Blue text on a black background

Description automatically generated

***Σχολή Μηχανικών***

***Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών***

***Διαδίκτυο των Αντικειμένων***

***Τελική Εργασία***

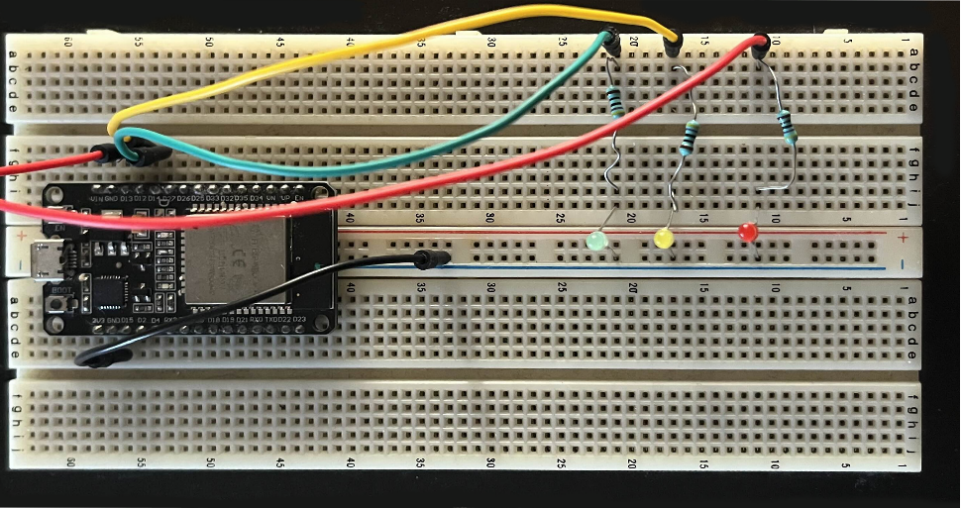
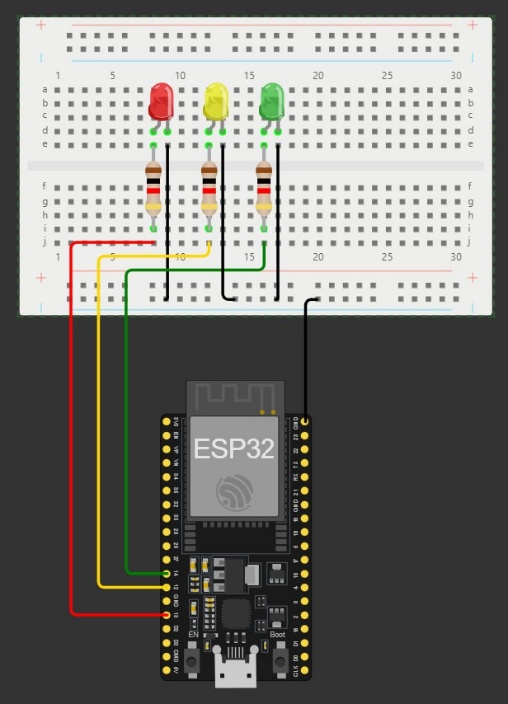
*ΜΕΝΤΖΕΛΟΣ ΑΓΓΕΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 21390132*

*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.



