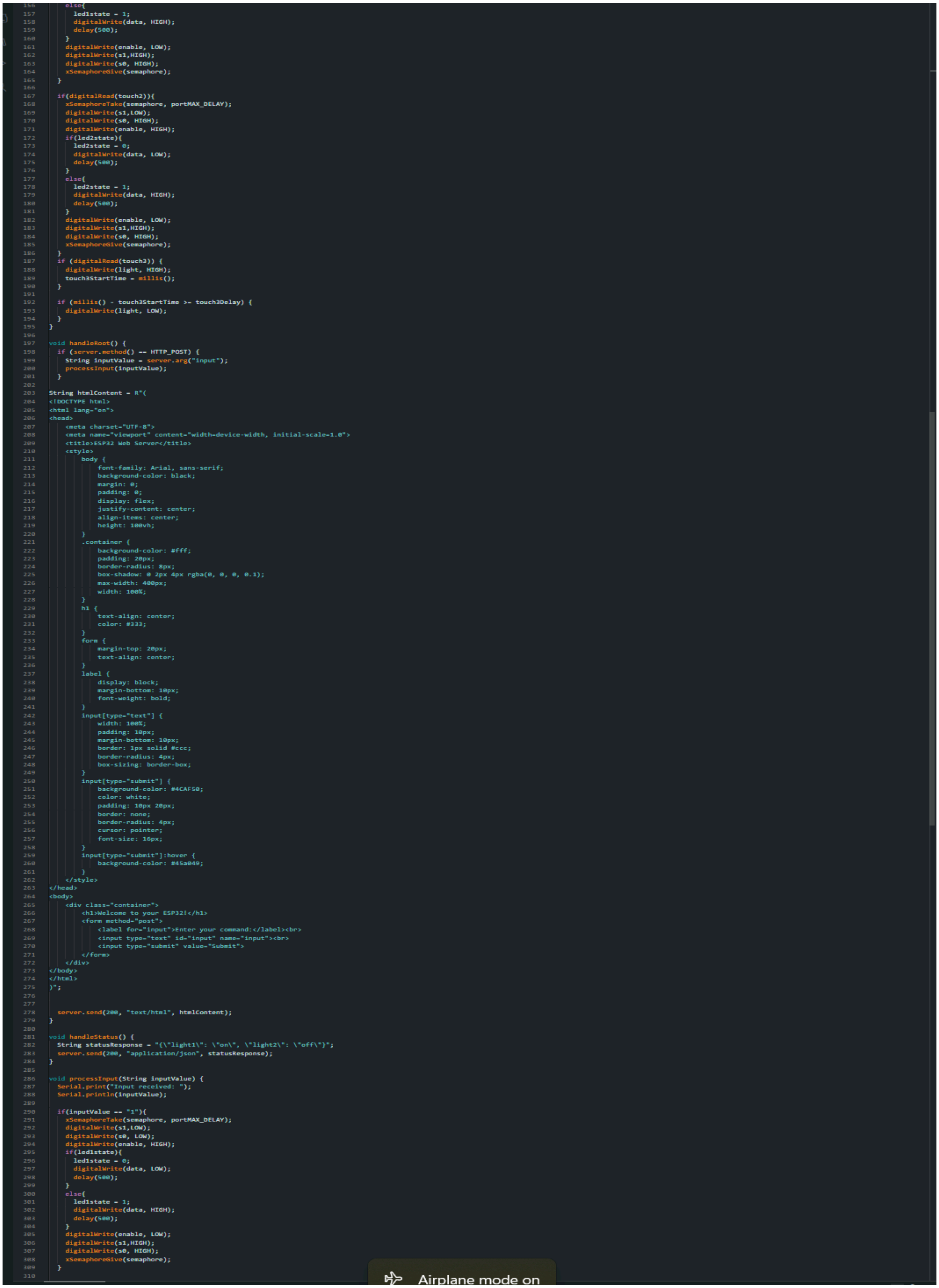
MQTT Protocol: A lightweight publish-subscribe messaging protocol widely used in IoT applications, facilitating communication between devices and servers with minimal overhead, ensuring efficient data exchange and scalable network architectures.

ESP32: A versatile microcontroller chip offering Wi-Fi and Bluetooth connectivity, alongside a rich set of peripherals and processing power, making it suitable for a wide range of IoT and embedded projects.

