ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

INTERNET OF THINGS

(Διαδίκτυο των Πραγμάτων)

ΠΑΥΛΟΣ Δ. ΑΡΓΥΡΙΟΥ

Επιβλέποντες: Έλληνας Ιωάννης

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2019

Περιεχόμενα

Εισαγωγή

Το διαδίκτυο άλλαξε ολοκληρωτικά τον κόσμο καθώς πλέον αποτελεί αναπόσπαστο κομμάτι της ζωής του ανθρώπου. Η ιδέα 'Internet of Things'' (IoT) βασίστηκε στην σημαντικότητα του διαδικτύου στον άνθρωπο, σε συνδυασμό με την ανάγκη του για μια πιο άνετη κι ευκολότερη καθημερινότητα. Υπάρχουν δισεκατομμύρια συνδεδεμένες συσκευές στο διαδίκτυο που ανταλλάσσουν δεδομένα και πληροφορίες. Οι συσκευές αυτές ολοένα και αυξάνονται επηρεάζοντας την ανθρωπότητα σε μεγαλύτερο βαθμό από ότι το ίδιο το διαδίκτυο. Σύμφωνα με εκτίμηση της Cisco ο συνολικός αριθμός των IoT συσκευών θα φτάσει 50 δις το 2020 [1] και 500 δις μέχρι το 2030 [2].

Σκοπός της πτυχιακής εργασίας είναι να κάνει μια εισαγωγή σε αυτό το σχετικά νέο πεδίο του τεχνολογικού κόσμου. Το IoT είναι για πολλούς το επόμενο βήμα εξέλιξης του διαδικτύου όπως το γνωρίζουμε σήμερα. Στα πρώτα δύο κεφάλαια περιγράφεται η έννοια και αναφέρονται κάποιες από τις εφαρμογές του καθώς οι δυνατότητες για δημιουργία νέων εφαρμογών είναι πρακτικά άπειρες.

Το τρίτο κεφάλαιο περιγράφει το πρακτικό κομμάτι της εργασίας και τη χρήση του πρωτόκολλου MQTT. Περισσότερη ανάλυση του τρόπου λειτουργίας του, των βασικών εννοιών για την κατανόηση του, γίνεται στο τέταρτο κεφάλαιο.

Στο πέμπτο κεφάλαιο γίνεται αναφορά και περιγραφή των υλικών που χρησιμοποιήθηκαν για την εργασία, ενώ στο έκτο κεφάλαιο παρουσιάζεται το κύκλωμα με σχηματικό διάγραμμα.

Τα κεφάλαια από το έβδομο έως και το ενδέκατο έχουν αναλυτικές οδηγίες για την εγκατάσταση και ρύθμιση των εφαρμογών που χρησιμοποιήθηκαν για την υλοποίηση του ζητούμενου της εργασίας.

1. Internet of Things

Ο όρος ‘’Internet of Things’’ (διαδίκτυο των πραγμάτων) ξεκίνησε ως ιδέα τη δεκαετία του 70’. Χρησιμοποιήθηκε πρώτη φορά από τον Kevin Ashton το 1999 ως θέμα της ομιλίας του σε μια παρουσίαση για την Procter & Gamble [3]. Τόνισε την αναγκαία προϋπόθεση της ύπαρξης RFID (Radio Frequency Identification) στην αλυσίδα ανεφοδιασμού της εταιρείας ώστε μέσω συλλογής πληροφοριών από υπολογιστές να μειωθούν τα απόβλητα, οι απώλειες και οι δαπάνες.

Αν θέλαμε περιληπτικά να ορίσουμε τον όρο Internet of Things θα λέγαμε πως είναι η διαδικτυακή σύνδεση υπολογιστικών συσκευών που είναι ενσωματωμένες σε καθημερινά αντικείμενα, με σκοπό να επικοινωνούν και να αλληλοεπιδρούν μεταξύ τους, αλλά και να ελέγχονται ή να παρακολουθούνται απομακρυσμένα.

2. Εφαρμογές

Τα τελευταία χρόνια η έννοια είναι σε τεράστιο βαθμό διαδεδομένη και πλέον η εφαρμογή της συναντάται παντού. Σε αρκετά πεδία η εφαρμογή της βρίσκεται ακόμη σε πειραματικό στάδιο και συνεχώς εξελίσσεται. Παρακάτω αναφέρονται μερικοί τομείς στους οποίους ήδη χρησιμοποιείται.

Έξυπνο Σπίτι

Ίσως η πρώτη εφαρμογή που έρχεται κατά νου όταν αναφερόμαστε σε IoT συστήματα μιας και χιλιάδες άνθρωποι μηνιαίως αναζητούν έξυπνες λύσεις για τα σπίτια τους. Παρέχονται δυνατότητες διαχείρισης θερμοστατών, κλιματισμού, εικόνας, ήχου, φωτισμού ακόμη και αυτόματου ταΐσματος κατοικίδιων.

Wearables

Σχεδιασμένες συσκευές ώστε να φοριούνται, τα smartwatches χρησιμοποιούνται ευρέως σε όλο τον κόσμο. Σε συνδυασμό με συνοδευτικές εφαρμογές που εγκαθίστανται σε smartphones, δίνουν τη δυνατότητα λήψης μηνυμάτων, τηλεφωνημάτων, καθώς και λήψης πληροφοριών σε σχέση με την υγεία και την φυσική κατάσταση.

Βιομηχανική Παραγωγή

Η απομακρυσμένη παρακολούθηση συσκευών και μηχανημάτων σε βιομηχανίες. Μια τεχνολογία που συμβάλει στην βελτίωση της παραγωγικότητας ως προς την ποιότητα και τον χρόνο, την εξοικονόμηση ενέργειας και πρώτης ύλης, αλλά και την αποτροπή δυστυχημάτων σε περιπτώσεις δυσλειτουργίας.

Γεωργία και Κτηνοτροφία

Αρκετοί αγρότες και κτηνοτρόφοι έχουν βασίσει την διαχείριση των καλλιεργειών και των ζώων τους σε έξυπνες συσκευές. Χρησιμοποιούνται εργαλεία ανάλυσης της σύνθεσης του εδάφους και πρόγνωσης καιρού, drones, συσκευές ικανές να ανιχνεύσουν ασθένειες σε μέλη του κοπαδιού και να εντοπίσουν τη θέση τους.

Υγειονομική Περίθαλψη

Υπάρχουν πολλαπλές εφαρμογές στον τομέα της υγείας όπως προσωπικοί αισθητήρες γυμναστικής, εξοπλισμό απομακρυσμένης παρακολούθησης των ασθενών ακόμη και χειρουργικά ρομπότ.

Αυτοκίνητο

Μια τεχνολογία η οποία συνδυάζοντας αισθητήρες, κεραίες και ενσωματωμένο λογισμικό βοηθάει την πλοήγηση και τη στάθμευση. Έχει ακόμη τη δυνατότητα να λαμβάνει αποφάσεις που σχετίζονται με την ταχύτητα, την ακρίβεια και την σταθερότητα.

3. Αντικείμενο Εργασίας

Η παρακάτω εργασία αφορά μια κατασκευή ενός απλού συστήματος μέτρησης θερμοκρασίας και υγρασίας. Οι λήψεις των μετρήσεων γίνονται μέσω αισθητήρα και υπάρχει η δυνατότητα γραφικής απεικόνισης των αποτελεσμάτων μέσω web interface σε συσκευές που βρίσκονται συνδεδεμένες στο ίδιο δίκτυο. Οι μετρήσεις του αισθητήρα λαμβάνονται και αποστέλλονται από ένα ESP32 με τη χρήση του MQTT πρωτόκολλου επικοινωνίας. Τη διαχείριση των δεδομένων από το ESP32 αλλά και προς αυτό, την αναλαμβάνει ένα Raspberry Pi ως MQTT Broker (IoT Server). Μέσα από την εφαρμογή Node-RED οι συνδεδεμένοι χρήστες μπορούν να βλέπουν τις μετρήσεις κι επιπλέον μπορούν να επέμβουν απομακρυσμένα ανάβοντας ή σβήνοντας ένα LED, το οποίο έχει συνδεθεί σε μια από τις εξόδους του ESP32. Στις επόμενες παραγράφους θα αποσαφηνιστούν οι ενδεχομένως άγνωστοι όροι, ώστε να γίνει πλήρως κατανοητό το αντικείμενο με το οποίο ενασχοληθήκαμε.

4. Τι είναι το MQTT και πως λειτουργεί

Το MQTT (Message Queuing Telemetry Transport) είναι πρωτόκολλο επικοινωνίας σχεδιασμένο ώστε να χρησιμοποιεί Publish/Subscribe λειτουργίες για την ανταλλαγή δεδομένων μεταξύ των clients και του server (broker). Η ελάχιστη απαίτηση πόρων, η χρήση μικρού δικτυακού εύρους ζώνης είναι μερικά από τα χαρακτηριστικά που το καθιστούν ιδανικό για τη χρήση του σε εφαρμογές IoT. [4]

Παρακάτω αναλύονται κάποιες βασικές έννοιες προς την κατανόηση του τρόπου λειτουργίας του MQTT.

4.1 Publish/Subscribe

Σε ένα σύστημα publish & subscribe μια συσκευή μπορεί να δημοσιεύσει (publish) ένα μήνυμα σε κάποιο topic, ή μπορεί να είναι εγγεγραμμένο (subscribed) σε ένα συγκεκριμένο topic ώστε να λαμβάνει μηνύματα.

Παραδείγματος χάρη το **DEVICE 1** κάνει publish σε ένα topic, ενώ το **DEVICE 2** είναι subscribed στο ίδιο topic οπότε λαμβάνει μηνύματα.

4.2 Messages

Τα μηνύματα (messages) είναι εντολές ή δεδομένα που ανταλλάζονται μεταξύ των client και του broker. Υπάρχουν οι εξής βασικοί τύποι μηνυμάτων:

Connect: Δημιουργία σύνδεσης με τον Broker

Disconnect: Διακοπή σύνδεσης με τον Broker

Publish: Δημοσίευση δεδομένων σχετικά με κάποιο θέμα μέσα στον Broker

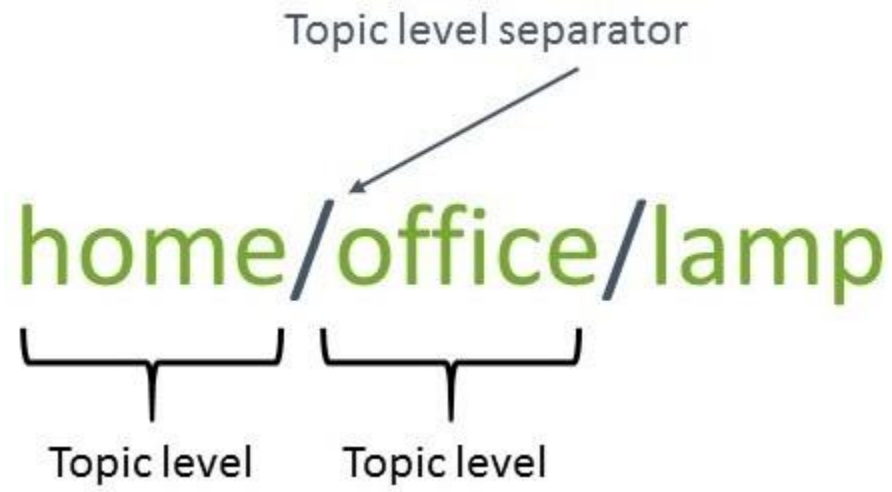
Subscribe: Εγγραφή σε ένα θέμα στον Broker

Unsubscribe: Διαγραφή του θέματος

4.3 Topics

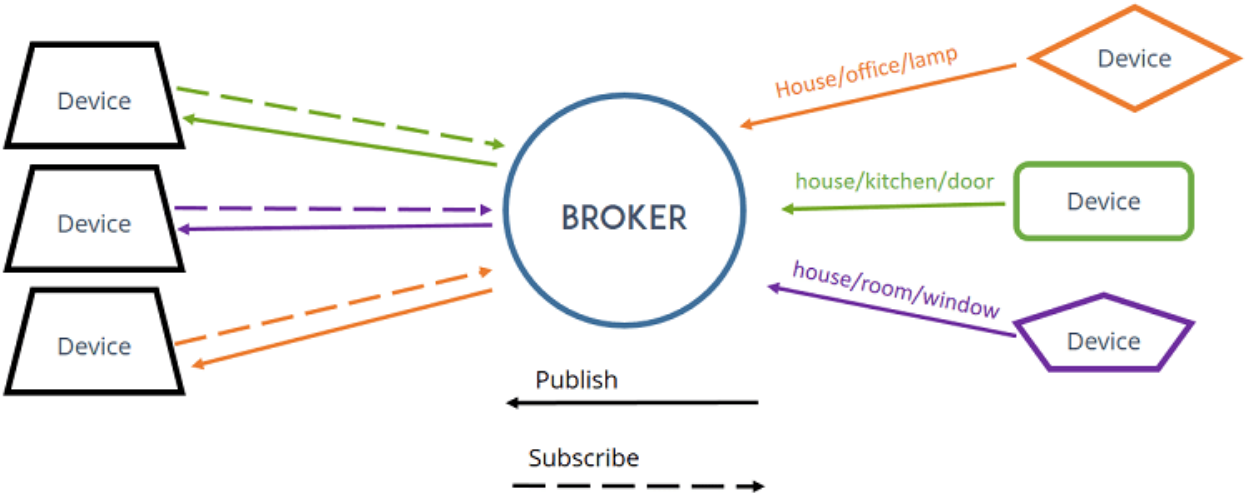
Τα θέματα (topics) είναι χαρακτήρες με κωδικοποίηση UTF-8 που χρησιμοποιεί ο broker ώστε να φιλτράρει τα μηνύματα του κάθε συνδεδεμένου client. Αποτελείται από ένα ή περισσότερα θεματικά επίπεδα τα οποία διαχωρίζονται μεταξύ τους με ένα forward slash (/).

