IoT Project

Διαδίκτυο των Αντικειμένων (Εργαστήριο) - Ιωάννης Καμπεράκης - 71347254

Πίνακας περιεχομένων

---

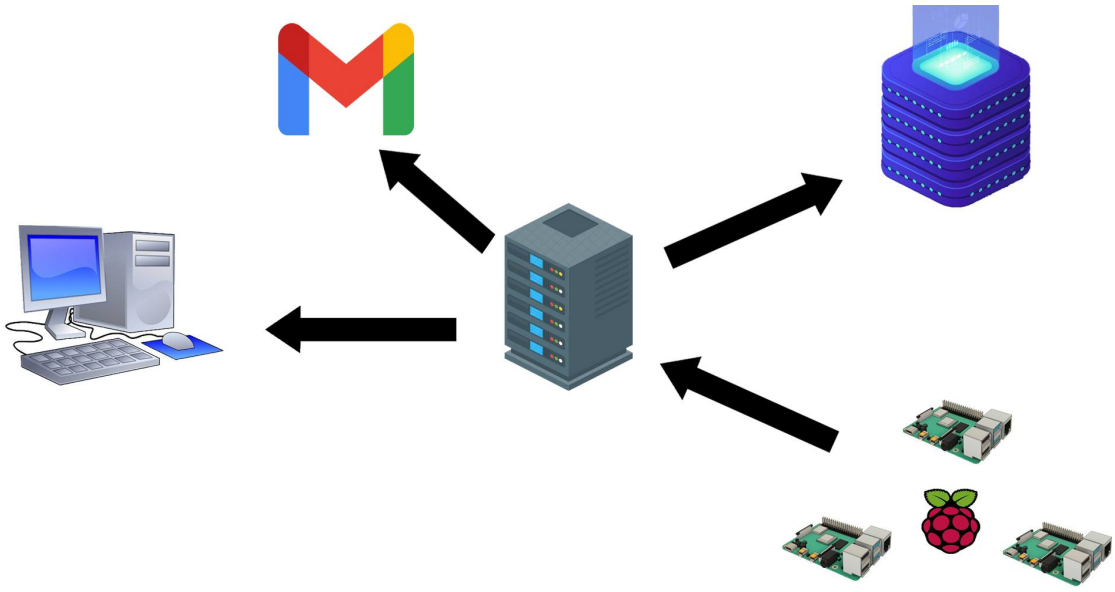
1. Εισαγωγή

Τι περιέχει το Project;

Σε αυτό το Project αναπτύσσεται λογισμικό το οποίο προσομοιώνει την λειτουργία των Raspberry Pis. Ουσιαστικά παράγονται τυχαίες τιμές για την υγρασία του αέρα, του εδάφους, για την ένταση του αέρα και την ατμοσφαιρική θερμοκρασία και έπειτα στέλνονται στον server ανά 30 δευτερόλεπτα όσες τιμές έχουν ποσοστιαία διαφορά μεγαλύτερη του 10%.

Αφού ο server δεχθεί τα δεδομένα επεξεργάζεται την θερμοκρασία και την υγρασία του αέρα. Αν διαπιστωθεί ότι η θερμοκρασία αυξηθεί κατά 40% ή η υγρασία αέρα μειωθεί κατά 50%, αποστέλλεται μήνυμα στο Gmail. Έπειτα τα δεδομένα αποθηκεύονται στην βάση δεδομένων και στέλνωνται στον client ώστε να παρουσιαστούν σε γράφημα.

> *Σχηματική αναπαράσταση του Project*



Για την καλύτερη κατανόηση των ζητουμένων είναι σημαντική η σχηματική αναπαράσταση του Project. Κάτω δεξιά φαίνονται τα Raspberry Pi τα οποία στέλνουν δεδομένα στον server. Ο server στέλνει τα δεδομένα στην database για να αποθηκευθούν και άμα υπάρξει άνοδος στην θερμοκρασία ή μείωση στην υγρασία αέρα τότε στέλνει μήνυμα στο Gmail για να ειδοποιήσει. Τέλος, τα δεδομένα στέλνονται στον client για να αναπαρασταθούν σε γράφημα.

