Blue text on a black background

Description automatically generated

***Σχολή Μηχανικών***

***Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών***

***Διαδίκτυο των Αντικειμένων***

***Τελική Εργασία***

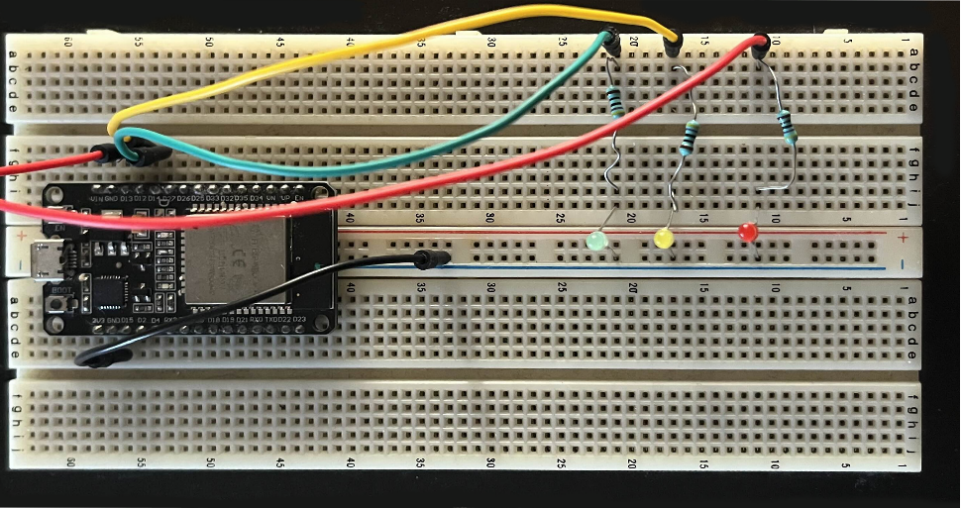
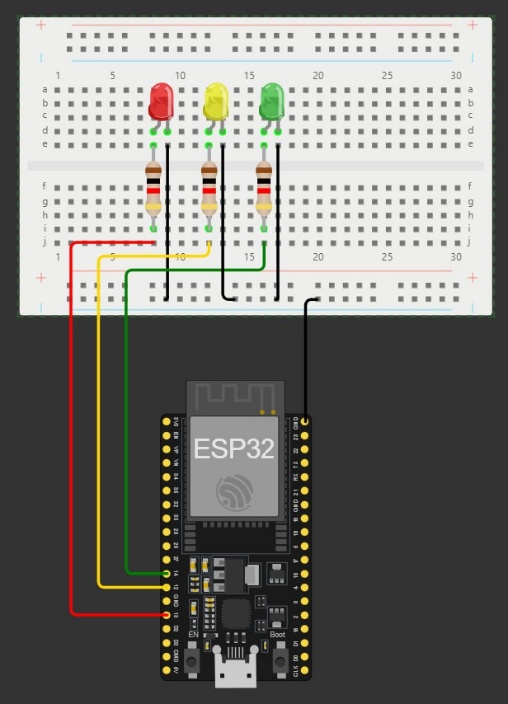
*ΜΕΝΤΖΕΛΟΣ ΑΓΓΕΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 21390132*

*ΚΟΝΤΟΥΛΗΣ ΔΗΜΗΤΡΙΟΣ 21390095*

*ΕΥΦΡΟΣΥΝΗ ΒΑΡΣΟΥ 21390021*

Εισαγωγή

Η εφαρμογή αποτελείται από έναν φωτεινο σηματοδότη (3 LED διαφορετικού χρώματος) του οποίου η λειτουργία ορίζεται από τον μικροελεγκτή ESP32 και απεικονίζεται στην πλατφόρμα IoT ThingSpeak. Η λειτουργικότητα της εφαρμογής αναλύεται σε τρία στάδια που θα αναλύσουμε στην συνέχεια σε κάθε δραστηριότητα. Ειδικότερα, στην πρώτη δραστηριότητα περιγράφεται πως γίνεται ο προγραμματισμός εναλλαγής των LED καθώς και η αποστολή της καταστασής τους σε κανάλι του ThingSpeak. Στην δευτερη και τρίτη δραστηριότητα απλά προσθέτουμε έξτρα λειτουργίες (μεταβλητή ειδοποιήσης) που αφορούν στην πλατφόρμα του ThingSpeak.