Για παράδειγμα η σύνταξη ενός topic για την λάμπα στο γραφείο του σπιτιού μας θα ήταν ως εξής:

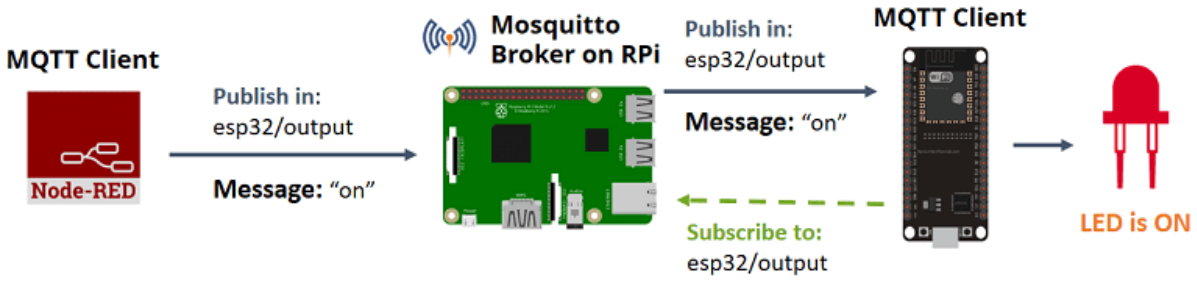


4.4 Broker

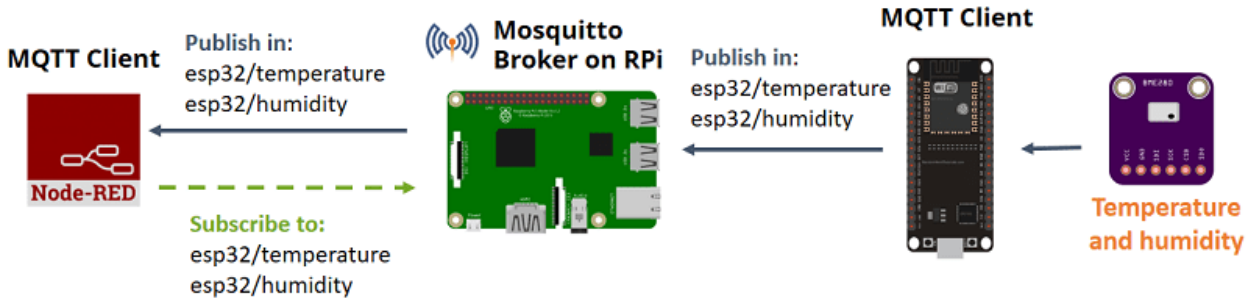
Ένας Broker αποτελεί την καρδιά ενός οποιουδήποτε συστήματος με πρωτόκολλο publish/subscribe. Διαχειρίζεται την λήψη και το φιλτράρισμα μηνυμάτων, αποφασίζει ποιοι θα είναι εγγεγραμμένοι σε κάθε μήνυμα και αποστέλλει το μήνυμα σε αυτούς τους εγγεγραμμένους χρήστες (subscribed clients).



Οι επόμενες εικόνες δείχνουν συνοπτικά τον ρόλο κάθε εξαρτήματος στην εργασία, καθώς και τη χρήση του πρωτόκολλου επικοινωνίας MQTT για τον έλεγχο μιας εξόδου του ESP32 αλλά και για τη δημοσίευση των μετρήσεων του αισθητήρα.

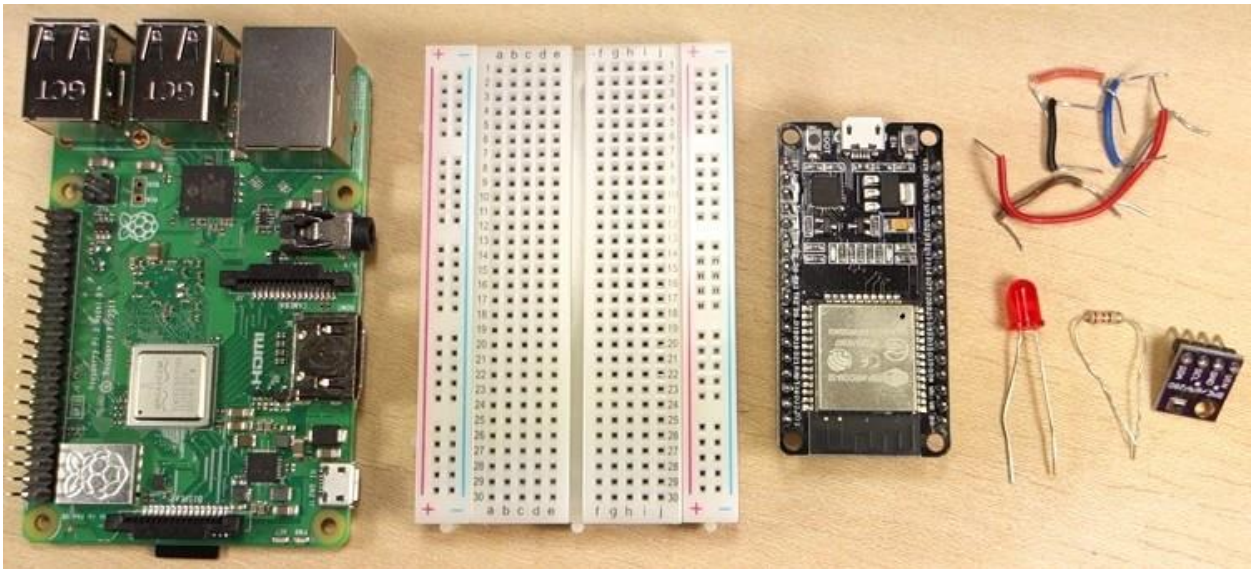


Οι συνδεδεμένοι χρήστες κάνουν αποστολή (publish) μηνυμάτων on ή off από το Node-RED στο topic esp32/output. Στο ίδιο topic είναι εγγεγραμμένο (subscribed) και το ESP32 ώστε λαμβάνοντας τα μηνύματα on ή off αντίστοιχα ανάβει ή σβήνει το LED.



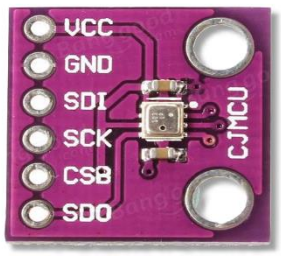
Όπως φαίνεται στην παραπάνω εικόνα το ESP32 κάνει publish τη θερμοκρασία στο topic esp32/temperature και την υγρασία στο topic esp32/humidity. Το Node-RED ως εγγεγραμμένος client στα ίδια topic, λαμβάνει και απεικονίζει τις μετρήσεις.

5. Απαραίτητος Εξοπλισμός



- BME280 Sensor Module
- Raspberry Pi 3 Model B+ (Keyboard & Mouse, microSD, Power supply 5V DC)
- Espressif ESP32 WLAN Dev Kit Board Development Bluetooth Wifi v1 WROOM32
- 1x 5mm LED
- 1x 220 Ohm Resistor
- Breadboard
- Jumper wires

5.1 BME280 Sensor Module



Το BME280 είναι ένας περιβαλλοντικός αισθητήρας ειδικά σχεδιασμένος για εφαρμογές όπου το μέγεθος και η χαμηλή κατανάλωση ισχύος αποτελούν βασικούς περιορισμούς σχεδιασμού. Η μονάδα συνδυάζει ξεχωριστούς αισθητήρες υψηλής ακρίβειας για την μέτρηση θερμοκρασίας, υγρασίας και βαρομετρικής πίεσης και η επικοινωνία επιτυγχάνεται με τη χρήση πρωτόκολλου επικοινωνίας SPI ή I2C.

5.2 Raspberry Pi

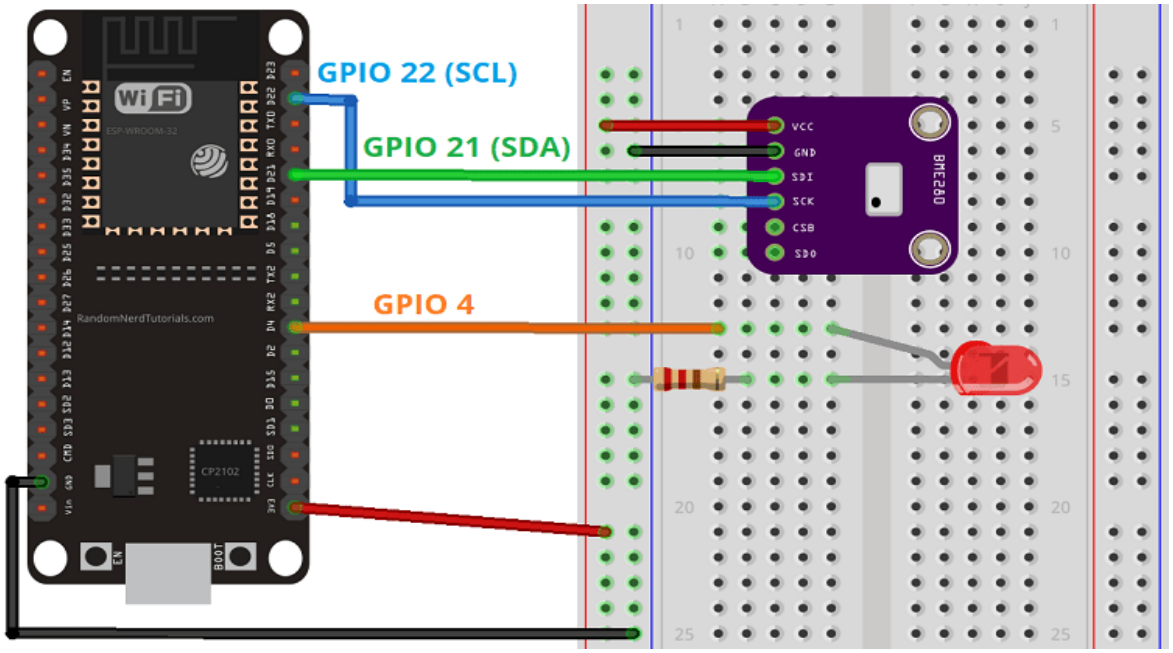
Το Raspberry Pi είναι ένας χαμηλού κόστους υπολογιστής που κατασκευάζεται από την Raspberry Pi Foundation UK. Έχει διαστάσεις πιστωτικής κάρτας με δυνατότητα σύνδεσης σε οθόνη ή τηλεόραση. Διαθέτει επίσης ακροδέκτες GPIO (general purpose input/output) ώστε να παρέχει τη δυνατότητα ελέγχου ηλεκτρονικών εξαρτημάτων. Υποστηρίζει Linux λειτουργικό σύστημα και γλώσσες προγραμματισμού Scratch και Python. Η πρώτη έκδοση βγήκε στην παραγωγή το 2012, το οποίο έχει μονοπύρηνο επεξεργαστή στα 700MHz και μόλις 256MB RAM. Πλέον το τελευταίο μοντέλο φέρει τετραπύρηνο επεξεργαστή στα 1.4GHz και 1GB RAM. (Για αναλυτικά χαρακτηριστικά δείτε εδώ)

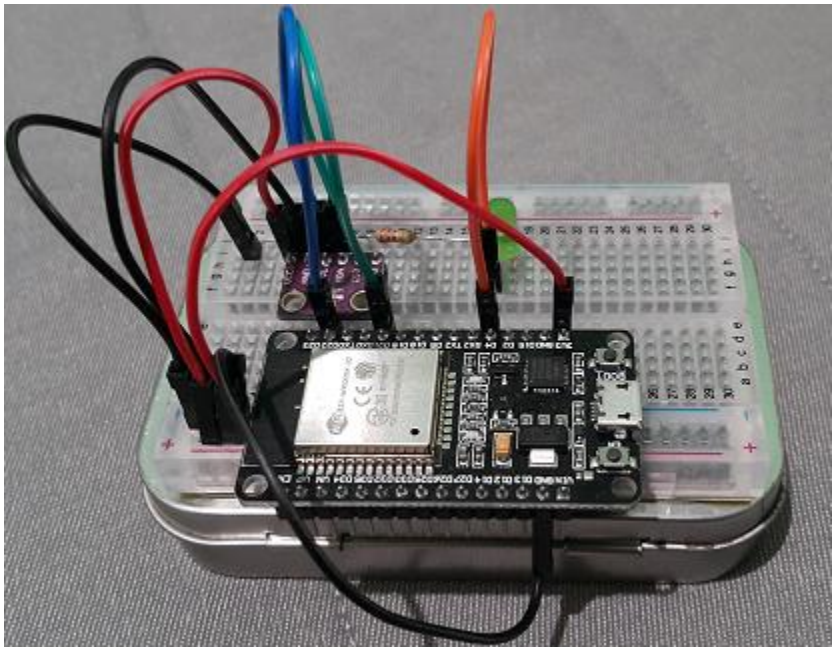
5.3 Espressif ESP32

Είναι ένα χαμηλού κόστους SOC (System on a Chip) το οποίο κατασκευάζεται στην Κίνα από την Espressif Systems. Έχει πολύ χαμηλή απαίτηση ισχύος, λειτουργεί σε θερμοκρασίες μεταξύ -40°C και +125°C και διαθέτει ενσωματωμένο Wi-Fi και dual-mode Bluetooth. Τα παραπάνω αλλά και τα επιπλέον χαρακτηριστικά που διαθέτει, το καθιστούν μια εξαιρετική επιλογή για εφαρμογές IoT.(Για αναλυτικά χαρακτηριστικά δείτε εδώ)

6. Το Κύκλωμα

Για την κατασκευή χρησιμοποιήθηκε πρωτόκολλο επικοινωνίας I2C. Γεφυρώθηκαν οι ακροδέκτες **GPIO 22 (SCL)** και **GPIO 21 (SDA)** του ESP32, αντίστοιχα με τους ακροδέκτες **SCK** και **SDI** του αισθητήρα όπως φαίνεται στο παρακάτω σχηματικό διάγραμμα. Επιπλέον χρησιμοποιήθηκε ο ακροδέκτης **GPIO 4** ως έξοδος για τον έλεγχο του LED.