---

2. Οδηγός εγκατάστασης λογισμικών

2.1. Install Python3

Αρχικά πρέπει να εγκατασταθεί η **Python** μιας και στο πρώτο βήμα το πρόγραμμα δημιουργείται με την χρήση της.

```
$ sudo apt-get update
$ sudo apt-get install python3.8 python3-pip
```

2.2. Install latest version of npm and Node.js

Εάν το πρόγραμμα **npm** δεν είναι εγκατεστημένο τότε δεν είναι εφικτό να κατεβάσουμε τα modules στον server (πχ Node.js), όμως η μη ανανέωση του μπορεί να αποτελέσει μεγάλο πρόβλημα καθώς δεν υπάρχει συμβατότητα με τις νέες εκδόσεις διαφόρων προγραμμάτων και τα νέα modules.

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
$ source ~/.profile
$ nvm install node
```

2.3. Install requests

Η **requests** είναι μία βιβλιοθήκη που παρέχει η Python για την αποστολή και ανταλλαγή HTTP

μηνυμάτων.

```
$ python -m pip install requests
```

2.4. Install JSON

Το **JSON** είναι ένα format για την αποθήκευση και ανταλλαγή δεδομένων.

```
$ npm install -g json
```

2.5. Install Express.js

Το **Express.js** είναι ένα framework για την Node.js που χρησιμοποιείται για την δημιουργία web applications και API.

```
$ npm install express --save
```

2.6. Install NeDB

Η **NeDB** είναι ένα ενσωματωμένο έγγραφο DBMS γραμμένο σε JavaScript. Το NeDB είναι χρήσιμο για την αποθήκευση μικρών ποσοτήτων δεδομένων στη μνήμη. Όταν η ποσότητα των δεδομένων υπερβαίνει τα όρια του τι μπορεί να κρατήσει αποτελεσματικά το NeDB, η μετάβαση σε MongoDB προορίζεται να είναι απλή επειδή χρησιμοποιεί το ίδιο API.

```
$ npm install nedb --save
```

2.7. Install dotenv

Με την **dotenv** μπορούμε να αποκρύψουμε προσωπικά δεδομένα που δεν θέλουμε να φανούν στον κώδικα όπως τα στοιχεία σύνδεσης σε μία βάση δεδομένων. Έτσι η σύνδεση γίνεται με μεγαλύτερη ασφάλεια και ο κώδικας μπορεί να δημοσιευθεί.

```
$ npm install dotenv --save
```

2.8. Install nodemailer

Ο **nodemailer** είναι ένα module της Node.js που επιτρέπει την εύκολη αποστολή email.

```
$ npm install nodemailer --save
```

2.9. Install Socket.IO for server

Το **Socket.IO** είναι μια βιβλιοθήκη JavaScript για εφαρμογές ιστού σε πραγματικό χρόνο που επιτρέπει αμφίδρομη επικοινωνία μεταξύ client και server.

```
$ npm install socket.io
```

2.10. Install Socket.IO for client

Πρέπει να εγκατασταθούν οι βιβλιοθήκες για τον client και για τον server για να γίνει σωστή εκτέλεση.

```
$ npm install socket.io-client -- save
```

---

3. Περιγραφή του κώδικα

Σε αυτό το σημείο θα γίνει περιγραφεί του κώδικα που αναπτύχθηκε. Θα αναφερθούν οι λύσεις που δόθηκαν και τα προβλήματα που αντιμετωπίστηκαν. Ξεκινάμε με το πρώτο ζητούμενο που είναι η υλοποίηση λογισμικού που συλλέγει περιβαλλοντικά δεδομένα.

3.1. Υλοποίηση λογισμικού συσκευής IoT για συλλογή περιβαλλοντικών δεδομένων

Για το συγκεκριμένο ερώτημα αναπτύχθηκε αλγόριθμος στην Python που παράγει ψευδοτυχαίες τιμές για την θερμοκρασία, την υγρασία αέρα, την υγρασία νερού και τα μποφόρ. Στην πρώτη επανάληψη δεν γίνεται υπολογισμός της ποσοστιαίας διαφοράς επειδή δεν υπάρχει προηγούμενη μέτρηση (flag = 0). Γι' αυτό χρησιμοποιείται το flag. Αναγκάζει το πρόγραμμα να κάνει υπολογισμούς από την δεύτερη επανάληψη και μετά (όταν δηλαδή το flag γίνει 1).

Στην αρχή του προγράμματος αρχικοποιούνται οι λίστες και οι μεταβλητές που θα χρησμοποιηθούν στο πρόγραμμα, μέσα στην while παράγονται οι μετρήσεις και εκτυπώνονται. Σημαντικό σημείο είναι η γραμμή 28 που φαίνεται στην πρώτη εικόνα που ακολουθεί, όπου το flag αποτρέπει το πρόγραμμα να εκτελεστεί την πρώτη φορά και το σημείο όπου γίνεται το flag = 1 (γραμμή 100, δεύτερη εικόνα).

Αρχικοποίηση τιμών, παραγωγή μετρήσεων και έλεγχος του flag