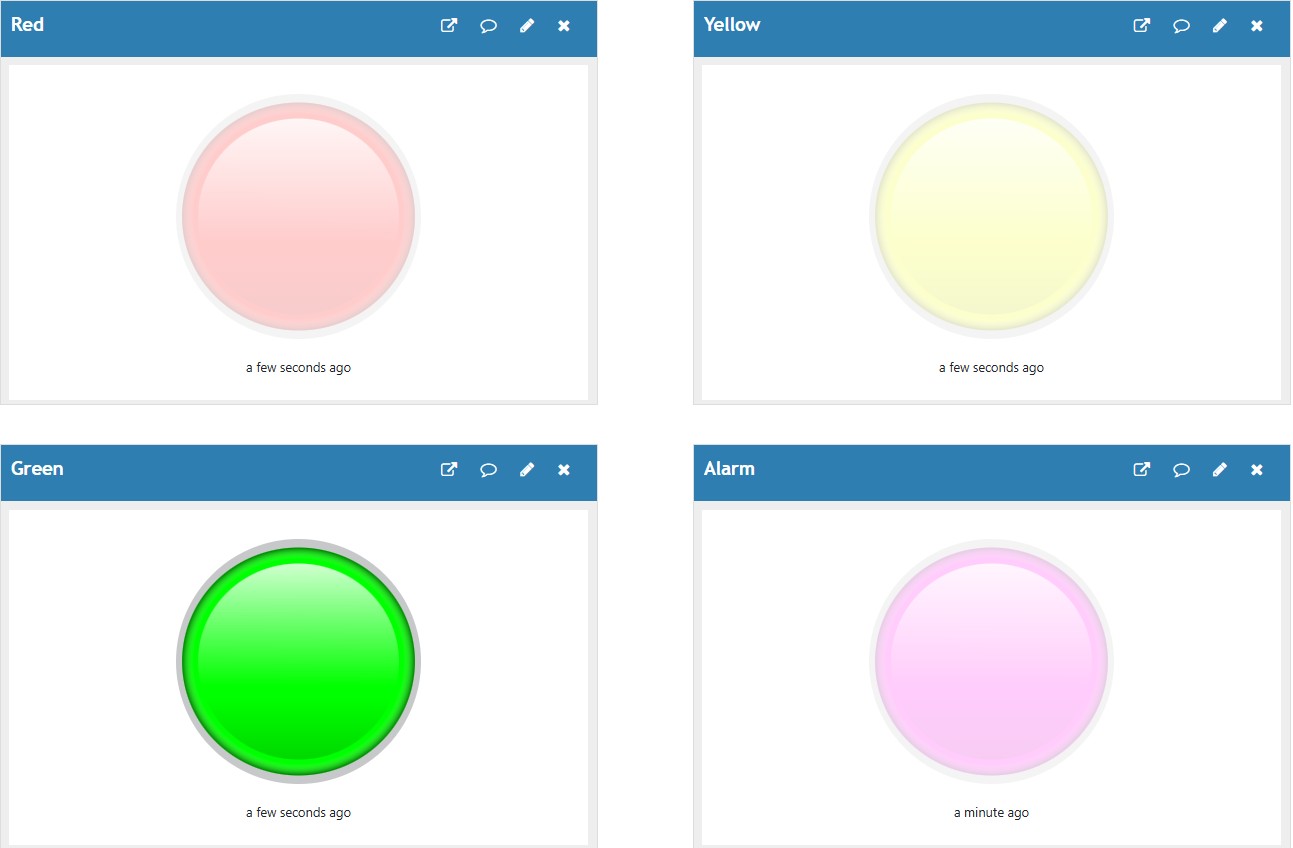
*Εικόνα 1: Προσομοιωτικη Απεικόνιση Κυκλώματος Εικόνα 2: Πραγματική Απεικόνιση Κυκλώματος*