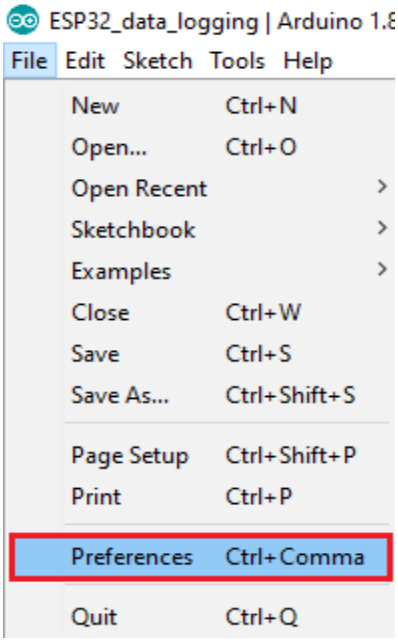




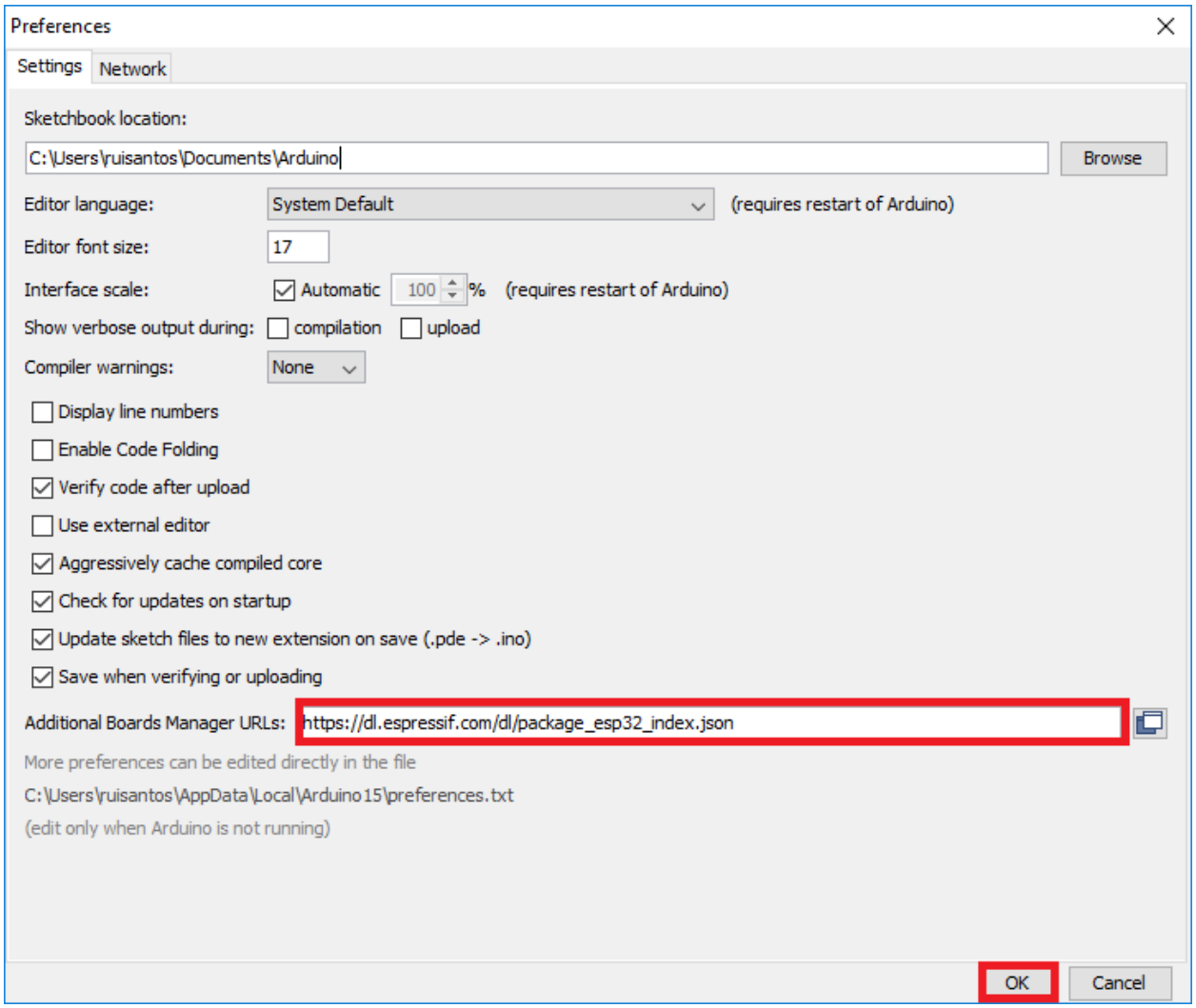
7. Εγκατάσταση και ρύθμιση του Arduino IDE

Για τον προγραμματισμό του ESP32 χρησιμοποιήθηκε η εφαρμογή Arduino IDE, η οποία φέρει μια επέκταση για τον σκοπό αυτό. Είναι διαθέσιμη στην επίσημη ιστοσελίδα. Επιλέχθηκε η έκδοση για λειτουργικό σύστημα Windows:

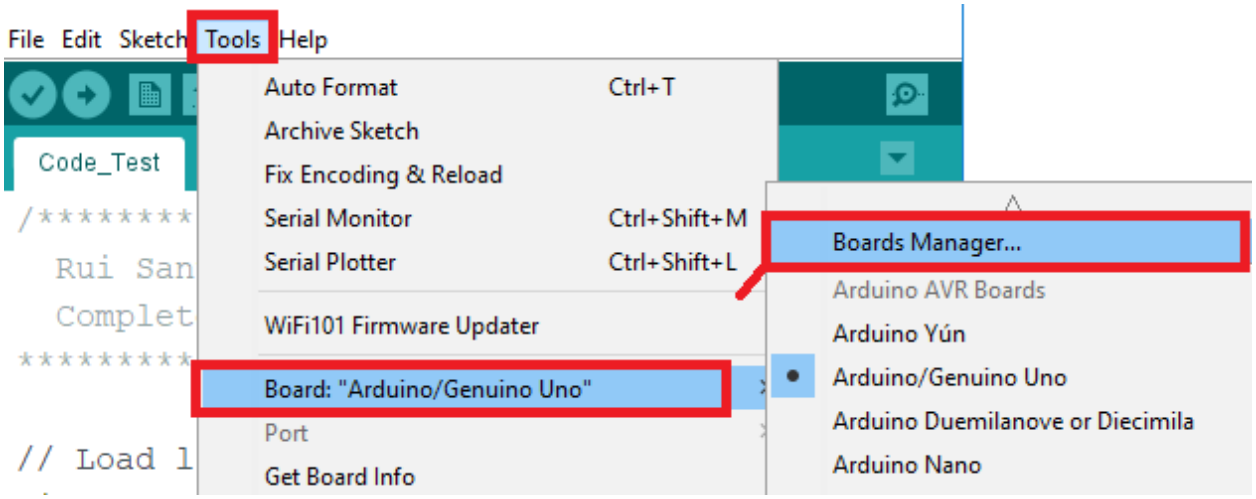
Ολοκληρώνοντας την εγκατάσταση της εφαρμογής, ήταν απαραίτητο να προστεθεί η υποστήριξη του ESP32 εγκαθιστώντας τα απαραίτητα πακέτα. Μέσα από την εφαρμογή επιλέχθηκε **File> Preferences**



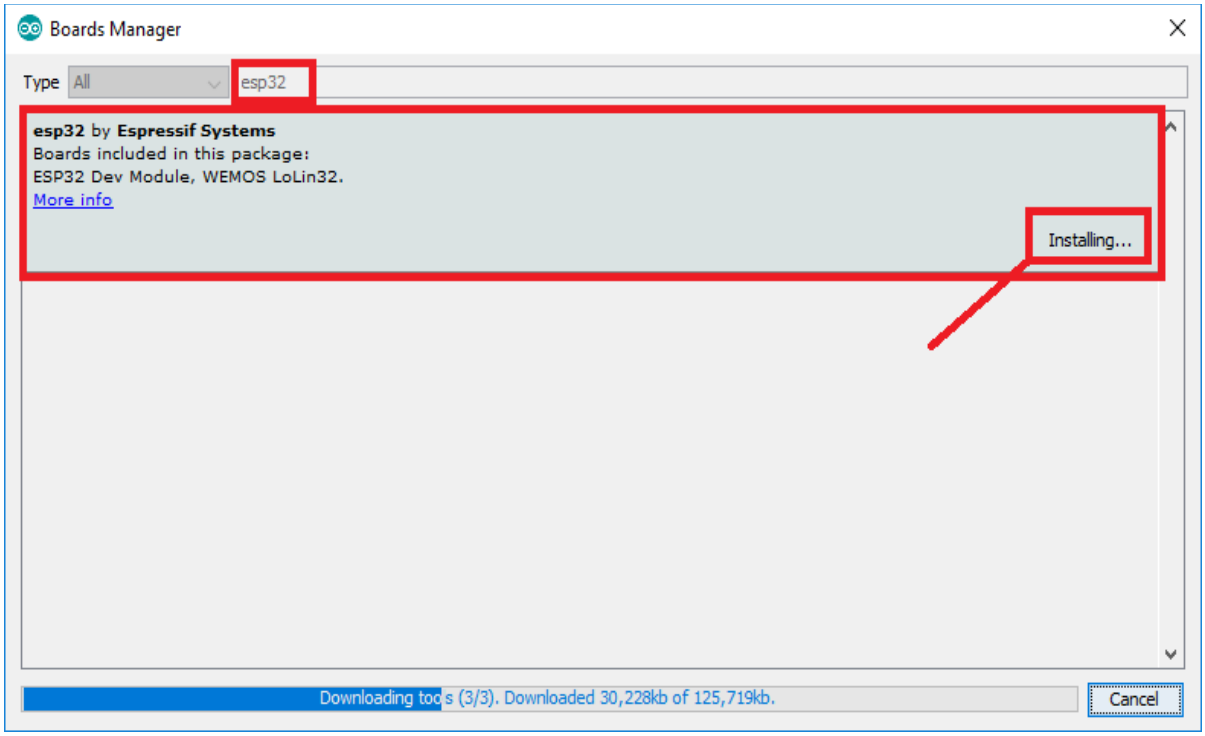
Στη συνέχεια έγινε εισαγωγή του παρακάτω συνδέσμου στο πεδίο ‘’Aditional Board Manager URLs’’: https://dl.espressif.com/dl/package_esp32_index.json



Επιλογή του Boards Manager πηγαίνοντας στο **Tools> Board> Board Manager**.



Μέσα στο πεδίο αναζήτησης πληκτρολογήθηκε esp32 και εγκαταστάθηκε το esp32 by Espressif Systems.



Για την ολοκλήρωση της προετοιμασίας του Arduino IDE προστέθηκαν τρεις απαραίτητες βιβλιοθήκες:

PubSubClient Library

Παρέχει έναν client ο οποίος εκτελεί publish/subscribe επικοινωνία με τον MQTT server. Πρακτικά επιτρέπει την επικοινωνία του ESP32 με το Raspberry διαμέσου Node-RED.