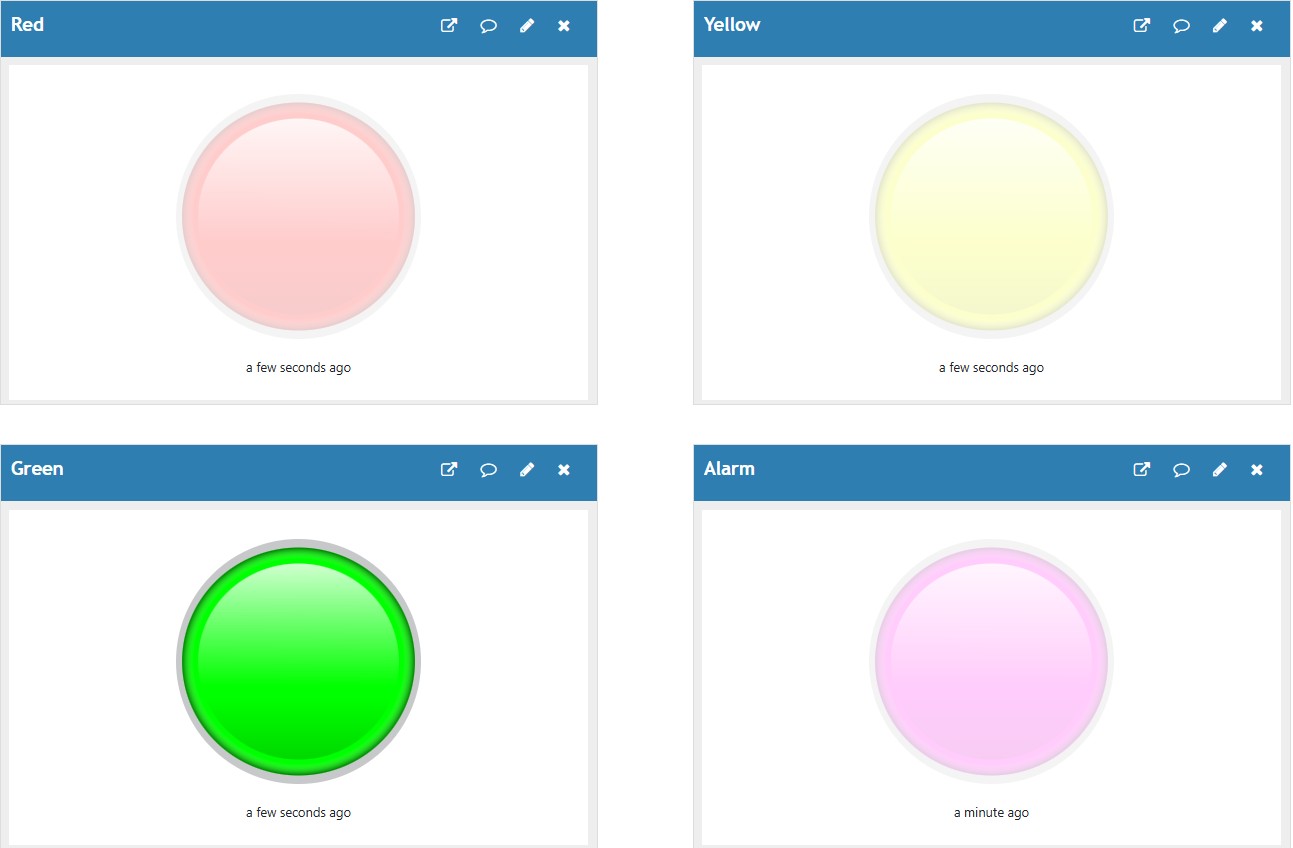
*Εικόνα 1: Προσομοιωτικη Απεικόνιση Κυκλώματος Εικόνα 2: Πραγματική Απεικόνιση Κυκλώματος*

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";

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");

}

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(20000);

  controlTrafficLight("green");

  sendData("1", "0", "0");

  delay(20000);

  controlTrafficLight("orange");

  sendData("0", "1", "0");

  delay(15000);

}

Περιγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do

Περιγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur

Δραστηριότητα 2. Αποστολή δεδομένων σε κανάλι άλλης εφαρμογής

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");

  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);

}

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore Περιγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elitΠεριγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elitΠεριγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit

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

    }

  }

}

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

    }

  }

}

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore Περιγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elitΠεριγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Lorem ipsum dolor sit amet, consectetur adipiscing elitΠεριγραφη κώδικα και λειτουργιας δραστηριοτητας Lorem ipsum dolor sit