Υλοποιήση Κυκλώματος Υλικού

Για την υλοποίηση του κυκλώματος επιλέξαμε να χρησιμοποιήσουμε τον μικροελεγκτή ESP32 καθώς έχει ενσωματωμένη την δυνατότητα σύνδεσης Wi-Fi καθιστώντας έτσι εύκολα την σύνδεση στην πλατφόρμα IoT. Επίσης, δεν χρειαζόταν να αλλάξουμε πλατφόρμα προγραμματισμού κάθως το ESP32 χρησιμοποιέι και αυτό την πλατφορμα Arduino IDE. Η υλοποιήση του κυκλώματος είναι σχετικά απλή (Εικόνα 1,2). Κάθε LED συνδέεται με μία έξοδο GPIO του ESP32 και μια αντίσταση σε σειρά, η οποία περιορίζει το ρεύμα που περνάει μέσα από το LED και το αρνητικό άκρο τoυ κάθε LED συνδέεται με το GND.

Κανάλι Thingspeak

Για την προβολή τον LED από το κανάλι του ThingSpeak, αρχικά στις ρυθμίσεις καναλιού(Channel Settings) ενεργοποιήσαμε τα πρώτα τρία field (Field1, Field2, Field3) και στην συνέχεια δημιουργήσαμε τρία Widgets «Lamp Indicators» αναθέτοντας για κάθε field ένα widget με διαφορετικό χρώμα (Εικόνα 3). Επιπλέον, όπως θα δούμε στις δραστηριότητες 2,3 έχουμε και ένα τέταρτο widget σε χρώμα μωβ που αντιστοιχει στο field8 το οποίο αποτελεί την μεταβλητη ειδοποιήσης.

*Εικόνα 3: Κανάλι Thingspeak*

Δραστηριότητα 1. Ηλεκτρικός φωτεινός σηματοδότης

#include <WiFi.h>

#include <HTTPClient.h>

const char\* ssid = "YOUR\_WIFI\_SSID";

const char\* password = "YOUR\_PASSWORD";

String api = "YOUR\_API\_KEY";

// LED pin setup

int greenLED = 14;  // Pin for green LED

int yellowLED = 12; // Pin for orange LED

int redLED = 13;   // Pin for red LED

void setup() {

  Serial.begin(115200);

  // Set up LED pins

  pinMode(greenLED, OUTPUT);

  pinMode(yellowLED, OUTPUT);

  pinMode(redLED, OUTPUT);

  WiFi.begin(ssid,password);

  Serial.print("Connecting to WiFi...");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("\nConnected to WiFi");

}

void sendData(String green, String orange, String red) {

  String url = "http://api.thingspeak.com/update?api\_key=" + api + "&field1=" + green + "&field2=" + orange + "&field3=" + red;

  if (WiFi.status() == WL\_CONNECTED) { // Check if connected to WiFi

    HTTPClient http;

    http.begin(url); // Specify the URL

    int httpResponseCode = http.GET(); // Send the GET request

    if (httpResponseCode > 0) {

      Serial.print("HTTP Response code: ");

      Serial.println(httpResponseCode);

    } else {

      Serial.print("Error on sending request: ");

      Serial.println(httpResponseCode);

    }

    http.end(); // Free resources

  } else {

    Serial.println("WiFi Disconnected");

  }

}

void controlTrafficLight(String state) {

  if (state == "green") {

    digitalWrite(greenLED, HIGH);

    digitalWrite(yellowLED, LOW);

    digitalWrite(redLED, LOW);

  } else if (state == "orange") {

    digitalWrite(greenLED, LOW);

    digitalWrite(yellowLED, HIGH);

    digitalWrite(redLED, LOW);

  } else if (state == "red") {

    digitalWrite(greenLED, LOW);

    digitalWrite(yellowLED, LOW);

    digitalWrite(redLED, HIGH);