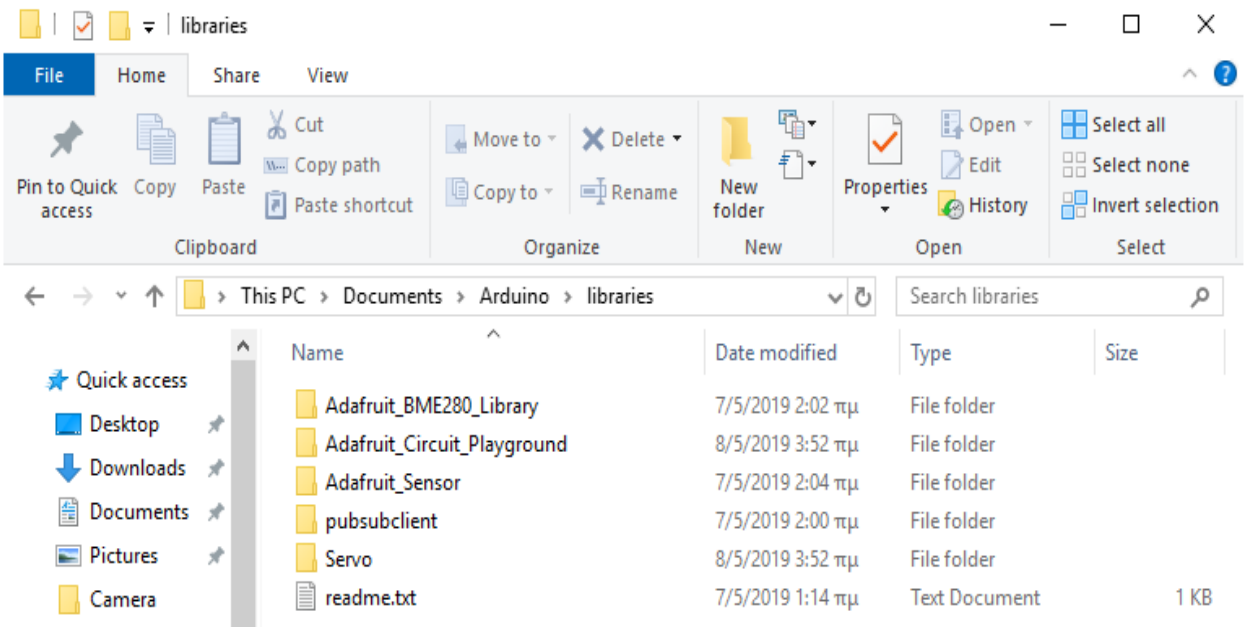
BME280 Library

Χρησιμεύει στο να καθιστά δυνατή την ανάγνωση των τιμών που λαμβάνονται από τον αισθητήρα.

Adafruit_Sensor Library

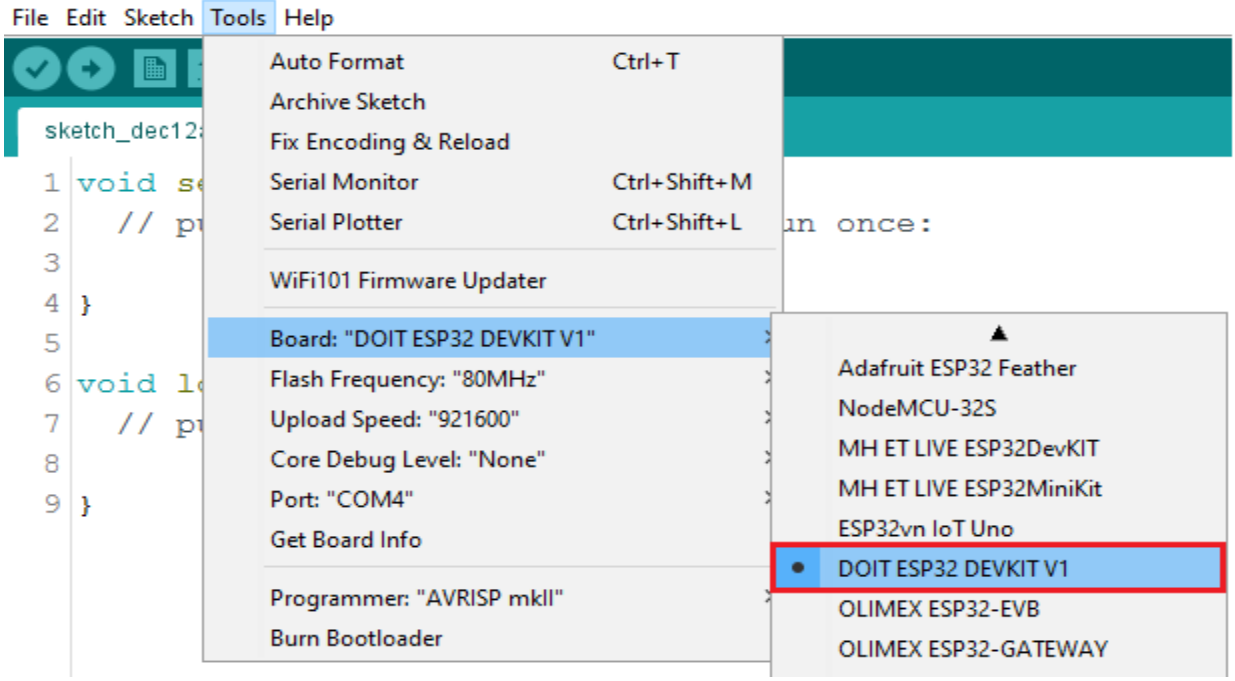
Αναγκαία για τη χρήση της BME280 βιβλιοθήκης.

1. Κατέβασμα των βιβλιοθηκών PubSubclient, BME280 και Adafruit_Sensor.
2. Αποσυμπίεση των φακέλων και μετονομασία τους σε **pubsubclient**, **Adafruit_BME280_Library** και **Adafruit_Sensor** αντίστοιχα.
3. Μετακίνηση των φακέλων που μετονομάστηκαν στον φάκελο εγκατάστασης **libraries** του Arduino IDE.

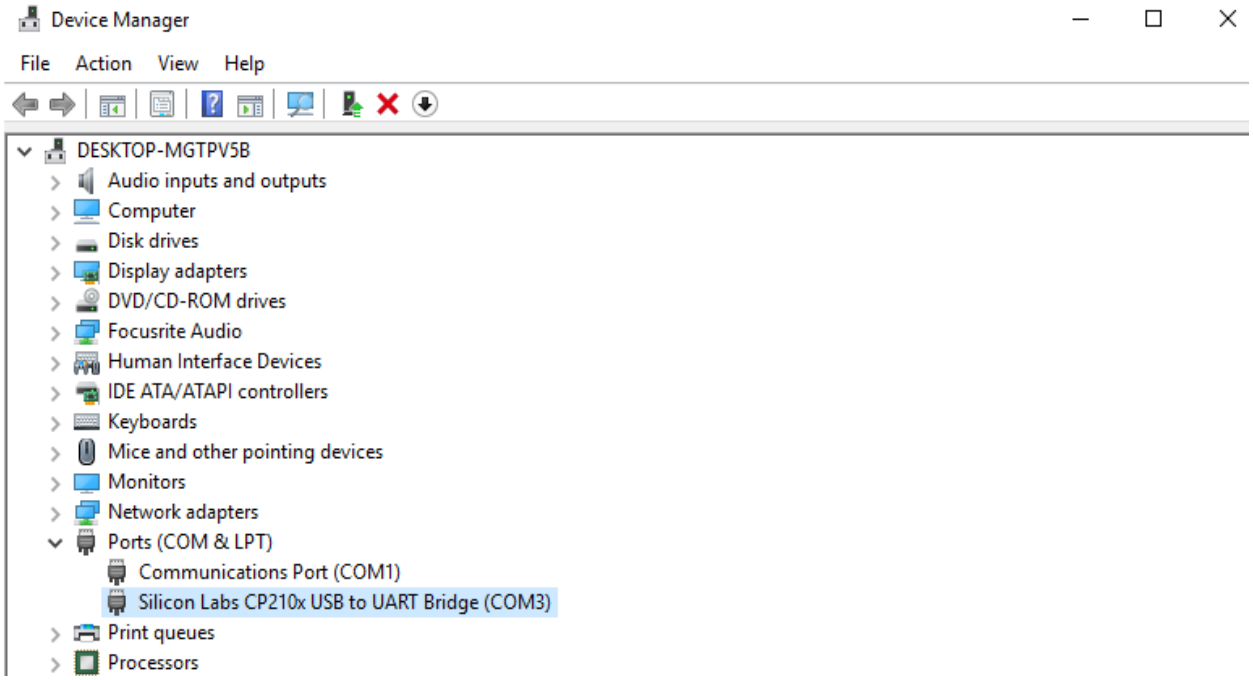


Σύνδεση του ESP32 στον υπολογιστή με τη χρήση ενός micro USB καλωδίου και εκτελέστηκαν τα εξής βήματα:

1. Άνοιγμα της εφαρμογής Arduino IDE.
2. Επιλογή του Board που διαθέτουμε, το οποίο στη συγκεκριμένη περίπτωση είναι το **DOIT ESP32 DEVKIT V1**, από το **Tools> Board**.

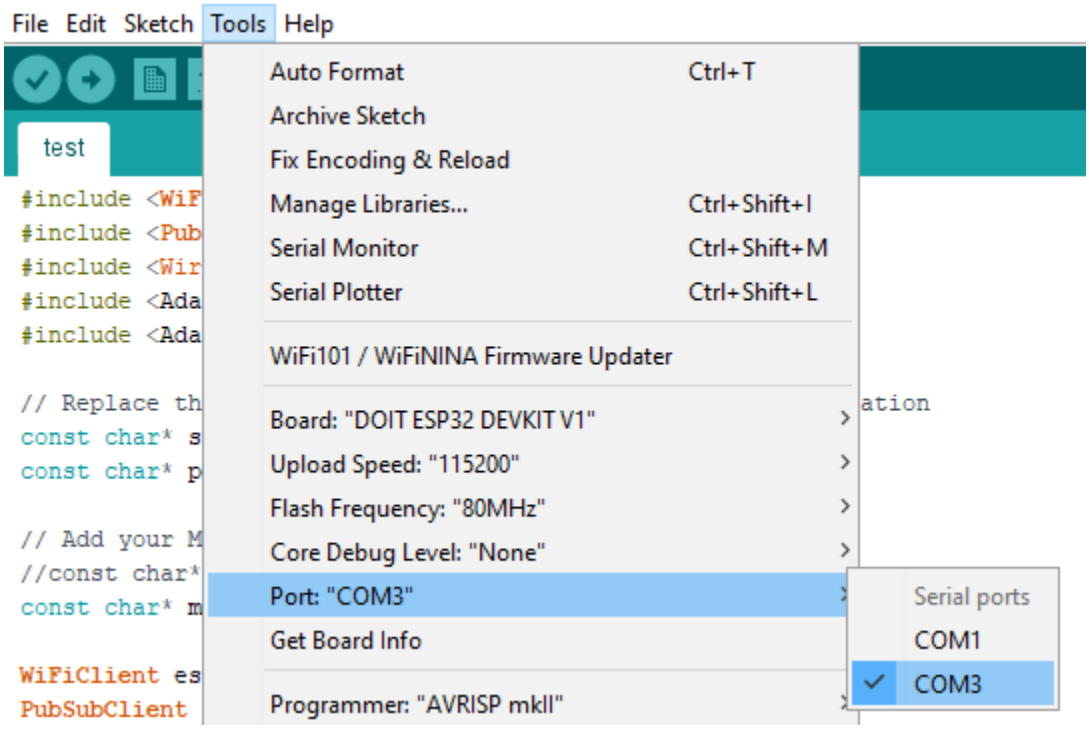


3. Μέσα από τη Διαχείριση Συσκευών (Device Manager) ελέγχθηκε σε ποιο Port του Η/Υ έχει αντιστοιχηθεί το ESP32.



Να σημειωθεί ότι σε περίπτωση που το ESP32 δεν αναγνωρίζεται από τον Η/Υ είναι απαραίτητο να εγκατασταθούν οδηγοί οι οποίοι είναι διαθέσιμοι εδώ.

Μετά τον έλεγχο στη Διαχείριση Συσκευών, επιλέγουμε το Port.



8. Προγραμματισμός του ESP32

Παρακάτω είναι ο κώδικας που χρησιμοποιήθηκε για τον προγραμματισμό του ESP32. Απαραίτητη είναι η εισαγωγή του ssid και του password από το ασύρματο δίκτυο, όπως επίσης και η διεύθυνση ip του Raspberry Pi.

