ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

IoT και Γεωργία Ακριβείας / Έξυπνο Σπίτι

	ΟΝΟΜΑ	**ΑΜ**
1	ΑΛΦΟΝΣΟΣ -ΛΕΩΝΙΔΑΣ ΚΩΤΣΙΟΣ	E21083
2	ΚΩΣΤΑΝΤΙΝΟΣ ΖΑΦΕΙΡΟΠΟΥΛΟΣ	E19044

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγή

Στα πλαίσια της εργασίας του μαθήματος Διαδίκτυο των πραγμάτων επιλέξαμε την θεματική περιοχή IoT και γεωργία ακριβείας , στην οποία περιοχή βασίζεται και η βιβλιογραφική ανασκόπηση που αναλύσαμε με τίτλο : **«Environmental Temperature and Humidity Monitoring at Agricultural Farms using Internet of Things & DHT22-Sensor»**

Στην πορεία παρεκκλίναμε από το αρχικό θέμα και επικεντρωθήκαμε στο concept του Smart Home καθώς μας φάνηκε πιο ενδιαφέρουσα προσέγγιση με μεγαλύτερο περιθώριο ανάπτυξης. Έχουμε προσπαθήσει να προσφέρουμε προσωποποιημένη χρήση για τον εκάστοτε χρήστη της εφαρμογής.

Η εφαρμογή μας αποτελείται από ένα Web base App σε Python – Flask Framework και από ένα IoT Device ανεπτυγμένο σε Esp32.

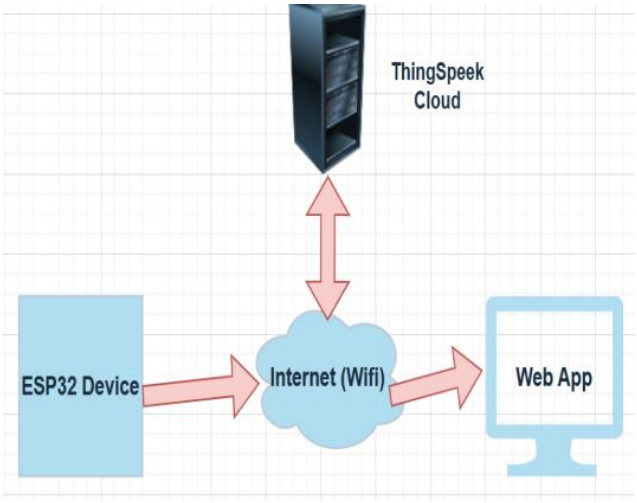
Οι αισθητήρες που χρησιμοποιήσαμε είναι :

- **DHT22 (ΘΕΡΜΟΚΡΑΣΙΑ , ΥΓΡΑΣΙΑ)**
- **Light Sensor (Μέτρηση Φωτός)**

Η συνδεσμολογία έγινε πάνω σε ένα Bread Board με μία αντίσταση και καλώδια σύνδεσης και η επικοινωνία με την εφαρμογή έγινε μέσω της πλατφόρμας ThingSpeak.

Περιγραφή Εφαρμογής

Η πορεία της εργασίας ξεκίνησε με την συνδεσμολογία του IoT Device με βάση τις οδηγίες των εργαστήριών και στην συνέχεια αναπτύξαμε των παρακάτω κώδικα για την συλλογή των μετρήσεων από τους αισθητήρες. Στην συνέχεια προσθέσαμε τον κώδικα σύνδεσης του Device με το cloud ThingSpeak.



ARDUINO & THING SPEAK

Αρχικά στο ThingSpeak δημιουργήσαμε ένα κανάλι με τρία fields τα οποία αντιστοιχούν στις μετρήσεις των αισθητήρων δηλαδή ένα για την μέτρηση του φωτός , ένα για την θερμοκρασία και ένα για την υγρασία.

Στην συνέχεια μέσω των μοναδικών API keys του καναλιού προχωρήσαμε στην σύνδεση με το IoT Device .

```
#include <WiFi.h>
#include "secrets.h"
#include "ThingSpeak.h"  // Include ThingSpeak header file

#include <DHTesp.h>
#define LIGHT_SENSOR_PIN 1
int TempLimit = 0;
int humLimit = 0;
```

Συμπερίληψη βασικών βιβλιοθηκών και αρχικοποίηση μεταβλητών θερμοκρασίας και υγρασίας , αντιστοίχιση pin στο light sensor.

```
char ssid[] = SECRET_SSID;     // Your network SSID (name)
char pass[] = SECRET_PASS;     // Your network password

WiFiClient client;

unsigned long myChannelNumber = SECRET_CH_ID;     // ThingSpeak Channel ID
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;  // Correct Write API Key


#define DHTPIN 0         // Digital pin connected to the DHT sensor
#define DHTTYPE DHT22

DHTesp dht;
```

Άντληση στοιχείων για την σύνδεση και ορισμός pin για τον DHT22, αρχικοποίηση σχετικού αντικειμένου για χρήση των μεθόδων.