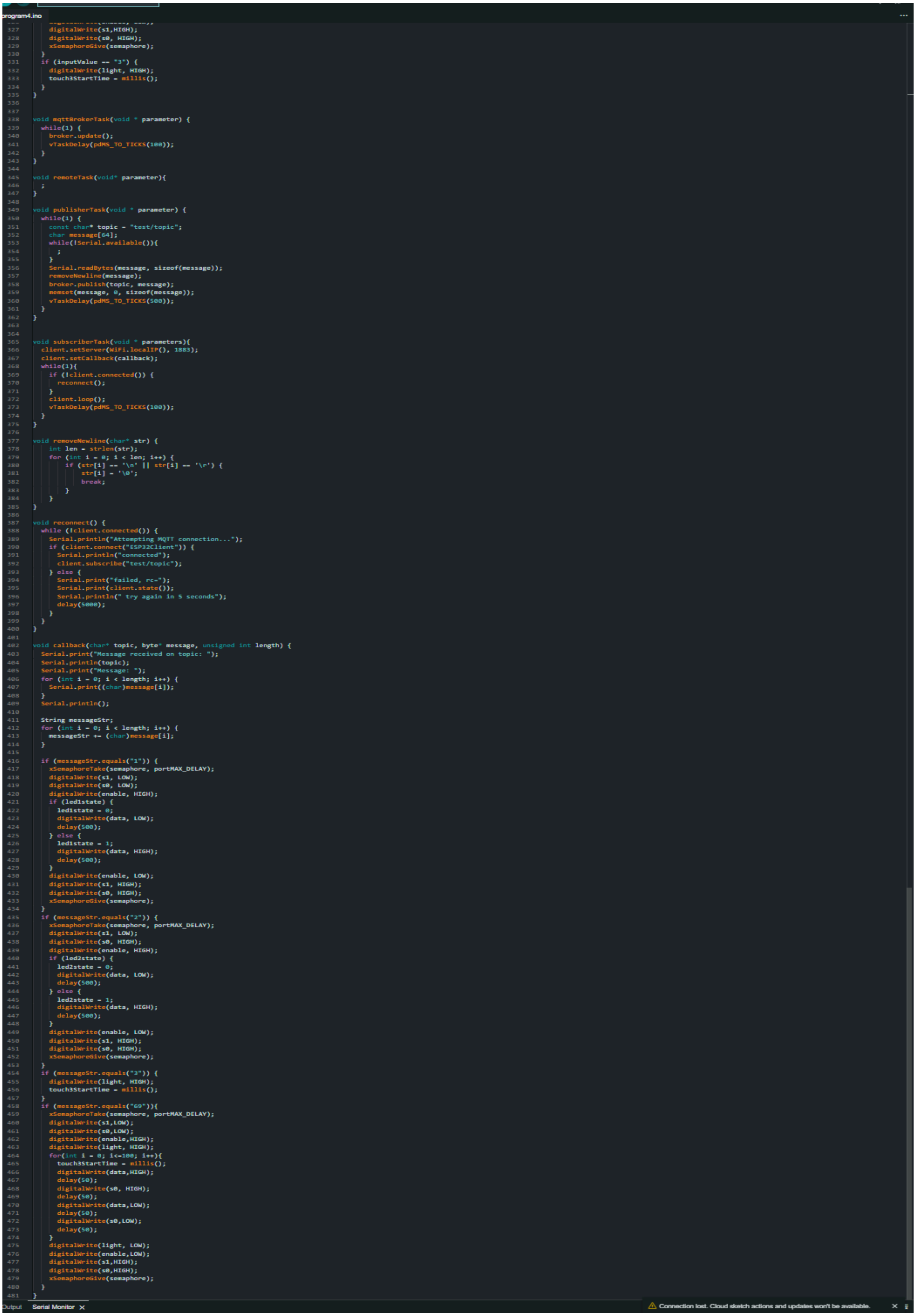
Source code



```
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <Arduino.h>
#include <WiFi.h>
#include <sMQTTBroker.h>
#include <PubSubClient.h>
#include <WebServer.h>

const char* ssid = "Srs707";
const char* password = "1234567s";
const unsigned short mqttPort = 1883;
WebServer server(80);
int led1state = 0;
int led2state = 0;
const uint64_t switch1 = 16753245;

unsigned long touch3StartTime = 0;
const unsigned long touch3Delay = 10000;

#define touch1 14
#define touch2 27
#define touch3 26

#define data 25
#define enable 15
#define s0 4
#define s1 2
#define light 32

WiFiClient espClient;

sMQTTBroker broker;

PubSubClient client(espClient);

TaskHandle_t mqttBrokerTaskHandle = NULL;
TaskHandle_t publisherTaskHandle = NULL;
TaskHandle_t subscriberTaskHandle = NULL;
TaskHandle_t remoteTaskHandle = NULL;

SemaphoreHandle_t semaphore;


void mqttBrokerTask(void * parameter);
void publisherTask(void * parameter);
void subscriberTask(void * parameter);
void remoteTask(void * paremeter);

IRrecv irrecv(23);
decode_results results;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());
  server.on("/", HTTP_GET, handleRoot);
  server.on("/", HTTP_POST, handleRoot);
  server.on("/status", HTTP_GET, handleStatus);
  server.begin();
  pinMode(touch1, INPUT);
  pinMode(touch2, INPUT);
  pinMode(touch3, INPUT);
  pinMode(data, OUTPUT);
  pinMode(enable, OUTPUT);
  pinMode(s0, OUTPUT);
  pinMode(s1, OUTPUT);
  pinMode(light, OUTPUT);

  digitalWrite(enable,HIGH);
  digitalWrite(s1,HIGH);
  digitalWrite(s0,HIGH);
  digitalWrite(light,LOW);

  broker.init(mqttPort);
  irrecv.enableIRIn();

  digitalWrite(light, HIGH);

  semaphore = xSemaphoreCreateBinary();
  xSemaphoreGive(semaphore);