  }

}

void loop() {

  // Simulate traffic light control

  controlTrafficLight("red");

  sendData("0", "0", "1");

  delay(20000);

  controlTrafficLight("green");

  sendData("1", "0", "0");

  delay(20000);

  controlTrafficLight("orange");

  sendData("0", "1", "0");

  delay(15000);

}

void sendData(String green, String orange, String red) {

  String url = "http://api.thingspeak.com/update?api\_key=" + api + "&field1=" + green + "&field2=" + orange + "&field3=" + red;

  if (WiFi.status() == WL\_CONNECTED) { // Check if connected to WiFi

    HTTPClient http;

    http.begin(url); // Specify the URL

    int httpResponseCode = http.GET(); // Send the GET request

    if (httpResponseCode > 0) {

      Serial.print("HTTP Response code: ");

      Serial.println(httpResponseCode);

    } else {

      Serial.print("Error on sending request: ");

      Serial.println(httpResponseCode);

    }

    http.end(); // Free resources

  } else {

    Serial.println("WiFi Disconnected");

  }

}

void controlTrafficLight(String state) {

  if (state == "green") {

    digitalWrite(greenLED, HIGH);

    digitalWrite(yellowLED, LOW);

    digitalWrite(redLED, LOW);

  } else if (state == "orange") {

    digitalWrite(greenLED, LOW);

    digitalWrite(yellowLED, HIGH);

    digitalWrite(redLED, LOW);

  } else if (state == "red") {

    digitalWrite(greenLED, LOW);

    digitalWrite(yellowLED, LOW);

    digitalWrite(redLED, HIGH);

  }

}

void loop() {

  // Simulate traffic light control

  controlTrafficLight("red");

  sendData("0", "0", "1");

  delay(30000);

  controlTrafficLight("green");

  sendData("1", "0", "0");

  delay(30000);

  controlTrafficLight("orange");

  sendData("0", "1", "0");

  delay(20000);

}

Σε αυτήν την δραστηρίοτητα, ορίζουμε την βασική δομή της εφαρμογής η οποία προγραμματίζει τις εναλλαγες των LED και στέλνει την κατασταση τους στο κανάλι ThingSpeak. Αρχικά, ορίζουμε το Wifi SSID και Password που θα συνδέσουμε το ESP32 καθώς και το Write API Key του καναλιού Thingspeak στο οποίο θα ενημερώνουμε τις μεταβλητές.

Στην συνάρτηση setup(), αναθέτουμε κάθε LED με τα αντίστοιχα GPIO pins που έχουμε συνδέσει στο κύκλωμα. Στην συνέχεια, αρχικοποιηούμε την σύνδεση Wifi με την συνάρτηση Wifi.begin() και μέσω της επανάληψης while (WiFi.status() != WL\_CONNECTED) τσεκάρουμε αν έχουμε επιτυχής σύνδεση και όταν βρεθεί βγαίνουμε από την while έτσι έχοντας εξασφαλίσει επιτυχής σύνδεση στο Internet.

Στην συνάρτηση sendData(), δημιουργούμε το URL που θα χρησιμοποιήσουμε για να στείλουμε τα δεδομένα στο ThingSpeak. Το URL περιλαμβάνει το Write API key του καναλιού και τις τιμές των LED που θέλουμε να ενημερώσουμε. Ελέγχουμε αν είμαστε συνδεδεμένοι στο Wifi και αν ναι, δημιουργούμε ένα αντικείμενο HTTPClient και χρησιμοποιούμε την begin() για να ορίσουμε το URL. Στην συνέχεια, στέλνουμε το αίτημα GET και ελέγχουμε τον κωδικό απόκρισης για να δούμε αν το αίτημα ήταν επιτυχές και αν ναι το τυπώνουμε. Τέλος, καλούμε την μέθοδο end() για να ελευθερώσουμε τους πόρους που χρησιμοποιήθηκαν.