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_BME280.h>
#include <Adafruit_Sensor.h>

// Replace the next variables with your SSID/Password combination
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Add your MQTT Broker IP address, example:
//const char* mqtt_server = "192.168.1.144";
const char* mqtt_server = "YOUR_MQTT_BROKER_IP_ADDRESS";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

//uncomment the following lines if you're using SPI
/*#include <SPI.h>
#define BME_SCK 18
#define BME_MISO 19
#define BME_MOSI 23
#define BME_CS 5*/

Adafruit_BME280 bme; // I2C
//Adafruit_BME280 bme(BME_CS); // hardware SPI
//Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK); // software SPI
float temperature = 0;
float humidity = 0;

// LED Pin
const int ledPin = 4;

void setup() {
  Serial.begin(115200);
  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  //status = bme.begin();
  if (!bme.begin(0x76)) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
  setup_wifi();
```

```
47.   client.setServer(mqtt_server, 1883);
48.   client.setCallback(callback);
49.
50.   pinMode(ledPin, OUTPUT);
51. }
52.
53. void setup_wifi() {
54.   delay(10);
55.   // We start by connecting to a WiFi network
56.   Serial.println();
57.   Serial.print("Connecting to ");
58.   Serial.println(ssid);
59.
60.   WiFi.begin(ssid, password);
61.
62.   while (WiFi.status() != WL_CONNECTED) {
63.     delay(500);
64.     Serial.print(".");
65.   }
66.
67.   Serial.println("");
68.   Serial.println("WiFi connected");
69.   Serial.println("IP address: ");
70.   Serial.println(WiFi.localIP());
71. }
72.
73. void callback(char* topic, byte* message, unsigned int length) {
74.   Serial.print("Message arrived on topic: ");
75.   Serial.print(topic);
76.   Serial.print(". Message: ");
77.   String messageTemp;
78.
79.   for (int i = 0; i < length; i++) {
80.     Serial.print((char)message[i]);
81.     messageTemp += (char)message[i];
82.   }
83.   Serial.println();
84.
85.   // Feel free to add more if statements to control more GPIOs with MQTT
86.
87.   // If a message is received on the topic esp32/output, you check if the messag
    e is either "on" or "off".
88.   // Changes the output state according to the message
89.   if (String(topic) == "esp32/output") {
90.     Serial.print("Changing output to ");
91.     if(messageTemp == "on"){
92.       Serial.println("on");
93.       digitalWrite(ledPin, HIGH);
94.     }
95.     else if(messageTemp == "off"){
96.       Serial.println("off");
97.       digitalWrite(ledPin, LOW);
98.     }
99.   }
```

```
}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Subscribe
      client.subscribe("esp32/output");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 5000) {
    lastMsg = now;

    // Temperature in Celsius
    temperature = bme.readTemperature();

    // Convert the value to a char array
    char tempString[8];
    dtostrf(temperature, 1, 2, tempString);
    Serial.print("Temperature: ");
    Serial.println(tempString);
    client.publish("esp32/temperature", tempString);

    humidity = bme.readHumidity();

    // Convert the value to a char array
    char humString[8];
    dtostrf(humidity, 1, 2, humString);
    Serial.print("Humidity: ");
    Serial.println(humString);
    client.publish("esp32/humidity", humString);
  }
}
```

8.1 Συνοπτική ανάλυση του κώδικα

Ο παραπάνω κώδικας στέλνει(κάνει publish) τιμές θερμοκρασίας και υγρασίας στα topics esp32/temperature και esp32/humidity αντίστοιχα δια μέσου του πρωτόκολλου MQTT.

Το ESP32 είναι εγγεγραμμένο(subscribed) στο topic esp32/output προκειμένου να λαμβάνει μηνύματα που γίνονται publish από το Node-RED. Με τον τρόπο αυτό ανάβει ή σβήνει το LED, ανάλογα με το μήνυμα που έχει λάβει.

Subscribe σε MQTT topics

Μέσα στη συνάρτηση *reconnect()* δίνεται η δυνατότητα εγγραφής(subscribe) σε MQTT topics. Στη συγκεκριμένη περίπτωση, μόνο το ESP32 έχει εγγραφεί στο esp32/output:

```
150.            client.subscribe("esp32/output");
```

Μέσα στη συνάρτηση *callback()*, το ESP32 λαμβάνει μηνύματα από τα εγγεγραμμένα topics. Ανάβει ή σβήνει το LED ανάλογα με το MQTT topic και το μήνυμα:

```
87.   // If a message is received on the topic esp32/output, you check if the messag
   e is either "on" or "off".
88.   // Changes the output state according to the message
89.   if (String(topic) == "esp32/output") {
90.     Serial.print("Changing output to ");
91.     if(messageTemp == "on"){
92.       Serial.println("on");
93.       digitalWrite(ledPin, HIGH);
94.     }
95.     else if(messageTemp == "off"){
96.       Serial.println("off");
97.       digitalWrite(ledPin, LOW);
98.     }
99.   }
```

Publish τα μηνύματα MQTT

Μέσα στη συνάρτηση *loop()* κάνουμε publish τις νέες μετρήσεις από τον αισθητήρα κάθε 5 δευτερόλεπτα:

```
127.        if (now - lastMsg > 5000) {...
```

Για να είναι εφικτό το publish της τιμής της θερμοκρασίας, είναι απαραίτητη η μετατροπή της μεταβλητής float σε πίνακα char array:

```
133.          // Convert the value to a char array
134.          char tempString[8];
135.          dtostrf(temperature, 1, 2, tempString);
136.          Serial.print("Temperature: ");
137.          Serial.println(tempString);
138.          client.publish("esp32/temperature", tempString);
```

Η ίδια διαδικασία απαιτείται και για τις τιμές υγρασίας:

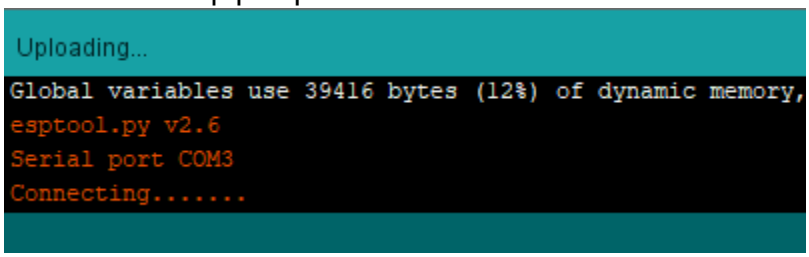
```
142.          // Convert the value to a char array
143.          char humString[8];
144.          dtostrf(humidity, 1, 2, humString);
145.          Serial.print("Humidity: ");
146.          Serial.println(humString);
147.          client.publish("esp32/humidity", humString);
```

8.2 Upload του κώδικα στο ESP32

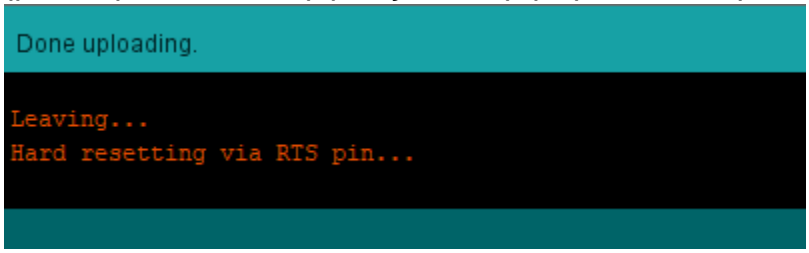
Έχοντας συνδέσει το ESP32 στον Η/Υ κάναμε τα εξής:

1. Μέσα από το Arduino IDE επιλέχθηκε το Upload  .

2. Μόλις δούμε το μήνυμα ‘’Connecting’’ στην κονσόλα, πατάμε το κουμπί **BOOT** επάνω στο ESP32 και το αφήνουμε.



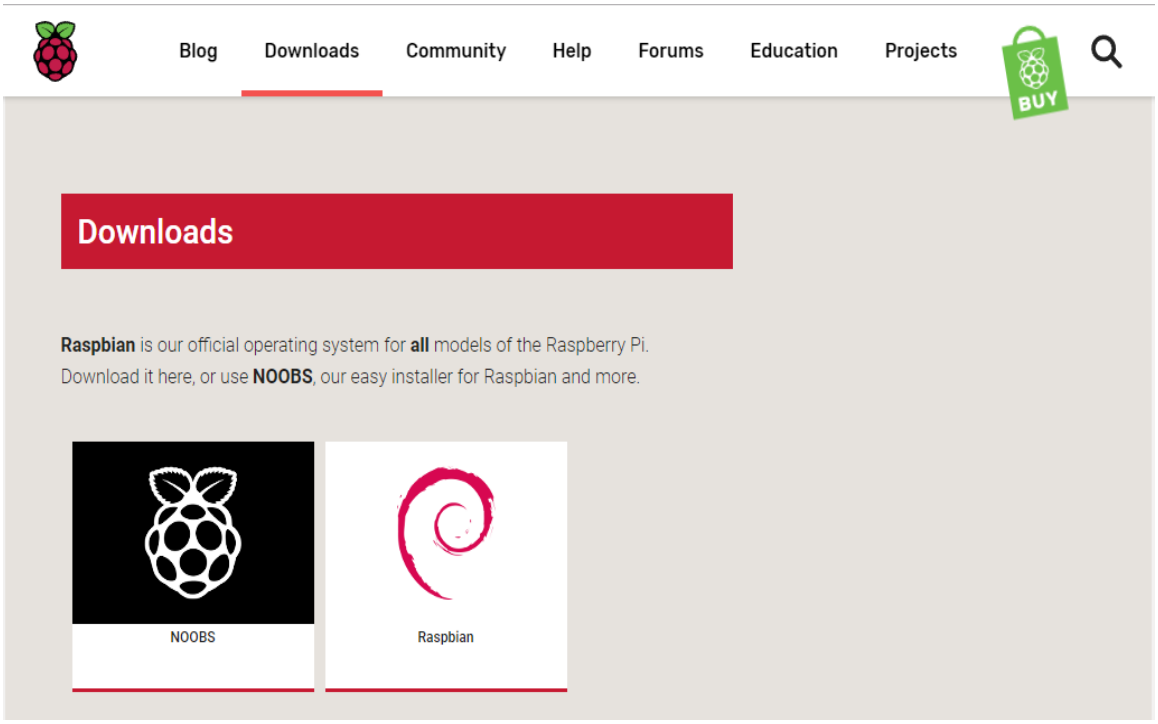
3. Όταν ολοκληρωθεί η διαδικασία εμφανίζεται το μήνυμα ‘’Done uploading’’.



9. Προετοιμασία του Raspberry Pi

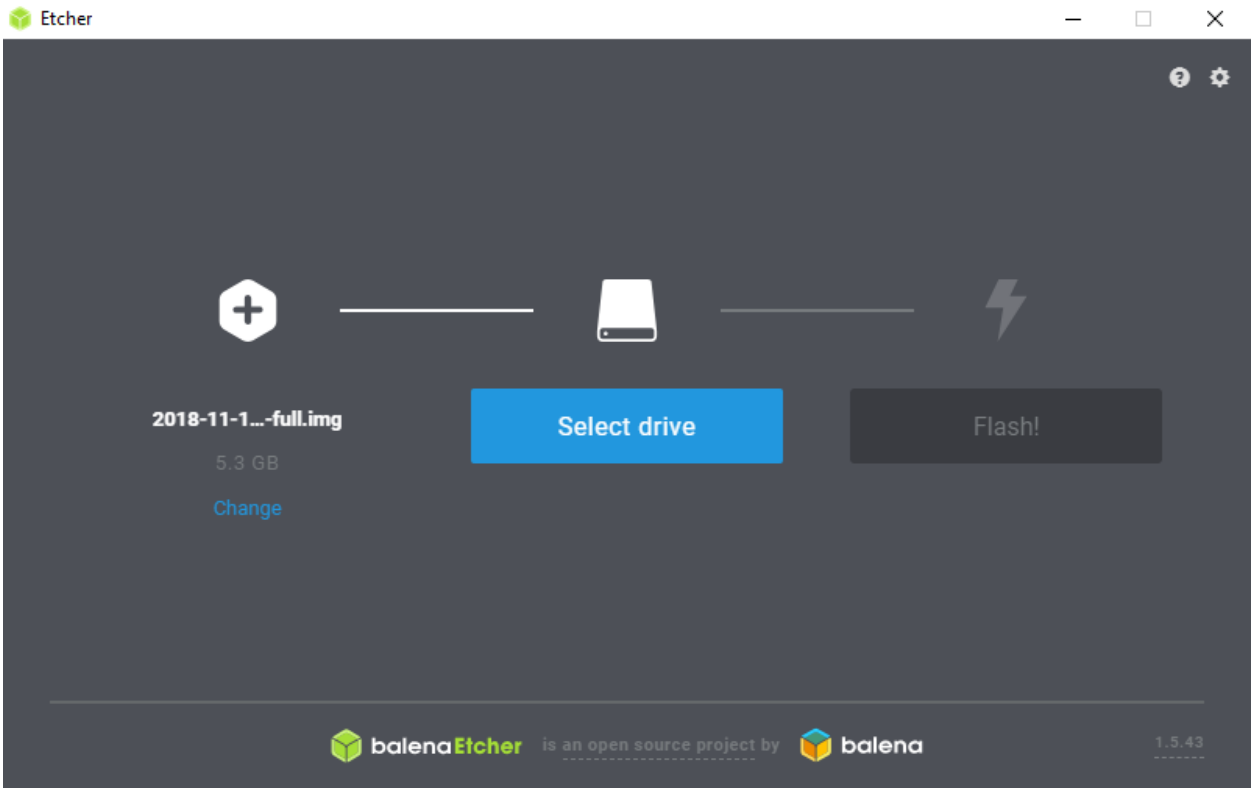
Ως ηλεκτρονικός υπολογιστής απαιτείται η εγκατάσταση λειτουργικού συστήματος κι επειδή δεν φέρει ενσωματωμένη μνήμη, χρησιμοποιούμε μια κάρτας μνήμης τύπου microSD (*Ελάχιστο προτεινόμενο μέγεθος 8GB*).

Υπάρχουν διαθέσιμα λειτουργικά συστήματα στην επίσημη ιστοσελίδα του κατασκευαστή και μπορεί κανείς να τα κατεβάσει από εδώ. Χρησιμοποιήθηκε το Raspbian καθώς έχει μεγάλη υποστήριξη από την Raspberry Pi κοινότητα.



Για την εγκατάσταση του κατεβασμένου αρχείου .zip χρησιμοποιήθηκε η εφαρμογή Etcher, όπως άλλωστε προτείνεται στις οδηγίες εγκατάστασης του κατασκευαστή. Η συγκεκριμένη εφαρμογή μας δίνει τη δυνατότητα εγγραφής στην microSD χωρίς να είναι απαραίτητη η αποσυμπίεση του αρχείου και μπορεί κανείς να την κατεβάσει από εδώ.

Ανοίγοντας την εφαρμογή ακολουθήθηκαν τα παρακάτω βήματα:

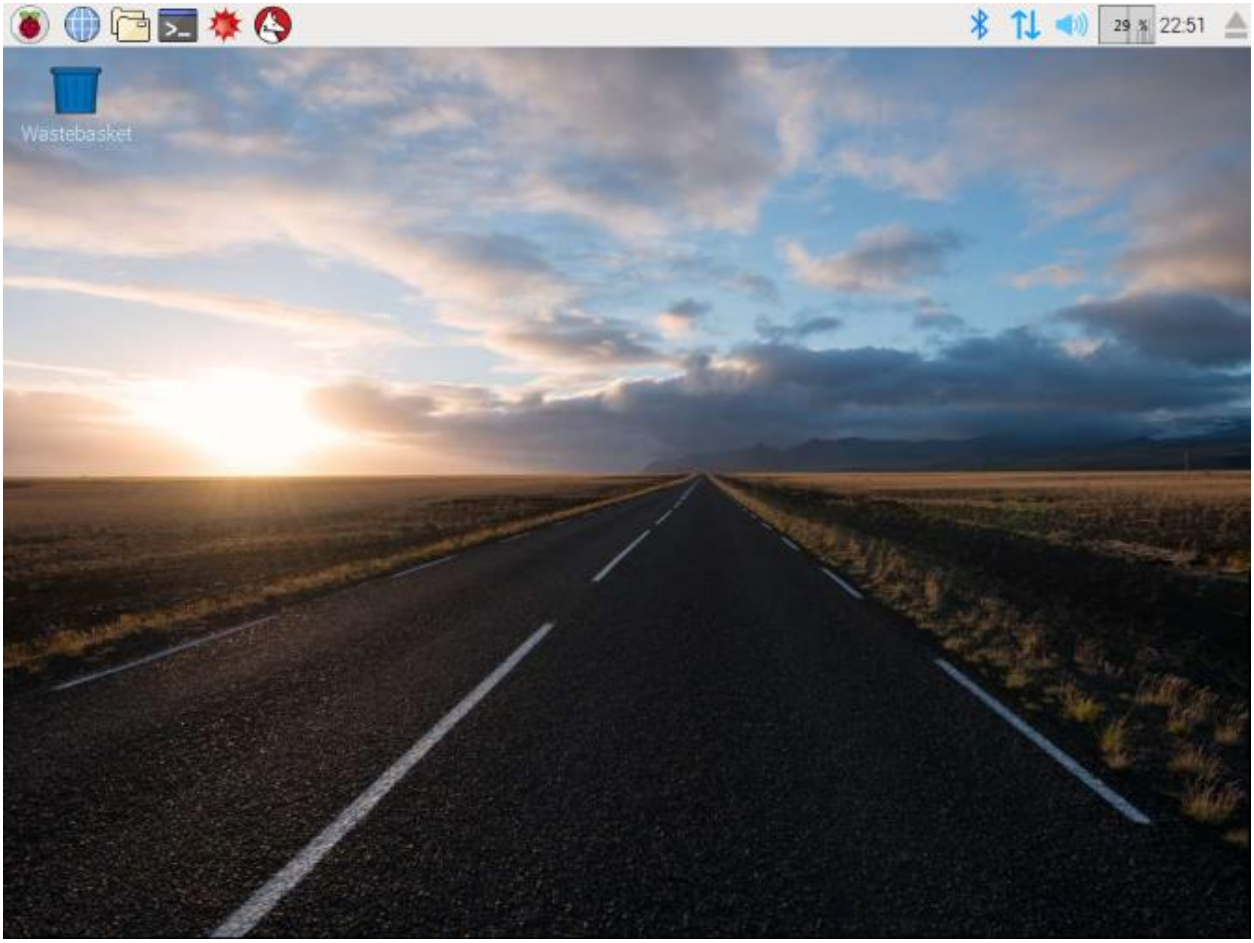


1. Επιλογή του αρχείο .zip από τον φάκελο λήψεων.
2. Επιλογή της microSD ως drive.
3. Επιλογή του Flash! ώστε να ξεκινήσει η διαδικασία εγγραφής της κάρτας μνήμης.

Μετά την ολοκλήρωση της εγγραφής συνδέσαμε την microSD, την οθόνη, το πληκτρολόγιο & το ποντίκι και τέλος τον μετασχηματιστή 5V DC στο Raspberry Pi. Από τις επιλογές εγκατάστασης που εμφανίζονται, επιλέχθηκε να εγκατασταθεί το Raspbian πατώντας Install.

Εμφανίζεται σχετικό μήνυμα μετά την ολοκλήρωση:

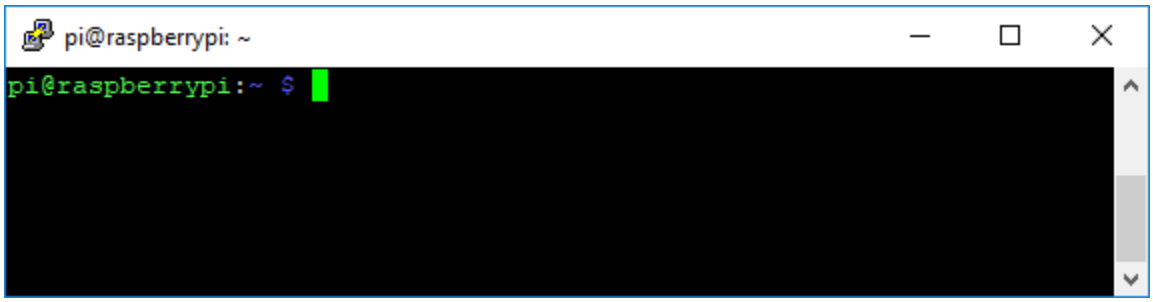
Εκτελώντας επανεκκίνηση με την επιλογή OK, ολοκληρώθηκε η διαδικασία εγκατάστασης του Raspbian.



10.Εγκατάσταση του Mosquitto στο Raspberry

Αν και υπάρχουν αρκετοί broker, επιλέξαμε να εγκαταστήσουμε τον Mosquitto Broker στο Raspberry Pi.

Ανοίγοντας ένα νέο terminal window πληκτρολογήθηκαν οι εξής εντολές:




```
pi@raspberry:~ $ sudo apt update

pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients

//type Y and press Enter to confirm installation

//To make Mosquitto auto start on boot up

pi@raspberry:~ $ sudo systemctl enable mosquitto.service
```

11. Εγκατάσταση του Node-RED στο Raspberry

Το Node-RED είναι ένα πανίσχυρο εργαλείο ανοιχτού κώδικα ανεπτυγμένο από την IBM στόχος του οποίου είναι η απλούστευση της δημιουργίας εφαρμογών IoT. Κάνοντας χρήση visual προγραμματισμού επιτρέπει τη σύνδεση διαφόρων block κώδικα, γνωστά ως nodes, για την εκτέλεση μιας εργασίας(Flow). Με τον τρόπο αυτόν ο χρήστης εξοικονομεί ώρες που θα χρειαζόταν σε ανάπτυξη κώδικα.

Μερικές από τις δυνατότητες που μας προσφέρει είναι:

- Πρόσβαση στα GPIOs του Raspberry Pi.
- Επίτευξη MQTT σύνδεσης με άλλα boards. (Arduino, ESP32, κλπ.)
- Δημιουργία γραφικού περιβάλλοντος για projects.
- Ανάκτηση δεδομένων από το διαδίκτυο. (emails, πρόγνωση καιρού, τιμές μετοχών, κλπ.)
- Αποθήκευση αλλά και ανάκτηση δεδομένων από μια βάση δεδομένων.
- Επικοινωνία με third-party services. (Adafruit.io, Thing Speak, κλπ.)

Η εγκατάσταση του Node-RED στο Raspberry Pi είναι απλή εκτελώντας τις εξής εντολές:

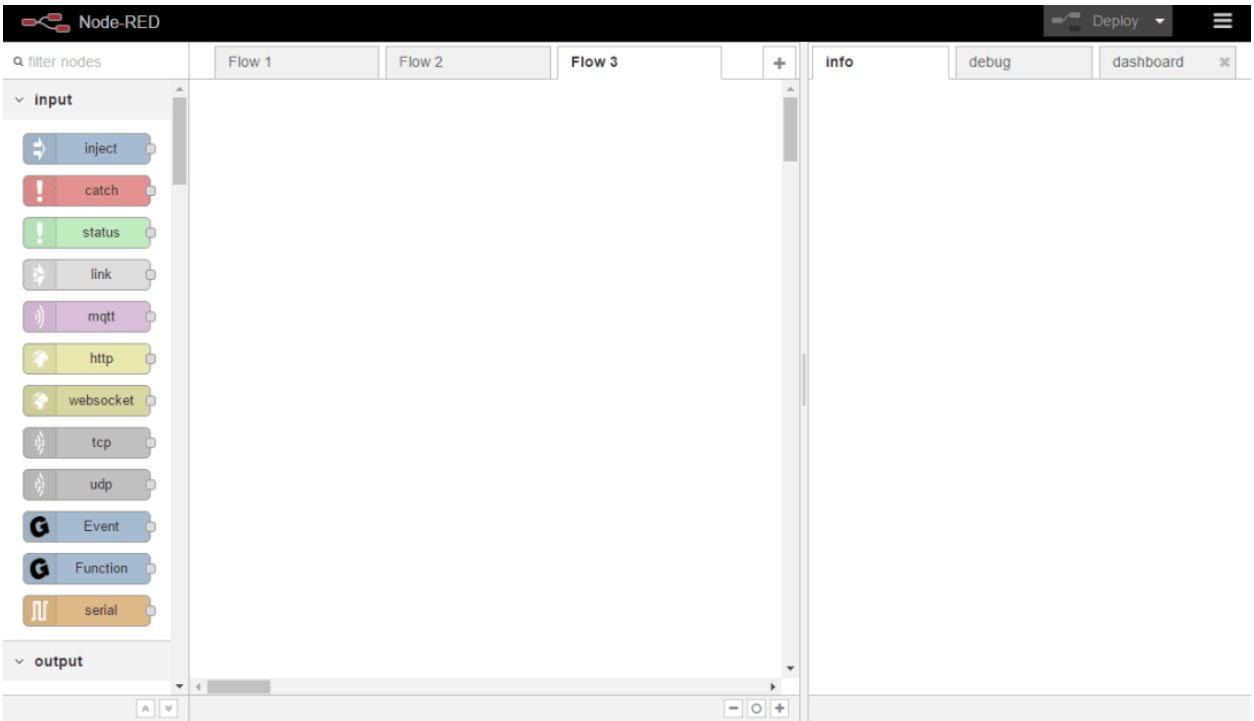
```
//To install Node-RED

pi@raspberry:~ $ bash <(curl -sL
https://raw.githubusercontent.com/node-red/ raspbian-deb-
package/master/resources/update-nodejs-and-nodered)