```
 1 import requests, json, random
 2 from time import sleep
 3
 4 url = 'http://localhost:3000'
 5 old = [0, 0, 0, 0] #list to hold the previous values
 6 new = [0, 0, 0, 0] #list to hold the new values
 7 names = ["temperature", "air humidity", "ground humidity", "beaufort"]
 8 flag = 0
 9 diff = 0
10 timer = 0
11
12 while True:
13   temp = random.randint(15,45) #Temperatures during summer in Greece
14   ahum = random.randint(0,100) #Air humidity
15   ghum = random.randint(0,100) #Ground humidity
16   wind = random.randint(0,17) #Beaufort (Bofor)
17
18   print("Temperature:",temp)
19   print("Air humidity:",ahum)
20   print("Ground humidity:",ghum)
21   print("Beaufort:",wind,"\n")
22
23   new[0]=temp #assigning to list new the new values
24   new[1]=ahum
25   new[2]=ghum
26   new[3]=wind
27
28   if flag == 1: #we don't want to do calculations with zeros
29     print("New temperature: ",new[0], ", Old temperature: ", old[0])
```

Το flag γίνεται 1 και πλέον υπολογίζεται η ποσοστιαία διαφορά των μετρήσεων

```
91   sleep(30) #produce data every 30 seconds
92   timer+=1
93
94
95   old[0]=temp #assigning to list old the old values
96   old[1]=ahum
97   old[2]=ghum
98   old[3]=wind
99
100  flag = 1
```

Αφού υπάρχουν παλιές και νέες τιμές πλέον μπορεί να γίνει η επεξεργασία. Αρχικά ελέγχεται αν έχουν περάσει 5 λεπτά απο την τελευταία φορά που στάλθηκαν δεδομένα στον server, αν δεν ισχύει εκτελείται μία for για να γίνουν οι υπολογισμοί και στις 4 τιμές που προέκυψαν. Μέσα στην for ελέγχονται οι περιπτώσεις που ο αριθμητής είναι 0 ή ο παρονομαστής και γίνεται κατάλληλος χειρισμός. Αφού υπολογιστεί η ποσοστιαία διαφορά ελέγχεται αν είναι μεγαλύτερη του 10%, αν είναι τότε στέλνονται οι τιμές στον server μέσω της εντολής requests.post. Αυτή η διαδικασία επαναλαμβάνεται ανά 30 δευτερόλεπτα.

Ελέγχεται αν έχουν περάσει 5 λεπτά και αν υπάρχουν μηδενικά ψηφία

```
28  if flag == 1: #we don't want to do calculations with zeros
29    print("New temperature: ",new[0], ", Old temperature: ", old[0])
30    print("New air humidity: ",new[1], ", Old air humidity: ", old[1])
31    print("New ground humidity: ",new[2], ", Old ground humidity: ", old[2])
32    print("New beaufort: ",new[3], ", Old beaufort: ", old[3],"\n")
33
34    if timer==10: #if 5 minutes have passed then send the new values to server (10 x 30secs = 300 secs = 5 mins)
35      print("Send to server: 5 mins passed!")#send all the new data to server
36
37      data = {0:new[i]} #sending the new value of temperature
38      r = requests.post(url, data) #sending a json object to the specified url
39
40      data = {1:new[i]} #sending the new value of air humidity
41      r = requests.post(url, data) #sending a json object to the specified url
42
43      data = {2:new[i]} #sending the new value of ground humidity
44      r = requests.post(url, data) #sending a json object to the specified url
45
46      data = {3:new[i]} #sending the new value of beaufort
47      r = requests.post(url, data) #sending a json object to the specified url
48
49    else:
50      for i in range(4):
51        if (new[i] == 0) and (old[i] == 0): #if the previous value and the new one are 0 then continue
52          continue
53        elif old[i] == 0: #we can't devide a number with zero
54          if new[i] > 2: #check if the new value is above 2 (10%)
55            diff = new[i]
56        elif new[i] == 0: #if the new value is 0 then
57          if old[i] > 2: #check if the old value is above 2 (10%)
58            diff = old[i]
59        else:
60          diff = round((abs((new[i]-old[i]))/old[i])*100,2) #calculating percentage difference
61          print("Percentage difference of",names[i],":",diff)
62
```

Αν το ποσοστο που προέκυψε είναι μεγαλύτερο του 10%, στείλε τις τιμές στον server