Η συνάρτηση controlTrafficLight(), είναι υπεύθυνη για τον έλεγχο των κατάστασεων των LED. Δέχεται ως παράμετρο ένα string που καθορίζει ποιο LED είναι σε κατάσταση ON ("green", "orange", "red"). Ανάλογα με την τιμή της, η συνάρτηση ενεργοποιεί το αντίστοιχο LED και απενεργοποιεί τα υπόλοιπα. Αυτό επιτυγχάνεται με τη χρήση της συνάρτησης digitalWrite() που ορίζει την κατάσταση των GPIO pins στα οποία είναι συνδεδεμένα τα LED.

Η συνάρτηση loop() είναι υπεύθυνη για την επαναλαμβανόμενη εκτέλεση των εντολών που ελέγχουν την εναλλαγή των LED και στέλνουν τα δεδομένα στο ThingSpeak. Αρχικά, καλούμε τη συνάρτηση controlTrafficLight() με την παράμετρο "red" για να ενεργοποιήσουμε το κόκκινο LED και να απενεργοποιήσουμε τα υπόλοιπα. Στη συνέχεια, καλούμε τη συνάρτηση sendData() με τις τιμές "0", "0", "1" για να ενημερώσουμε το ThingSpeak ότι το κόκκινο LED είναι ενεργοποιημένο. Μετά από μια καθυστέρηση 20 δευτερολέπτων (20000 milliseconds), επαναλαμβάνουμε τη διαδικασία για το πράσινο και το πορτοκαλί LED, με τις αντίστοιχες τιμές για το ThingSpeak και τις αντίστοιχες καθυστερήσεις.

Δραστηριότητα 2. Αποστολή δεδομένων σε κανάλι άλλης εφαρμογής

#include <WiFi.h>

#include <HTTPClient.h>

const char\* ssid = "YOUR\_WIFI\_SSID";

const char\* password = "YOUR\_PASSWORD";

String api = "YOUR\_WRITE\_API\_KEY"; // Write Thingspeak Channel API Key

String sec\_api = "OTHER\_WRITE\_API\_KEY"; // Write Thingspeak Channel API Key for the other app

// LED pin setup

int greenLED = 14; // Pin for green LED

int yellowLED = 12; // Pin for orange LED

int redLED = 13; // Pin for red LED

void setup() {

  Serial.begin(115200);

  // Set up LED pins

  pinMode(greenLED, OUTPUT);

  pinMode(yellowLED, OUTPUT);

  pinMode(redLED, OUTPUT);

  WiFi.begin(ssid,password);

  Serial.print("Connecting to WiFi...");

  while (WiFi.status() != WL\_CONNECTED) {

    delay(1000);

    Serial.print(".");

  }

  Serial.println("\nConnected to WiFi");

// Set field8 to 0 for both channels

  setField8("0", sec\_api);

  setField8("0", api);

}

void setField8(String value, String api) {

  // Construct and send the HTTP GET request

  String url = "http://api.thingspeak.com/update?api\_key=" + api + "&field8=" + value;

   if (WiFi.status() == WL\_CONNECTED) { // Check if connected to WiFi

    HTTPClient http;

    http.begin(url); // Specify the URL

    int httpResponseCode = http.GET(); // Send the GET request

    if (httpResponseCode > 0) {

      Serial.print("HTTP Response code: ");

      Serial.println(httpResponseCode);

    } else {

      Serial.print("Error on sending request: ");

      Serial.println(httpResponseCode);

    }

    http.end(); // Free resources

  } else {

    Serial.println("WiFi Disconnected");

  }

}

void loop() {

  setField8("0", sec\_api); // Set Field 8 of other channel to 0

  setField8("0", api); // set Field 8 of our channel to 0

  // Simulate traffic light control

  controlTrafficLight("red");

  sendData("0", "0", "1");

  delay(20000);

  controlTrafficLight("green");

  sendData("1", "0", "0");

  delay(20000);

  controlTrafficLight("orange");

  sendData("0", "1", "0");

  delay(15000);

}

void setField8(String value, String api) {

  // Construct and send the HTTP GET request

  String url = "http://api.thingspeak.com/update?api\_key=" + api + "&field8=" + value;

   if (WiFi.status() == WL\_CONNECTED) { // Check if connected to WiFi

    HTTPClient http;

    http.begin(url); // Specify the URL

    int httpResponseCode = http.GET(); // Send the GET request

    if (httpResponseCode > 0) {

      Serial.print("HTTP Response code: ");