Ο κώδικας της συνάρτησης setup εκτελείται μια φορά κατά την εκκίνηση και η λειτουργεία του έχει ως εξής:

- Ενεργοποίηση σειριακής επικοινωνίας.
- Ρύθμιση συσκευής σε Wifi Station mode .
- Αναμονή μέχρι την επιτυχή σύνδεση.
- Εκτύπωση της IP της συσκευής.
- Προετοιμασία σύνδεσης με ThingSpeak, DHT22, Αισθητήρα Φωτός.

```
void setup() {

  Serial.begin(115200);
  while (!Serial) {
    ;
  }

  Serial.println("Connecting to WiFi...");
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);  // Connect to WiFi

  // Wait until the WiFi is connected
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

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

  ThingSpeak.begin(client);  // Initialize ThingSpeak
  delay(1000);
  dht.setup(DHTPIN, DHTesp::DHT22);

  // set the ADC attenuation to 11 dB (up to ~3.3V input)
  analogSetAttenuation(ADC_11db);
}
```

```
void loop() {

// Check WiFi status and reconnect if necessary
  if(WiFi.status() != WL_CONNECTED){
    Serial.println("WiFi connection lost. Reconnecting...");
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass); // Reconnect to WiFi
      delay(5000);
      Serial.print(".");
    }
    Serial.println("Reconnected to WiFi");
  }
```

Εδώ σε περίπτωση που χαθεί η σύνδεση από το διαδίκτυο το σύστημα προσπαθεί διαρκώς να ξανασυνδεθεί.

```
// reads the input on analog pin (value between 0 and 4095)
int analogValue = analogRead(LIGHT_SENSOR_PIN);
TempAndHumidity newValues = dht.getTempAndHumidity();

// Set multiple fields
ThingSpeak.setField(1, analogValue);
ThingSpeak.setField(2, newValues.temperature);
ThingSpeak.setField(3, newValues.humidity);

Serial.print("LIGHT VALUE_FIELD1 = ");
Serial.print(analogValue);   // the raw analog reading


Serial.print(F(" Temperature_FIELD2: "));
Serial.println(newValues.temperature);  // display temperature

Serial.print(F(" Humidity_FIELD3: "));
Serial.println(newValues.humidity);     // display humidity
```

Εδώ γίνεται η ανάγνωση των αισθητήρων όπου διαβάζουμε τις μετρήσεις και έπειτα αντιστοίχιση των Fields του καναλιού στις μετρήσεις του αισθητήρα. Τέλος εμφάνιση μετρήσεων για έλεγχο.

```
// We'll have a few threshholds, qualitatively determined
if (analogValue < 850) {
  Serial.println(" => Dark");
} else if (analogValue < 1000) {
  Serial.println(" => Dim");
} else if (analogValue < 1200) {
  Serial.println(" => Light");
} else if (analogValue < 1500) {
  Serial.println(" => Bright");
} else {
  Serial.println(" => Very bright");
}
```

Βάση της μέτρησης της φωτεινότητας προσθήκη λεκτικής περιγραφής για έλεγχο.

```
int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

  if(x == 200){
  Serial.println("Channel update successful.");
}
else{
  Serial.print("Problem updating channel. HTTP error code ");
  Serial.println(x);
  Serial.println("Error message: ");
  if (x == 401) {
    Serial.println("Unauthorized - Please check the API key.");
  } else if (x == 400) {
    Serial.println("Bad Request - Check field number and API format.");
  } else {
    Serial.println("Other error occurred.");
  }
}


delay(10000);
```

Αποστολή δεδομένων στο κανάλι και εκτύπωση μηνύματος επιτυχίας ή αποτυχίας αντίστοιχα. (αποστολή κάθε 10s)

PYTHON ΕΦΑΡΜΟΓΗ

Όπως αναφέρθηκε η εφαρμογή μας αναπτύχθηκε σε python – flask framework. Ουσιαστικά η εφαρμογή μας παρουσιάζει τις τιμές του IoT Device και παράλληλα έχει ένα Api που παρουσιάζει τον καιρό της Περιοχής. Ο κάθε χρήστης μπορεί να ορίσει τα δικά του κατώφλια(threshold) ώστε η εφαρμογή να επικοινωνεί μαζί του μέσω email για την λήψη των απαραίτητων μέτρων .