//To auto run Node-RED when the Raspberry boots up

pi@raspberry:~ $ sudo systemctl enable nodered.service

//Restarts the Raspberry Pi so the auto start takes effect

pi@raspberry:~ $ sudo reboot
```

11.1 Node-RED visual interface

Η πρόσβαση στο Node-RED visual interface γίνεται με την εισαγωγή σε έναν περιηγητή της διεύθυνσης ip του Raspberry Pi ακολουθούμενη από το port number 1880:

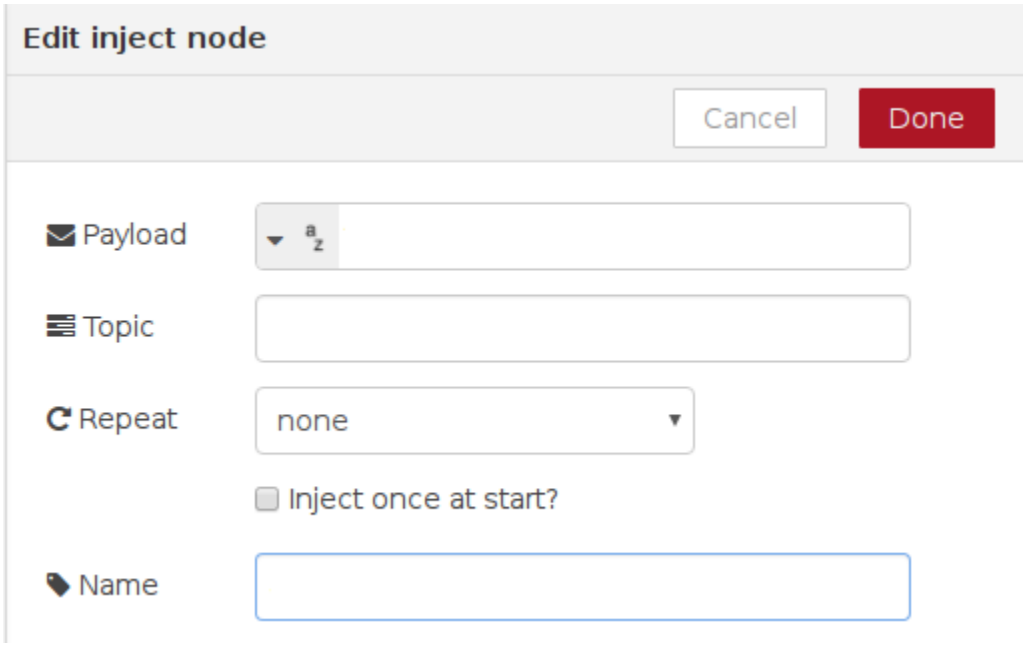
```
http://RPi_ip_adress:1880
```

Εμφανίζεται μια σελίδα όπως φαίνεται στην παρακάτω εικόνα:

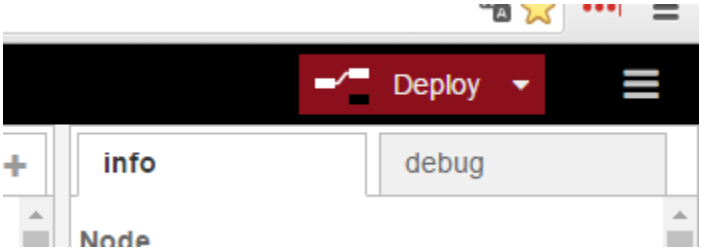


Στην αριστερή πλευρά βρίσκεται η λίστα με τα nodes τα οποία χωρίζονται σε κατηγορίες ανάλογα με τη λειτουργικότητα τους. Επιλέγοντας ένα Node εμφανίζονται πληροφορίες για το πως λειτουργεί στο info tab. Στο κέντρο βρίσκονται τα Flow πάνω στα οποία τοποθετούνται τα nodes.

Αφού τοποθετήσουμε nodes τα ενώνουμε μεταξύ τους και αν επιθυμούμε να τα επεξεργαστούμε, κάνουμε διπλό κλικ επάνω τους. Ανοίγει ένα μενού με διάφορες ρυθμίσεις ανάλογα με την λειτουργικότητα του node:



Αν επιθυμούμε να σώσουμε την εφαρμογή μας επιλέγουμε το Deploy:



11.2 Προσθήκη του Node-RED Dashboard

Το Node-RED Dashboard είναι ένα πρόσθετο module το οποίο παρέχει ένα σετ από Nodes για την γρήγορη δημιουργία ενός live data dashboard.

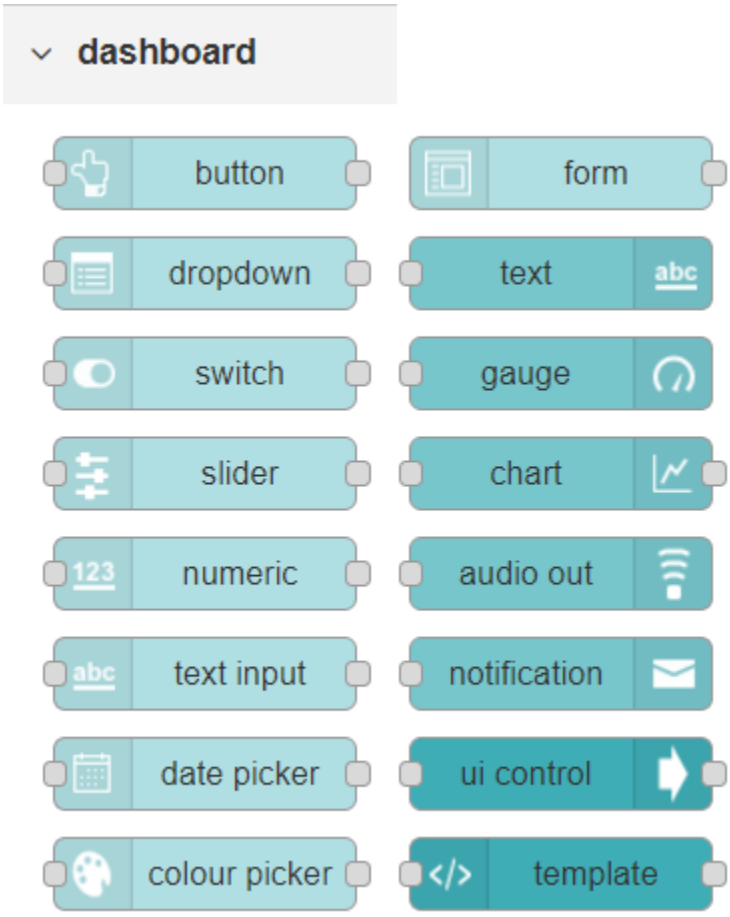
Η εγκατάσταση του έγινε με τις παρακάτω εντολές:

```
pi@raspberry:~ $ node-red-stop
```

```
pi@raspberry:~ $ cd ~/.node-red
```

```
pi@raspberry:~/.node-red $ npm install node-red-dashboard
```

Με την ολοκλήρωση της εγκατάστασης εμφανίζεται μια επιπλέον κατηγορία στην λίστα των nodes με την ονομασία dashboard, παρέχοντας επιπλέον widgets για την διαμόρφωση του user interface:

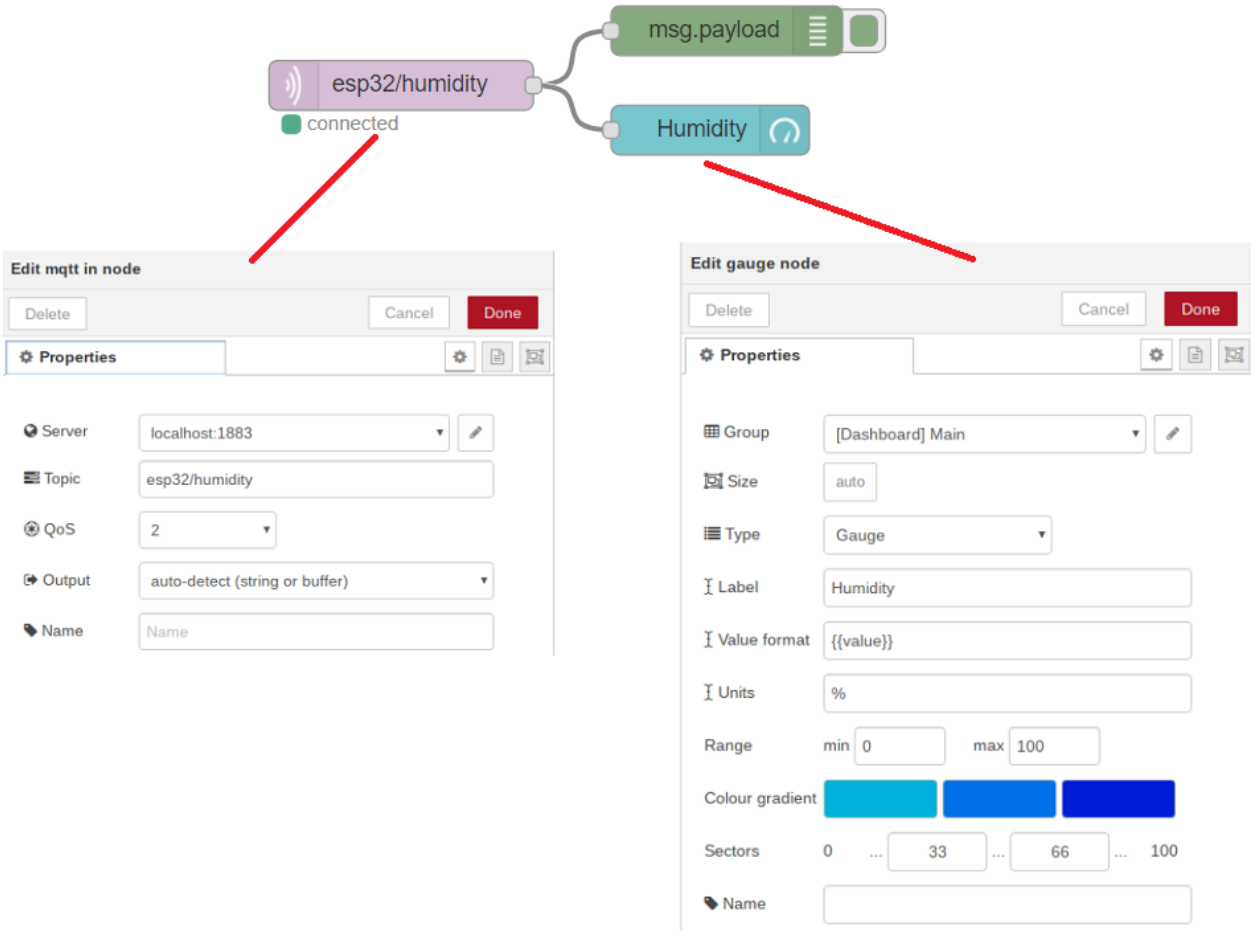


11.3 Δημιουργία του Flow

Για το ζητούμενο της εργασίας και τη δημιουργία του αντίστοιχου Flow χρειάστηκαν συνολικά 8 nodes.

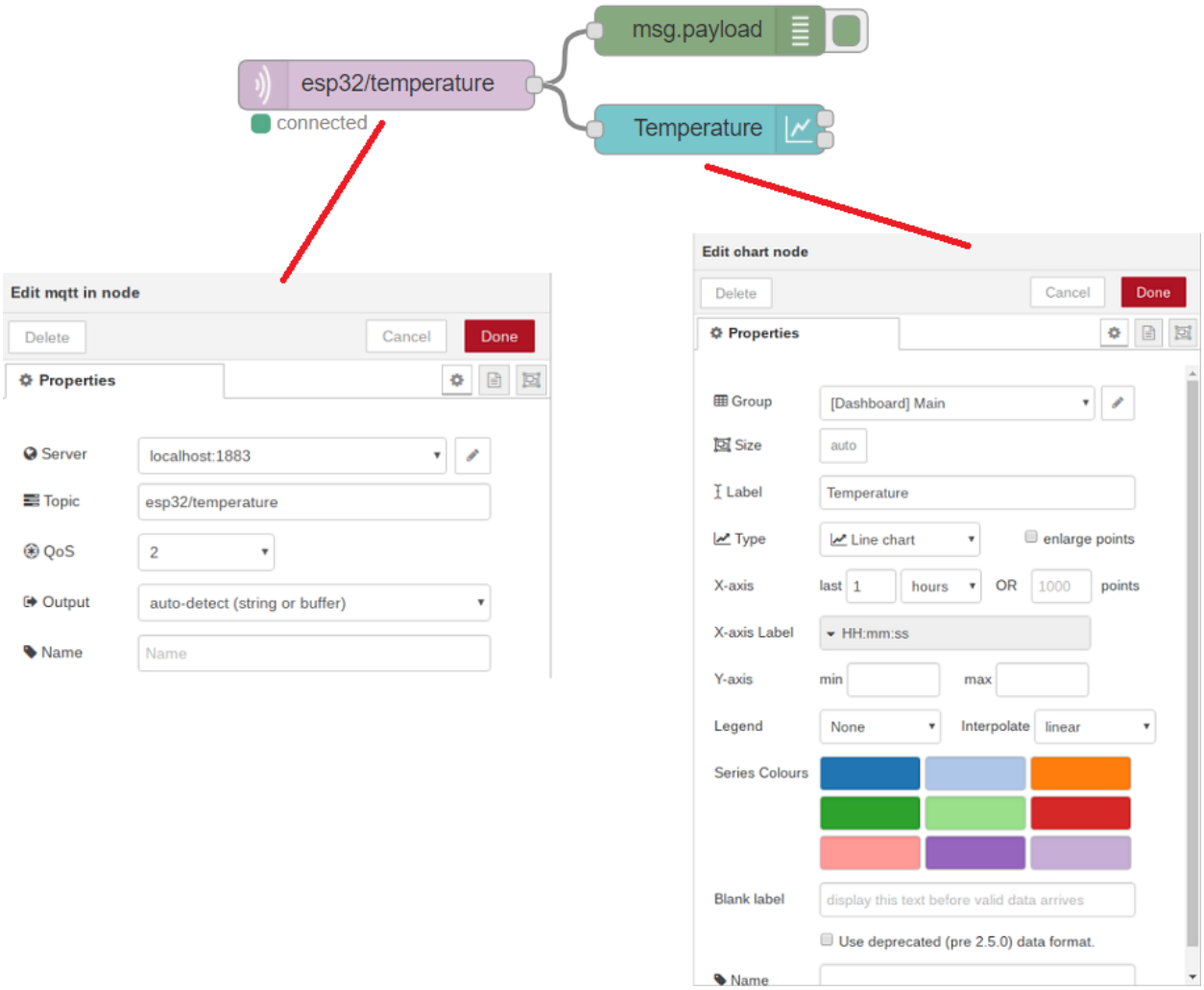
Για την ανάγνωση υγρασίας χρησιμοποιήθηκαν:

- 1x mqtt input node με Topic: esp32/humidity.
- 1x gauge dashboard node με Group: [Dashboard] Main, Label: Humidity, Units: %, Range: min=0 και max=100, Colour gradient: όπως φαίνεται στην εικόνα, Sectors: 0…33…66…100.
- 1x debug output node.



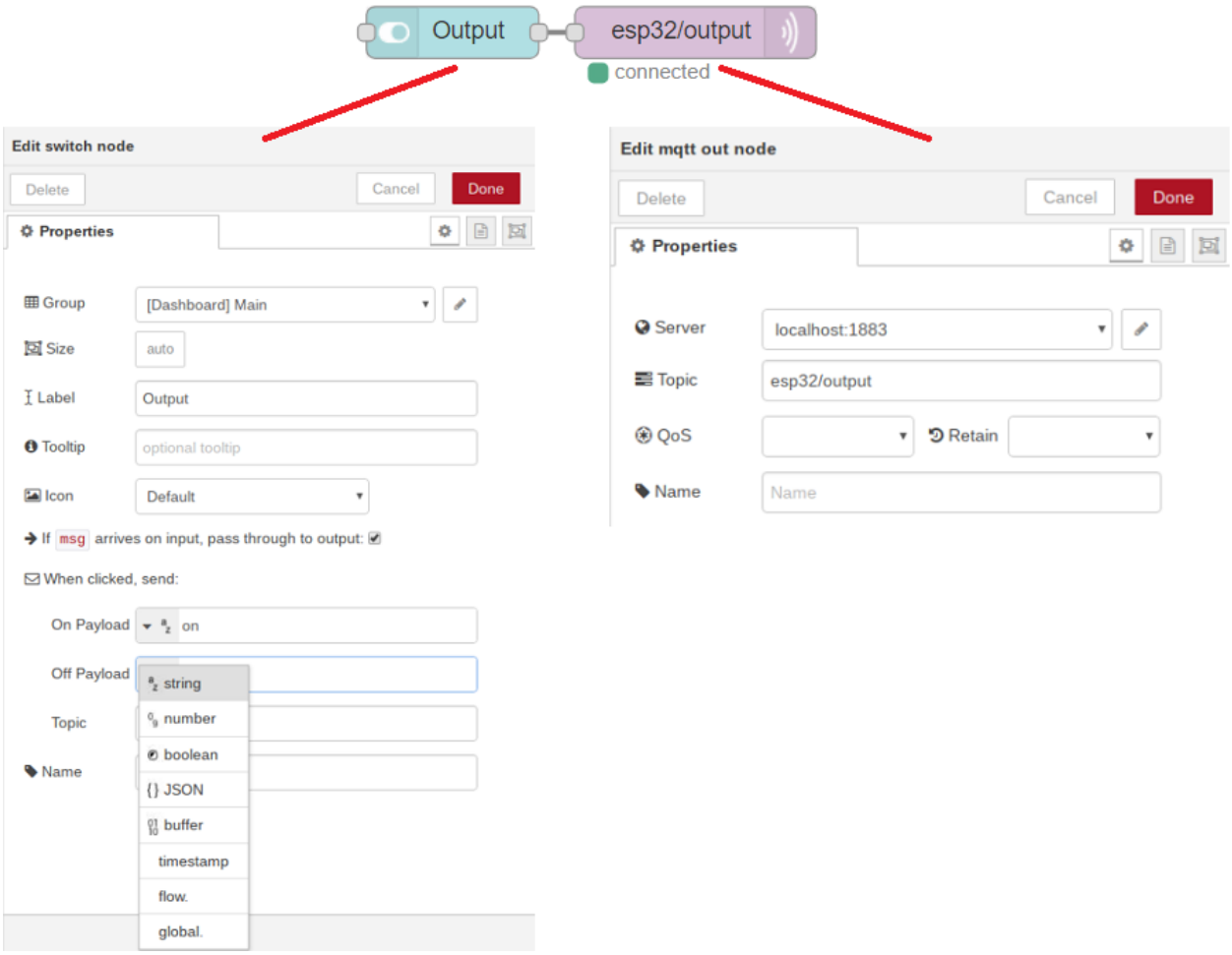
Για την ανάγνωση της θερμοκρασίας χρησιμοποιήθηκαν:

- 1x mqtt input node με Topic: esp32/temperature.
- 1x chart dashboard node με Group: [Dashboard] Main, Label: Temperature
- 1x debug output node.

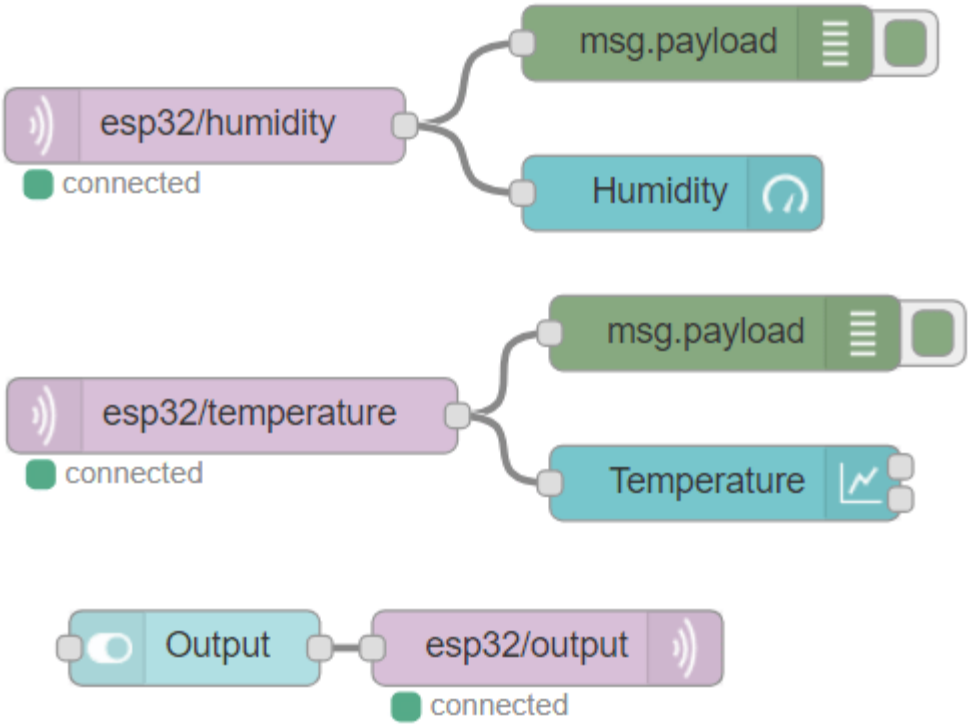


Για την δημιουργία του διακόπτη που θα κάνει on/off το LED χρησιμοποιήθηκαν:

- 1x switch dashboard node με Group: [Dashboard] Main, Label: Output, On Payload as string: on, Off Payload as string: off.
- 1x mqtt output node με Topic: esp32/output.



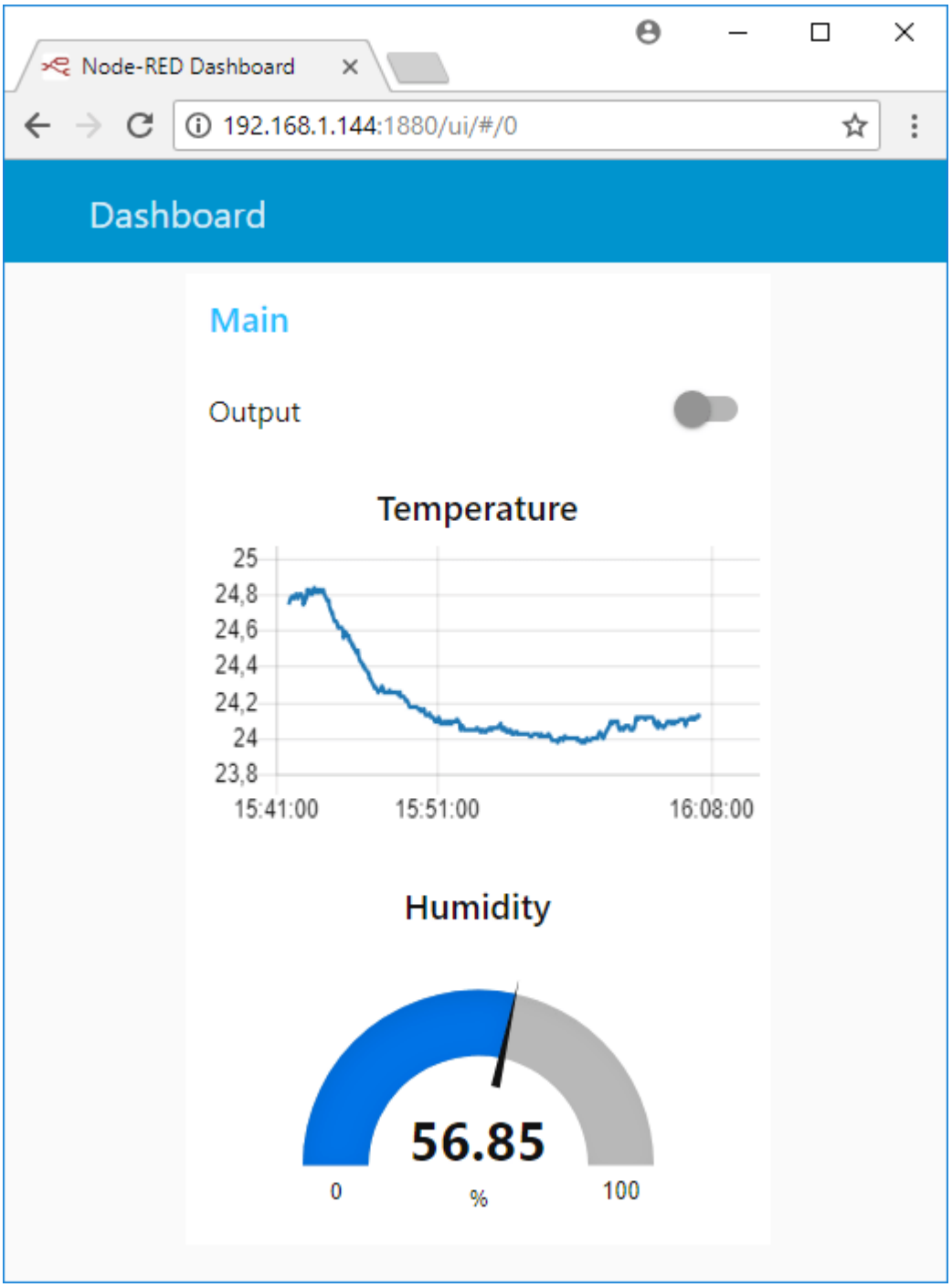
Η παρακάτω εικόνα απεικονίζει το Flow στην τελική του μορφή:



Η πρόσβαση στο Node-RED user interface γίνεται με την εισαγωγή σε έναν περιηγητή της διεύθυνσης ip του Raspberry Pi ακολουθούμενη από το port number 1880, forward slash (/) ui:

```
http://RPi_ip_adress:1880/ui
```

Αυτή είναι τελική εικόνα που έχει κάθε client:



Απεικονίζεται υπό τη μορφή γραφήματος η θερμοκρασία, η υγρασία ως ποσοστό σε δείκτη και υπάρχει η δυνατότητα ελέγχου του LED μέσω διακόπτη.

12. Βιβλιογραφία

[1] D. Evans, (Απρίλιος 2011), ‘’Internet of Things – How the Next Evolution of The Internet Is Changing Everything’’, Διαθέσιμο: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (τελευταία πρόσβαση στις 10/06/2019).

[2] CISCO, ‘’Internet of Thing – At a Glance’’, Διαθέσιμο https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf (τελευταία πρόσβαση στις 10/06/2019).

[3] K. Ashton, (22 Ιουνίου 2009), ‘’That ‘Internet of Things’ Thing’’, Διαθέσιμο: https://www.rfidjournal.com/articles/view?4986 (τελευταία πρόσβαση στις 10/06/2019).

[4] OASIS, (10 Δεκεμβρίου 2015), ‘’MQTT Version 3.1.1 Plus Errata 01 - Abstract’’, Διαθέσιμο http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (τελευταία πρόσβαση στις 10/06/2019).