```
63          if diff > 10: #if the new and old values differ by 10 then send the new value to server
64            timer = 0
65            print('Sending to server (10%)...')
66
67            if(i == 0):
68
69              data = {0:old[i]} #sending the old value
70              r = requests.post(url, data)
71              data = {0:new[i]} #sending the new value
72
73            elif(i == 1):
74
75              data = {1:old[i]} #sending the old value
76              r = requests.post(url, data)
77              data = {1:new[i]} #sending the new value
78
79            elif(i == 2):
80
81              data = {2:new[i]} #sending the old value
82
83            elif(i == 3):
84
85              data = {3:new[i]} #sending the old value
86
87            r = requests.post(url, data) #sending a json object to the specified url
88
89      print('\n')
90
91    sleep(30) #produce data every 30 seconds
92    timer+=1
93
94
95    old[0]=temp #assigning to list old the old values
96    old[1]=ahum
97    old[2]=ghum
98    old[3]=wind
```

3.2. Αποστολή δεδομένων σε απομακρυσμένο server

Για να λάβει ο server τα δεδομένα πρέπει να πάρει με post τα requested δεδομένα. Στην γραμμή 43 γίνεται αυτή η διαδικασία. Στις επόμενες εικόνες φαίνονται οι βασικές εντολές που χρειάζονται για την δημιουργία ενός REST API server.

Δημιουργία server

```
1 const app = require('express')(); //loading express module
2 const http = require('http').createServer(app);  //creating server
```

```
43 app.post('/', (request, response) => { //server sends data to Gmail, database and the client
```

```
180 http.listen(3000, () => { console.log('Server is listening...'); }); //listening at port 3000
```

3.3. Υλοποίηση υπηρεσίας επιτήρησης πραγματικού χρόνου

Για το συγκεκριμένο ερώτημα ήταν αναγκαία η προσθήκη των modules που φαίνονται στην παρακάτω εικόνα.

```
3 const parser = require('body-parser'); //loading body-parser module
4 const nodemailer = require('nodemailer'); //loading nodemailer module
```

Επίσης παράχθηκε και το αρχείο package.json το οποίο περιέχει τα dependencies που χρειάστηκαν.

package.json

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "Raspberry Pi data",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "example",
    "data",
    "raspberry",
    "pi"
  ],
  "author": "John Kaberakis",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "nedb": "^1.8.0",
    "nodemailer": "^6.4.17",
    "socket.io-client": "^3.1.0"
  }
}
```

Με τον body-parser λαμβάνουμε τα δεδομένα του python προγράμματος. Γι' αυτό και ορίζουμε το body-parser να δέχεται δεδομένα οποιουδήποτε τύπου. Το module nodemailer είναι αναγκαίο για την αποστολή email στο gmail. Στις επόμενες εικόνες φαίνονται οι αρχικοποίηση των μεταβλητών και των λιστών που χρησιμοποιούνται για την επεξεργασία των τιμών της θερμοκρασίας και της υγρασίας, η συνάρτηση που πραγματοποιεί του υπολογισμούς, ο προσδιορισμός της υπηρεσίας που θα σταλθεί email και η επεξεργασία των τιμών της θερμοκρασίας και της υγρασίας αέρος.

Αρχικοποίηση μεταβλητών και λιστών

```
10 var temp_old_value = 0; //inserting the previous value of temperature
11 var temp_flag = 0; //flag to check if we got the previous value of temperature
12 var temp_result = 0; //inserting the percentage difference of temperature
13
14 var hum_old_value = 0; //inserting the previous value of air humidity
15 var hum_flag = 0; //flag to check if we got the previous value of air humidity
16 var hum_result = 0; //inserting the percentage difference of air humidity
```