Επεξήγηση Κώδικα

```
from flask import Flask, render_template, jsonify, request, redirect, url_for, flash
import requests
import time
from retry_requests import retry
from send_email import (send_alert_email, user_settings, last_alert_times)
import requests

CHANNEL_ID = "2945205"
READ_API_KEY = "GV1LOU25EOX27SW2"
THING_SPEAK_URL = f"https://api.thingspeak.com/channels/{CHANNEL_ID}/feeds/last.json?api_key={READ_API_KEY}"

app = Flask(__name__)
app.secret_key = 'your_secret_key'  # Add this line if not already present
```

Εισαγωγή βιβλιοθηκών , αρχικοποίηση κλειδιών για σύνδεση με ThingSpeak και αρχικοποίηση εφαρμογής.

```python
@app.route('/settings', methods=['GET', 'POST'])
def settings():
    global user_settings
    if request.method == 'POST':
        user_settings['email'] = request.form['email']
        user_settings['LIGHT_THRESHOLD'] = float(request.form['LIGHT_THRESHOLD'])
        user_settings['TEMP_THRESHOLD'] = float(request.form['TEMP_THRESHOLD'])
        user_settings['HUMIDITY_THRESHOLD'] = float(request.form['HUMIDITY_THRESHOLD'])
        user_settings['LIGHT_LOW_THRESHOLD'] = float(request.form['LIGHT_LOW_THRESHOLD'])
        user_settings['TEMP_LOW_THRESHOLD'] = float(request.form['TEMP_LOW_THRESHOLD'])
        user_settings['HUMIDITY_LOW_THRESHOLD'] = float(request.form['HUMIDITY_LOW_THRESHOLD'])
        user_settings['COOLDOWN_SECONDS'] = int(request.form['COOLDOWN_SECONDS'])
        flash('Settings saved successfully!', 'success')
        return redirect(url_for('settings'))
    return render_template('settings.html', settings=user_settings)
```

Ορισμός Route για σελίδα settings για προβολή ή αποθήκευση threshold χρήστη.

```python
def get_weather_data():
    API_KEY = 'b4cbf658a1c946e7a1f144613251605'
    url = f"http://api.weatherapi.com/v1/current.json?key={API_KEY}&q=Athens&aqi=no"
    try:
        response = requests.get(url)
        data = response.json()
        return {
            "city": data['location']['name'],
            "temp_c": data['current']['temp_c'],
            "uv": data['current']['uv'],
            "humidity": data['current']['humidity']
        }
    except Exception as e:
        return {"error": str(e)}
```

Αίτηση για τον καιρό στην Αθήνα από το **weatherapi** και επιστροφή των σχετικών τιμών.

```python
def get_sensor_data():
    try:
        response = requests.get(THING_SPEAK_URL)
        response.raise_for_status()
        data = response.json()
        light = data.get("field1")
        temperature = data.get("field2")
        humidity = data.get("field3")

        light_val = float(light) if light else None
        temp_val = float(temperature) if temperature else None
        humidity_val = float(humidity) if humidity else None

        now = time.time()
```

Συνάρτηση για την λήψη των τελευταίων δεδομένων αισθητήρων από το ThingSpeak.

```python
    # Έλεγχος φωτός
    if light_val and light_val > user_settings["LIGHT_THRESHOLD"]:
        if now - last_alert_times["light"] > user_settings["COOLDOWN_SECONDS"]:
            send_alert_email("Light Alert", f"Light level is too high: {light_val} maybe is time to close the
            curtains!!🌞")
            last_alert_times["light"] = now
```

Εδώ γίνεται ο έλεγχος της τιμής του αισθητήρα σε σχέση με το κατώφλι που έχει ορίσει ο χρήστης έτσι ώστε να ληφθεί η απόφαση για το αν χρειάζεται να σταλθεί η ειδοποίηση.