  xTaskCreatePinnedToCore(mqttBrokerTask, "mqttBrokerTask", 4096, NULL, 1, &mqttBrokerTaskHandle, 0);
  xTaskCreatePinnedToCore(publisherTask, "publisherTask", 4096, NULL, 1, &publisherTaskHandle, 1);
  xTaskCreatePinnedToCore(subscriberTask, "subscriberTask", 4096, NULL, 1, &subscriberTaskHandle, 1);
}

void loop() {
  server.handleClient();

  if(irrecv.decode(&results)){
    Serial.println(results.value);
    if(results.value == 16753245){
    xSemaphoreTake(semaphore, portMAX_DELAY);
      digitalWrite(s1,LOW);
      digitalWrite(s0, LOW);
      digitalWrite(enable,HIGH);
      if(led1state){
        led1state = 0;
        digitalWrite(data, LOW);
        delay(500);
      }
      else{
        led1state = 1;
        digitalWrite(data, HIGH);
        delay(500);
      }
      digitalWrite(enable,LOW);
      digitalWrite(s1,HIGH);
      digitalWrite(s0, HIGH);
      xSemaphoreGive(semaphore);
    }
    else if(results.value == 16736925){
    xSemaphoreTake(semaphore, portMAX_DELAY);
      digitalWrite(s1,LOW);
      digitalWrite(s0, HIGH);
      digitalWrite(enable, HIGH);
      if(led2state){
        led2state = 0;
        digitalWrite(data, LOW);
        delay(500);
      }
      else{
        led2state = 1;
        digitalWrite(data, HIGH);
        delay(500);
      }
      digitalWrite(enable, LOW);
      digitalWrite(s1,HIGH);
      digitalWrite(s0, HIGH);
      xSemaphoreGive(semaphore);
  }
  else if(results.value == 16769565){
    digitalWrite(light, HIGH);
    touch3StartTime = millis();
  }
    delay(100);
    irrecv.resume();
  }


  if(digitalRead(touch1)){
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1,LOW);
    digitalWrite(s0, LOW);
    digitalWrite(enable, HIGH);
    if(led1state){
      led1state = 0;
      digitalWrite(data, LOW);
      delay(500);
    }
```