Συνάρτηση υπολογισμού, ορισμός υπηρεσίας και επεξεργασία των θερμοκρασιών

```
function calculate(old, neww){ //calculating the percentage difference between the old and the new measure
  return (neww-old)/old*100;
}

app.post('/', (request, response) => { //server sends data to Gmail, database and the client
  let transporter = nodemailer.createTransport({//to notify the client we need specify the name of the email service we will use
    service: 'gmail',
    auth: { //the file .env has the email and password and the file .gitignore makes sure the file .env never gets pushed
      user: process.env.EMAIL,
      pass: process.env.PASSWORD
    }
  });

  if (request.body[0]){
    if(temp_flag == 0){ //if we don't have the previous value then..
      temp_old_value = request.body[0]; //..insert it to temp_old_value
      temp_flag = 1;  //then set temp_flag to 1
    } else if(temp_flag == 1){ //now that we have the previous value we will get the new value..
      console.log('Old value:',temp_old_value,', New value:',request.body[0]);
      temp_result = calculate(temp_old_value, request.body[0]); //..to find the percentage defference between the new and old value

      console.log('Temperature percentage difference:',temp_result.toFixed(1) + "%!\n");
```

Επεξεργασία υγρασίας αέρος

```
  } else if (request.body[1]) {
    if(hum_flag == 0){  //if we don't have the previous value then..
      hum_old_value = request.body[1]; //..insert it to temp_old_value
      hum_flag = 1; //then set temp_flag to 1
    } else if(hum_flag == 1){ //now that we have the previous value we will get the new value..
      console.log('Old value:',hum_old_value,', New value:',request.body[1]);
      hum_result = calculate(hum_old_value, request.body[1]);//..to find the percentage defference between the new and old value

      console.log('Air humidity percentage difference:',hum_result.toFixed(1) + '%!');
```

Αφού υπολογιστούν οι ποσοστιαίες διαφορές ελέγχεται αν η θερμοκρασία ανέβηκε κατά 40% και αν η υγρασία μειώθηκε κατα 50%. Αν συμβαίνει ένα από τα δύο τότε στέλνεται μήνυμα στο email που φαίνεται για να ειδοποιηθεί ο διαχειριστής.

Έλεγχος θερμοκρασίας, αποστολή email

```
      if(temp_result > 40){ //if temperature increased by 50% then send an email to kampe.test@gmail.com to warn him

        let mailOptions = {
          from: process.env.EMAIL,
          to: 'kampe.test@gmail.com',
          subject: 'ATTENTION',
          text: 'Temperature increased alot!'
        };

        transporter.sendMail(mailOptions, function(err, data) { //sending the email
          if (err){
            console.log('Error Occurs');
          } else {
            console.log('Email sent (for temperature).');
          }
        });
      }
    }
```

Έλεγχος υγρασίας αέρα, αποστολή email

```
      if(hum_result < 50){ //if humidity decreased by 50% then send an email to kampe.test@gmail.com to warn him
        let mailOptions = {
          from: process.env.EMAIL,
          to: 'kampe.test@gmail.com',
          subject: 'ATTENTION',
          text: 'Humidity decreased alot!'
        };

        transporter.sendMail(mailOptions, function(err, data) { //sending the email
          if (err){
            console.log('Error Occurs');
          } else {
            console.log('Email sent (for air humidity).');
          }
        });
      }

    }
```

Στο συγκεκριμένο ερώτημα παρουσιάστηκαν ορισμένα προβλήματα. Αρχικά ήθελα να υλοποιήσω το ερώτημα στέλνοντας SMS μηνύματα όμως οι υπηρεσίες που έβρισκα (Twilio, Sinch, way2sms, infobip) δεν ήταν ελληνικές και δεν παρείχαν δωρεάν υπηρεσίες για άλλες χώρες πέρα απο την δική τους. ΒΡήκα και μία Ελληνική όμως έπρεπε να πληρώσω κάποιο ποσό. Το Gmail δεν με άφηνε να στείλω μηνύματα για

> λόγους προστασίας όμως κατάφερα να στείλω ενεργοποιώντας μία λειτουργία που δίνει μεγαλύτερη ελευθερία στα μηνύματα που δέχεται το email μου.