      Serial.println(httpResponseCode);

    } else {

      Serial.print("Error on sending request: ");

      Serial.println(httpResponseCode);

    }

    http.end(); // Free resources

  } else {

    Serial.println("WiFi Disconnected");

  }

}

Σε αυτήν την δραστηριότητα, τροποποιούμε την υπάρχουσα δομή της πρώτης δραστηριότητας και προσθέτουμε την δυνατότητα ορισμού τιμής για μια μεταβλητή άλλου καναλιού. Ειδικότερα, θα θέσουμε στην μεταβλητή Field8 (Μωβ Widget Alarm) του καναλιού με API Key == sec\_api την τιμή 0 καθώς και στο κανάλι μας με API Key == api. Οι συναρτήσεις sendData(), controlTrafficLight() και loop() παραμένουν ίδιες με την πρώτη δραστηρίοτητα και για αυτό το λόγο δεν θα αναλυθούν ξανά.

Η συνάρτηση setup() παραμένει ίδια με την πρώτη δραστηριότητα, με μόνη διαφορά την κλήση της συνάρτησης setField8() και για τα δύο κανάλια, που αρχικοποιεί την τιμή του field8 σε 0. Η συνάρτηση setField8() είναι υπεύθυνη για την αποστολή της τιμής value (0 ή 1) στο field8 του καναλιού με παράμετρο api.H λειτουργία της είναι ανάλογη της sendData() με την διαφορά μόνο στο URL που θα κληθει για το GET request. To URL που καλεί αλλάζει την μεταβλητή field8 για όποιο κανάλι έχει δοθεί ως παράμετρος.

Δραστηριότητα 3. Ανάγνωση δεδομένων σε κανάλι άλλης εφαρμογής

int getField8(String api) {

  String url = "http://api.thingspeak.com/channels/2704086/fields/8.json?api\_key=" + api + "&results=1";

  HTTPClient http;

  http.begin(url); // Specify the URL

  int httpResponseCode = http.GET(); // Send the GET request

  if (httpResponseCode == 200) {

    String payload = http.getString();

    Serial.println("JSON Response: ");

    Serial.println(payload);  // Print the full response

    // Call helper functions to parse the JSON and extract 'field8' value

    String feedsSection = extractFeedsSection(payload);

    int field8Value = extractField8Value(feedsSection);

    return field8Value;

  } else {

    Serial.print("Error on sending request: ");

    Serial.println(httpResponseCode);

    return -1;  // Return -1 if request fails

  }

  http.end(); // Free resources

}

// Function to extract the 'feeds' section from the JSON response

String extractFeedsSection(String payload) {

  int startIdx = payload.indexOf("\"feeds\":[");

  int endIdx = payload.indexOf("]}", startIdx); // Closing braces for the feeds array

  if (startIdx == -1 || endIdx == -1) {

    Serial.println("Error: 'feeds' section not found.");

    return "";  // Return empty if 'feeds' section is not found

  }

  return payload.substring(startIdx + 8, endIdx + 1); // Extract the feeds section

}

// Function to extract the 'field8' value from the 'feeds' section

int extractField8Value(String feedsSection) {

  int field8Pos = feedsSection.indexOf("\"field8\":\"");

  if (field8Pos == -1) {

    Serial.println("Error: 'field8' value not found.");

    return -1; // Return -1 if 'field8' is not found

  }

  String field8String = feedsSection.substring(field8Pos + 10, field8Pos + 11); // Extract "1" or "0"

  return field8String.toInt();  // Convert to integer and return

}

long lastField8Update = 0;

long lastCycleUpdate = 0;

const long field8Interval = 1 \* 60 \* 1000;  // 1 minute (60 seconds)

const long field8Duration = 20 \* 1000;      // 20 seconds

bool field8Active = false;

void loop() {

  unsigned long currentTime = millis();  // Get the current time in milliseconds

  if (currentTime - lastField8Update >= field8Interval && !field8Active) {

    setField8("1", api);  // Activate Field 8

    field8Active = true;

    lastField8Update = currentTime;