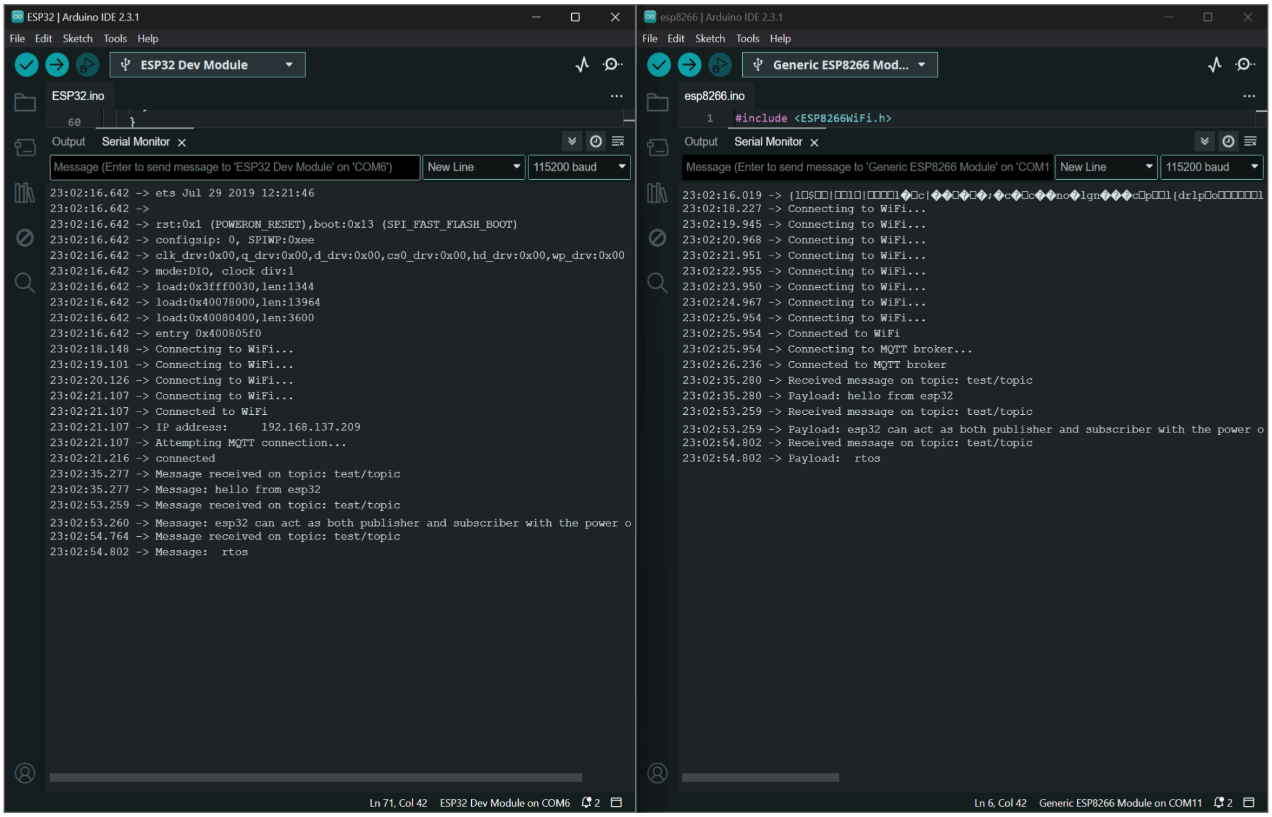
```
    else{
      led1state = 1;
      digitalWrite(data, HIGH);
      delay(500);
    }
    digitalWrite(enable, LOW);
    digitalWrite(s1,HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }

  if(digitalRead(touch2)){
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1,LOW);
    digitalWrite(s0, HIGH);
    digitalWrite(enable, HIGH);
    if(led2state){
      led2state = 0;
      digitalWrite(data, LOW);
      delay(500);
    }
    else{
      led2state = 1;
      digitalWrite(data, HIGH);
      delay(500);
    }
    digitalWrite(enable, LOW);
    digitalWrite(s1,HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }
  if (digitalRead(touch3)) {
    digitalWrite(light, HIGH);
    touch3StartTime = millis();
  }

  if (millis() - touch3StartTime >= touch3Delay) {
    digitalWrite(light, LOW);
  }
}

void handleRoot() {
  if (server.method() == HTTP_POST) {
    String inputValue = server.arg("input");
    processInput(inputValue);
  }

String htmlContent = R"(
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ESP32 Web Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: black;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
        .container {
            background-color: #fff;
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
            max-width: 400px;
            width: 100%;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        form {
            margin-top: 20px;
            text-align: center;
        }
        label {
            display: block;
            margin-bottom: 10px;
            font-weight: bold;
        }
        input[type="text"] {
            width: 100%;
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
            box-sizing: border-box;
        }
        input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to your ESP32!</h1>
        <form method="post">
            <label for="input">Enter your command:</label><br>
            <input type="text" id="input" name="input"><br>
            <input type="submit" value="Submit">
        </form>
    </div>
</body>
</html>
)";


  server.send(200, "text/html", htmlContent);
}

void handleStatus() {
  String statusResponse = "{\"light1\": \"on\", \"light2\": \"off\"}";
  server.send(200, "application/json", statusResponse);
}

void processInput(String inputValue) {
  Serial.print("Input received: ");
  Serial.println(inputValue);

  if(inputValue == "1"){
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1,LOW);
    digitalWrite(s0, LOW);
    digitalWrite(enable, HIGH);
    if(led1state){
      led1state = 0;
      digitalWrite(data, LOW);
      delay(500);
    }
    else{
      led1state = 1;
      digitalWrite(data, HIGH);
      delay(500);
    }
    digitalWrite(enable, LOW);
    digitalWrite(s1,HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }

```



```
    digitalWrite(s1,HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }
  if (inputValue == "3") {
    digitalWrite(light, HIGH);
    touch3StartTime = millis();
  }
}


void mqttBrokerTask(void * parameter) {
  while(1) {
    broker.update();
    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

void remoteTask(void* parameter){
  ;
}

void publisherTask(void * parameter) {
  while(1) {
    const char* topic = "test/topic";
    char message[64];
    while(!Serial.available()){
      ;
    }
    Serial.readBytes(message, sizeof(message));
    removeNewline(message);
    broker.publish(topic, message);
    memset(message, 0, sizeof(message));
    vTaskDelay(pdMS_TO_TICKS(500));
  }
}


void subscriberTask(void * parameters){
  client.setServer(WiFi.localIP(), 1883);
  client.setCallback(callback);
  while(1){
    if (!client.connected()) {
      reconnect();
    }
    client.loop();
    vTaskDelay(pdMS_TO_TICKS(100));
  }
}

void removeNewline(char* str) {
    int len = strlen(str);
    for (int i = 0; i < len; i++) {
        if (str[i] == '\n' || str[i] == '\r') {
            str[i] = '\0';
            break;
        }
    }
}

void reconnect() {
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    if (client.connect("ESP32Client")) {
      Serial.println("connected");
      client.subscribe("test/topic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.println(topic);
  Serial.print("Message: ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
  }
  Serial.println();

  String messageStr;
  for (int i = 0; i < length; i++) {
    messageStr += (char)message[i];
  }

  if (messageStr.equals("1")) {
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1, LOW);
    digitalWrite(s0, LOW);
    digitalWrite(enable, HIGH);
    if (led1state) {
      led1state = 0;
      digitalWrite(data, LOW);
      delay(500);
    } else {
      led1state = 1;
      digitalWrite(data, HIGH);
      delay(500);
    }
    digitalWrite(enable, LOW);
    digitalWrite(s1, HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }
  if (messageStr.equals("2")) {
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1, LOW);
    digitalWrite(s0, HIGH);
    digitalWrite(enable, HIGH);
    if (led2state) {
      led2state = 0;
      digitalWrite(data, LOW);
      delay(500);
    } else {
      led2state = 1;
      digitalWrite(data, HIGH);
      delay(500);
    }
    digitalWrite(enable, LOW);
    digitalWrite(s1, HIGH);
    digitalWrite(s0, HIGH);
    xSemaphoreGive(semaphore);
  }
  if (messageStr.equals("3")) {
    digitalWrite(light, HIGH);
    touch3StartTime = millis();
  }
  if (messageStr.equals("69")){
    xSemaphoreTake(semaphore, portMAX_DELAY);
    digitalWrite(s1,LOW);
    digitalWrite(s0,LOW);
    digitalWrite(enable,HIGH);
    digitalWrite(light, HIGH);
    for(int i = 0; i<=100; i++){
      touch3StartTime = millis();
      digitalWrite(data,HIGH);
      delay(50);
      digitalWrite(s0, HIGH);
      delay(50);
      digitalWrite(data,LOW);
      delay(50);
      digitalWrite(s0,LOW);
      delay(50);
    }
    digitalWrite(light, LOW);
    digitalWrite(enable,LOW);
    digitalWrite(s1,HIGH);
    digitalWrite(s0,HIGH);
    xSemaphoreGive(semaphore);
  }
}
```