Αντίστοιχα γίνονται όλοι οι έλεγχοι για όλα τα μέγιστα και τα ελάχιστα κατώφλια σε όλα τα πεδία μετρήσεων.

```python
@app.route('/')
def index():
    sensor_data = get_sensor_data()
    weather_data = get_weather_data()
    return render_template('index.html', sensor_data=sensor_data, weather_data=weather_data)

@app.route('/data')
def data():
    return j

if __name__
    app.run(debug=True)
```

```
(parameter) debug: bool | None

debug
if given, enable or disable debug mode. See debug .
```

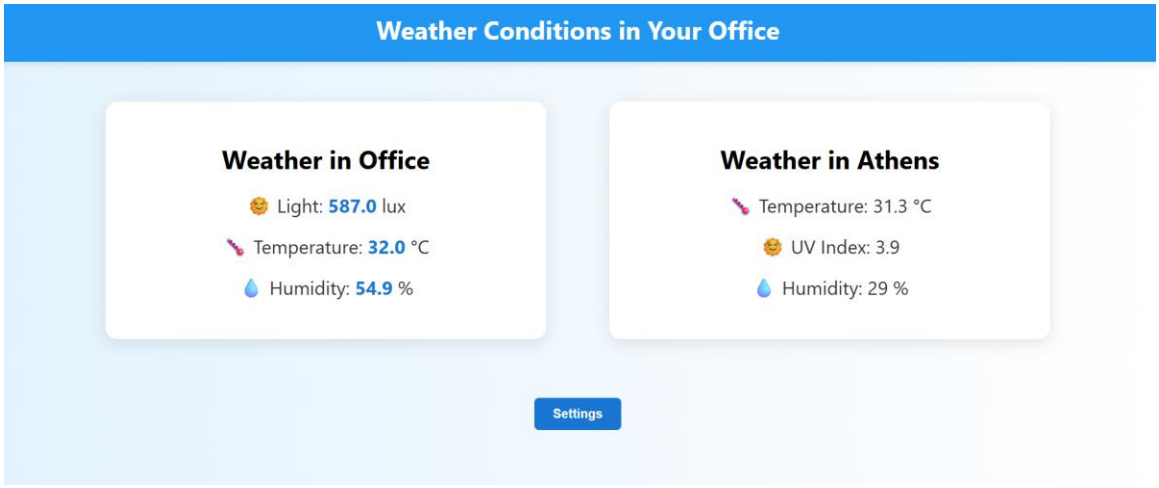
Φόρτωση δεδομένων αισθητήρων και καιρού , εμφάνιση σελίδας index.html με τα δεδομένα και τρέξιμο του Flask Server.

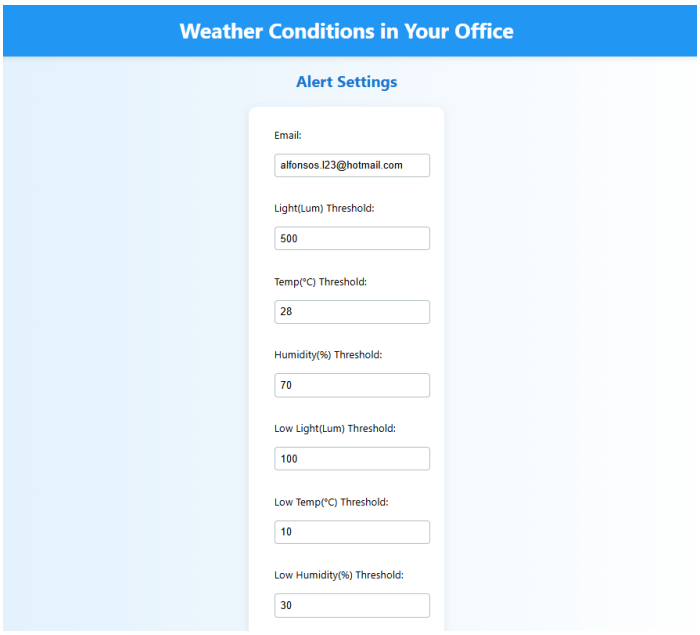
```python
def send_alert_email(subject, body):
    msg = EmailMessage()
    msg['Subject'] = subject
    msg['From'] = EMAIL_ADDRESS
    msg['To'] = user_settings["email"]
    msg.set_content(body)

    try:
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
            smtp.send_message(msg)
    except Exception as e:
        print(f"Failed to send email: {e}")
```