3.4. Αποθήκευση των δεδομένων σε βάση δεδομένων

Για την αποθήκευση των δεδομένων δημιουργήθηκε μία NeDB. Επιλέχθηκε η συγκεκριμένη βάση δεδομένων επειδή δεν υπάρχουν υψηλές απαιτήσεις μνήμης από την εφαρμογή που υλοποιείται. Στην πρώτη εικόνα φαίνονται τα modules που προστέθηκαν για το συγκεκριμένο ερώτημα. Με την socket.io θα στέλνονται τα δεδομένα στον client και θα απεικονίζονται στα γραφήματα. Η dotenv παρέχει προστασία των δεδομένων, σε αυτή την περίπτωση προστατεύονται τα στοιχεία σύνδεσεις για την βάση δεδομένων με την χρήση του αρχείου .env και .gitignore.

Νέα modules

```
5 const io = require('socket.io')(http); //loading socket.io module using HTTP transport
6 const Data = require('nedb'); //loading nedb module
7 require('dotenv').config(); //for database security
```

Στην συνέχεια φαίνεται η αρχικοποίηση των μεταβλητών και των λιστών που χρησιμοποιούνται για την αποστολή των δεδομένων στον client.

Αρχικοποίηση μεταβλητών και λιστών

```
18 var temp_list = []; //we need the lists to send the data to client
19 var air_list = [];
20 var ground_list = [];
21 var beauf_list = [];
22
23 var temp_count = 0; //our lists will have 10 values so we need counters to insert and update the values
24 var air_count = 0;
25 var ground_count = 0;
26 var beauf_count = 0;
```

Έπειτα δημιουργείται η βάση δεδομένων που θα χρησιμοποιηθεί με όνομα rasp_measurements.db. Στις γραμμές 34 έως 36 στέλνεται στον client η html σελίδα που θα βλέπει τα αποτελέσματα.

Δημιουργία βάσης δεδομένων

```
30 const database = new Data('rasp_measurements.db'); //creating database with name rasp_measurements.db
31 database.loadDatabase(); //loading database or creating it if it doesn't exist
32
33
34 app.get('/', (req,res) => { //sending the page.html as a result of clients request
35   res.sendFile(__dirname + '/page.html');
36 });
```

Η ιστοσελίδα που θα βλέπει ο χρήστης αποτελείται από τον παρακάτω κώδικα.

page.html

```html
<!DOCTYPE HTML>
<html>
  <head>
    <title>Raspberry Pi Measures</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <script src="/socket.io/socket.io.js"></script>
  </head>

  <body>
    <div id="myDiv"></div> <!-- Presenting the graph at localhost:3000/ -->
    <script>
      const socket = io(); //connecting with server

      var data = []; //initializing list data, we will put the
                     //different traces (trace0, trace1, trace2, trace3) in order to plot them

      var trace0 = {}; //trace of temperature
      var trace1 = {}; //trace of air humidity
      var trace2 = {}; //trace of ground humidity
      var trace3 = {};  //trace of beaufort

      socket.on('graph', (value) => { //on event with name 'graph' get the values from the server
        if(value[10] == 0){ //if the id = 0 then it's temperature values
          trace0 = { //update the trace of temperature
            x: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
            y: [value[0],value[1],value[2],value[3],value[4],value[5],value[6],value[7],value[8],value[9]],
            type: 'scatter',
            mode: 'lines',
            name: 'Temperature'
          };
        } else if(value[10] == 1){ //if the id = 1 then it's air humidity values
          trace1 = { //update the trace of air humidity
            x: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
            y: [value[0],value[1],value[2],value[3],value[4],value[5],value[6],value[7],value[8],value[9]],
            type: 'scatter',
            mode: 'lines',
            name: 'Air humidity'
          };
        }else if(value[10] == 2){ //if the id = 2 then it's ground humidity values
          trace2 = { //update the trace of ground humidity
            x: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
            y: [value[0],value[1],value[2],value[3],value[4],value[5],value[6],value[7],value[8],value[9]],
            type: 'scatter',
            mode: 'lines',
            name: 'Ground humidity'
          };
        }else if(value[10] == 3){ //if the id = 3 then it's beaufort values
          trace3 = { //update the trace of beaufort
            x: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
            y: [value[0],value[1],value[2],value[3],value[4],value[5],value[6],value[7],value[8],value[9]],
            type: 'scatter',
            mode: 'lines',
            name: 'Beaufort'
          };
        }
        data = [trace0, trace1, trace2, trace3]; //inserting in list data the traces
        console.log(data); //printing the traces for debugging reasons
        Plotly.newPlot('myDiv', data); //plotting the traces
      });
      </script>
    </body>
</html>
```

Όταν η ποσοστιαία διαφορά των μετρήσεων είναι μεγαλύτερη του 10 τότε εκχωρούνται οι τιμές σε μία λίστα όπου στο τέλος της μπαίνει ένα συγκεκριμένο id ανάλογα την μέτρηση (πχ για temp → 0). Μόλις αυτές οι λίστες γεμίσουν τότε στέλνονται με custon event (emit) στον client ώστε οι τιμές που περιέχουν να απεικονιστούν στο γράφημα. Αυτη η διαδικασία γίνεται για όλες τις μετρήσεις. Το id βοηθάει στην σωστή απεικόνιση των δεδομένων.