    Serial.println("Field 8 activated");

  }

  // Check if Field 8 has been active for 1 minute, then deactivate it

  if (field8Active && currentTime - lastField8Update >= field8Duration) {

    setField8("0", api);  // Deactivate Field 8

    field8Active = false;

    lastField8Update = currentTime;

    Serial.println("Field 8 deactivated");

  }

  // Get the current value of Field 8

  int field8Value = getField8(sec\_api);

  Serial.print("Current Field 8 value: ");

  Serial.println(field8Value);

  // Override normal traffic light behavior if Field 8 is active

  if (field8Value == 1) {

    controlTrafficLight("orange");

    sendData("0", "1", "0");  // Send data for orange light

  } else {

    // Normal traffic light behavior

    if (currentTime - lastCycleUpdate >= 20000) {  // 20 seconds per light

      controlTrafficLight("red");

      sendData("0", "0", "1");  // Send data for red light

      delay(20000);

      controlTrafficLight("green");

      sendData("1", "0", "0");  // Send data for green light

      delay(20000);

      controlTrafficLight("orange");

      sendData("0", "1", "0");  // Send data for orange light

      delay(15000);

      lastCycleUpdate = currentTime;  // Update the last cycle time

    }

  }

}

// Function to extract the 'feeds' section from the JSON response

String extractFeedsSection(String payload) {

int startIdx = payload.indexOf("\"feeds\":[");

int endIdx = payload.indexOf("]}", startIdx); // Closing braces for the feeds array

if (startIdx == -1 || endIdx == -1) {

Serial.println("Error: 'feeds' section not found.");

return ""; // Return empty if 'feeds' section is not found

}

return payload.substring(startIdx + 8, endIdx + 1); // Extract the feeds section

}

// Function to extract the 'field8' value from the 'feeds' section

int extractField8Value(String feedsSection) {

int field8Pos = feedsSection.indexOf("\"field8\":\"");

if (field8Pos == -1) {

Serial.println("Error: 'field8' value not found.");

return -1; // Return -1 if 'field8' is not found

}

String field8String = feedsSection.substring(field8Pos + 10, field8Pos + 11); // Extract "1" or "0"

return field8String.toInt(); // Convert to integer and return

}

long lastField8Update = 0;

long lastCycleUpdate = 0;

const long field8Interval = 1 \* 60 \* 1000; // 1 minute (60 seconds)

const long field8Duration = 20 \* 1000; // 20 seconds

bool field8Active = false;

void loop() {

unsigned long currentTime = millis(); // Get the current time in milliseconds

if (currentTime - lastField8Update >= field8Interval && !field8Active) {

setField8("1", api); // Activate Field 8

field8Active = true;

lastField8Update = currentTime;

Serial.println("Field 8 activated");

}

// Check if Field 8 has been active for 1 minute, then deactivate it

if (field8Active && currentTime - lastField8Update >= field8Duration) {

setField8("0", api); // Deactivate Field 8

field8Active = false;

lastField8Update = currentTime;

Serial.println("Field 8 deactivated");

}

// Get the current value of Field 8

int field8Value = getField8(sec\_api);

Serial.print("Current Field 8 value: ");

Serial.println(field8Value);

// Override normal traffic light behavior if Field 8 is active

if (field8Value == 1) {

controlTrafficLight("orange");

sendData("0", "1", "0"); // Send data for orange light

} else {

// Normal traffic light behavior

if (currentTime - lastCycleUpdate >= 20000) { // 20 seconds per light

controlTrafficLight("red");

sendData("0", "0", "1"); // Send data for red light

delay(20000);

controlTrafficLight("green");

sendData("1", "0", "0"); // Send data for green light

delay(20000);

controlTrafficLight("orange");

sendData("0", "1", "0"); // Send data for orange light

delay(15000);

lastCycleUpdate = currentTime; // Update the last cycle time

}

}

}