⚠ Connection lost. Cloud sketch actions and updates won't be available.

Output



```
}
```

Message (Enter to send message to 'ESP32 Dev Module' on 'COM6') New Line 115200 baud

```
23:02:16.642 -> ets Jul 29 2019 12:21:46
23:02:16.642 ->
23:02:16.642 -> rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
23:02:16.642 -> configsip: 0, SPIWP:0xee
23:02:16.642 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
23:02:16.642 -> mode:DIO, clock div:1
23:02:16.642 -> load:0x3fff0030,len:1344
23:02:16.642 -> load:0x40078000,len:13964
23:02:16.642 -> load:0x40080400,len:3600
23:02:16.642 -> entry 0x400805f0
23:02:18.148 -> Connecting to WiFi...
23:02:19.101 -> Connecting to WiFi...
23:02:20.126 -> Connecting to WiFi...
23:02:21.107 -> Connecting to WiFi...
23:02:21.107 -> Connected to WiFi
23:02:21.107 -> IP address:     192.168.137.209
23:02:21.107 -> Attempting MQTT connection...
23:02:21.216 -> connected
23:02:35.277 -> Message received on topic: test/topic
23:02:35.277 -> Message: hello from esp32
23:02:53.259 -> Message received on topic: test/topic
23:02:53.260 -> Message: esp32 can act as both publisher and subscriber with the power o
23:02:54.764 -> Message received on topic: test/topic
23:02:54.802 -> Message:  rtos
```

Ln 71, Col 42 ESP32 Dev Module on COM6 2

```
#include <ESP8266WiFi.h>
```

Message (Enter to send message to 'Generic ESP8266 Module' on 'COM1 New Line 115200 baud

```
23:02:16.019 -> {1□$□□|□□1□|□□□□1�□c|��□�□�;�c�□c��no�lgn���c□p□□l{drlp□o□□□□□□l
23:02:18.227 -> Connecting to WiFi...
23:02:19.945 -> Connecting to WiFi...
23:02:20.968 -> Connecting to WiFi...
23:02:21.951 -> Connecting to WiFi...
23:02:22.955 -> Connecting to WiFi...
23:02:23.950 -> Connecting to WiFi...
23:02:24.967 -> Connecting to WiFi...
23:02:25.954 -> Connecting to WiFi...
23:02:25.954 -> Connected to WiFi
23:02:25.954 -> Connecting to MQTT broker...
23:02:26.236 -> Connected to MQTT broker
23:02:35.280 -> Received message on topic: test/topic
23:02:35.280 -> Payload: hello from esp32
23:02:53.259 -> Received message on topic: test/topic
23:02:53.259 -> Payload: esp32 can act as both publisher and subscriber with the power o
23:02:54.802 -> Received message on topic: test/topic
23:02:54.802 -> Payload:  rtos
```

Ln 6, Col 42 Generic ESP8266 Module on COM11 2

Applications

•Energy efficiency: The home automation system can be designed to optimize energy usage by controlling the lighting, heating, and cooling systems based on occupancy, time of day, and weather conditions.
•Security: The home automation system can be integrated with security cameras, motion sensors, and door locks to provide enhanced security and surveillance
•Convenience: The home automation system can provide users with a seamless and intuitive experience by allowing them to control their devices through voice assistants, mobile applications, and web services
•Sensor integration: The home automation system can be designed to integrate with various sensors, such as temperature, humidity, and air quality sensors, to provide users with real-time data and insights about their home environment
•Remote access: The home automation system can be accessed remotely, allowing users to control their devices and monitor their home from anywhere in the world
•Personalization: The home automation system can be customized to meet the specific needs and preferences of individual users, such as setting up personalized lighting and temperature profiles.

Conclusion

In conclusion, the development and implementation of a home automation system utilizing a Real-Time Operating System (RTOS) represent a significant advancement in the field of smart home technology. Through the meticulous application of OS concepts such as real-time scheduling, resource management, and inter-task communication, the system has demonstrated its ability to efficiently manage real-time tasks, resources, and communication within the home automation framework.

The utilization of FreeRTOS has provided a solid foundation for orchestrating seamless interaction between nodes distributed throughout the home and a central server, enabling comprehensive control and monitoring of home automation functions. By integrating with IoT devices and smart home appliances, the system offers users an intuitive and seamless experience, enhancing comfort, efficiency, and overall quality of life.

Furthermore, the project has highlighted the importance of scalability and modularity in designing a home automation system that can evolve with changing requirements and preferences. The modular architecture allows for the seamless addition of new nodes, sensors, and devices, ensuring adaptability and future-proofing the system against technological advancements.

Moving forward, continued research and development in this field will further enhance the capabilities and functionalities of home automation systems utilizing RTOS. With the ever-increasing integration of IoT devices and smart home appliances into our daily lives, the demand for intelligent, responsive, and user-centric home automation solutions will only continue to grow. By leveraging the power of RTOS and embracing emerging technologies, we can unlock new possibilities and usher in a new era of smart living.