Είσοδος των τιμών της θερμοκρασίας στην DB και αποστολή των 10 πρώτων τιμών της στον client

```
60      database.insert({measure: 'temperature',value:parseInt(request.body[0])}); //inserting the new values to database
61      temp_flag = 0; //making temp_flag = 0 to do the same things for the next values
62
63      temp_list[temp_count] = parseInt(request.body[0]); //inserting the new value to list temp_list at temp_count position
64      console.log('List of temperatures:',temp_list); //for better understanding
65      temp_count +=1; //increasing the temp_count counter
66
67      if(temp_count == 10){ //if we have 10 values in the list then insert at the 11th position the correct id (0 for temperature)
68        temp_list[temp_count] = 0; //inserting the correct id
69        temp_count = 0; //start from the begining and update the values
70
71        io.on('connection', socket => { //then send the list to client
72          console.log('We have a connection');
73          socket.emit('graph', temp_list);
74        });
75      }
```

Είσοδος των τιμών της υγραίας αέρος στην DB και αποστολή των 10 πρώτων τιμών της στον client

```
104     database.insert({measure: 'air humidity',value:parseInt(request.body[1])}); //inserting the new values to database
105     hum_flag = 0; //making hum_flag = 0 to do the same things for the next values
106
107     air_list[air_count] = parseInt(request.body[1]); //inserting the new value to list air_list at air_count position
108     console.log('List of air humidity:',air_list); //for better understanding
109     air_count +=1; //increasing the air_count counter
110
111     if(air_count == 10){ //if we have 10 values in the list then insert at the 11th position the correct id (1 for air humidity)
112       air_list[air_count] = 1; //inserting the correct id
113       air_count = 0; //start from the begining and update the values
114
115       io.on('connection', socket => { //then send the list to client
116         console.log('We have a connection');
117         socket.emit('graph', air_list);
118       });
119     }
```

Είσοδος των τιμών της υγραίας εδάφους και των μποφόρ στην DB και αποστολή των 10 πρώτων τιμών τους στον client

```
139   } else if (request.body[2]) {
140     ground_list[ground_count] = parseInt(request.body[2]); //inserting the new value to list ground_list at ground_count position
141     console.log('List of air humidity:',ground_list); //for better understanding
142     ground_count +=1; //increasing the air_count counter
143
144     if(ground_count == 10){ //if we have 10 values in the list then insert at the 11th position the correct id (2 for ground humidity)
145       ground_list[ground_count] = 2; //inserting the correct id
146       ground_count = 0; //start from the begining and update the values
147
148       io.on('connection', socket => { //then send the list to client
149         console.log('We have a connection');
150         socket.emit('graph', ground_list);
151       });
152     }
153     console.log('Ground humidity value: ',request.body[2]);
154     database.insert({measure: 'ground humidity',value:parseInt(request.body[2])}); //inserting the new values to database
155
156   } else if (request.body[3]) {
157     beauf_list[beauf_count] = parseInt(request.body[3]); //inserting the new value to list beauf_list at beauf_count position
158     console.log('List of beaufort:',beauf_list); //for better understanding
159     beauf_count +=1; //increasing the air_count counter
160
161     if(beauf_count == 10){ //if we have 10 values in the list then insert at the 11th position the correct id (3 for beaufort)
162       beauf_list[beauf_count] = 3; //inserting the correct id
163       beauf_count = 0; //start from the begining and update the values
164
165       io.on('connection', socket => { //then send the list to client
166         console.log('We have a connection');
167         socket.emit('graph', beauf_list);
168       });
169     }
170     console.log('Ground beaufort value: ',request.body[3]);
171     database.insert({measure: 'beaufort',value:parseInt(request.body[3])}); //inserting the new values to database
172
173   }
174
175   response.end(); //ending the event
```

Last updated 2021-01-24 23:11:07 +0200