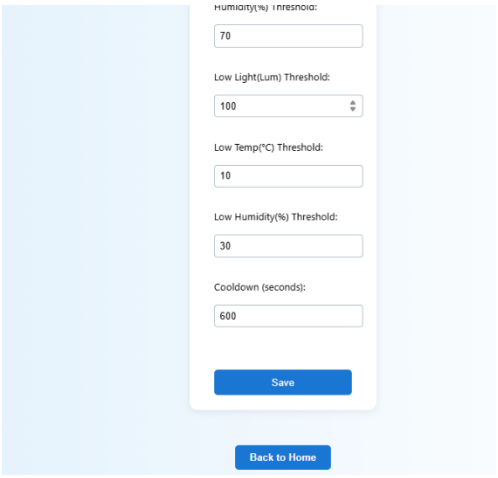
Το παραπάνω screenshot αφορά το αρχείο send email και ρυθμίζει την αποστολή των ειδοποιήσεων μέσω της βιβλιοθήκης **smtplib** και της **email.message.**

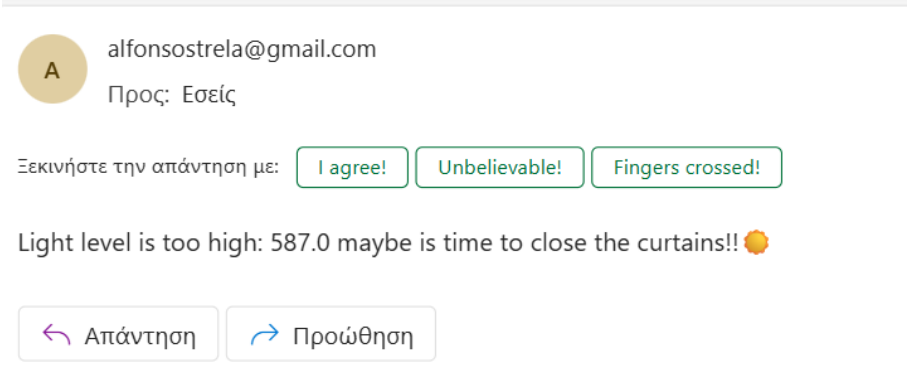
UI ΕΦΑΡΜΟΓΗΣ

Παρακάτω παρατίθεται screenshots από την εκτέλεση της εφαρμογής:









Στόχοι και οφέλη

Κύριοι στόχοι της εφαρμογής είναι ο έλεγχος των συνθηκών καιρού στον εσωτερικό χώρο , η έγκαιρη και η συνεχείς ενημέρωση του χρήστη και μία εύκολη και κατανοητή προς τον χρήστη εφαρμογή.

Μερικά πλεονεκτήματα της εφαρμογής είναι:

- Η ενημέρωση του χρήστη μόνο όταν είναι απαραίτητη χωρίς να γίνεται κουραστική.
- Η απλή ενσωμάτωση της συσκευής στον χώρο.
- Η εύκολή σύνδεση.
- Η χαμηλή κατανάλωση ενέργειας

Προβλήματα και αδυναμίες

Ο τρέχων σχεδιασμός της συσκευής δεν είναι συμπαγής και βρίσκεται σε πρώιμο στάδιο ανάπτυξης. Επιπλέον έχει περιορισμένες λειτουργείες καθώς μπορεί να λάβει δεδομένα μόνο για τρία φυσικά φαινόμενα. Δεν μπορεί να λαμβάνει συνεχόμενα δεδομένα λόγω του περιορισμού που θέτει η έκδοση του ThingSpeak. Τέλος δεν έχει άμεση συνδεσιμότητα με άλλες συσκευές του χρήστη.

Πιθανή εξέλιξη

Η πιθανή εξέλιξη της συγκεκριμένης εφαρμογής θα ήταν:

- Η προσθήκη παραπάνω αισθητήρων και πιο αξιόπιστων ώστε να γίνεται η συλλογή των δεδομένων με καλύτερο τρόπο.

- Η ανάπτυξη κάποιου Android / IOS app ώστε η εμπειρία του χρήστη να είναι ακόμα πιο ευέλικτη.

- Πιθανή λήψη τοποθεσίας χρήστη και κατά επέκταση πιο εύστοχη προσέγγιση συνθηκών περιοχής.
- Απευθείας διασύνδεση με ηλεκτρικές συσκευές με σκοπό την αυτόματη ρύθμιση και την αυτονομία της εφαρμογής.

----------ΤΕΛΟΣ----------