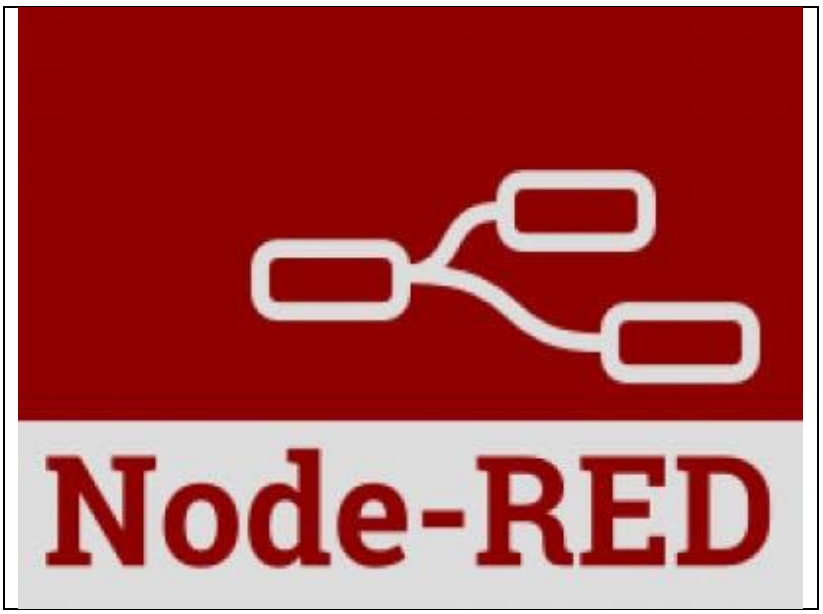
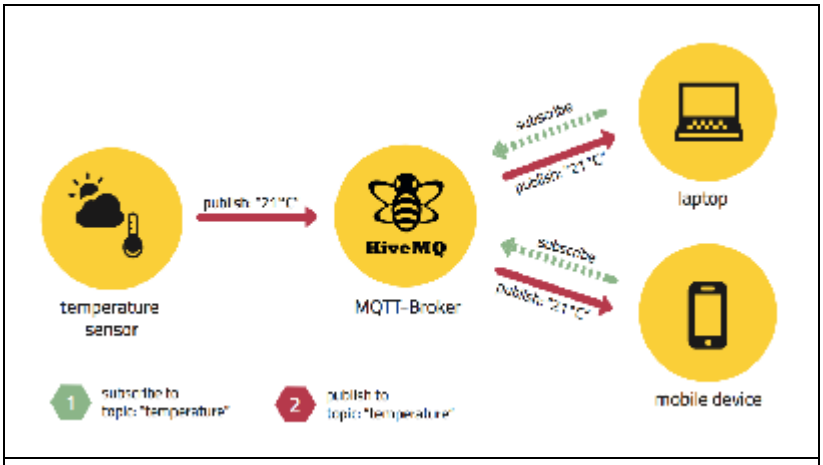
Μετεωρολογικός Σταθμός με χρήση Arduino sensors - MQTT Broker- Node RED

2019





Καθηγητής : Γκιουλέκας Φώτιος

Λόκας Δημήτρης

MSc IoT – Smart Systems

7/1/2019

Dimitris Lokas

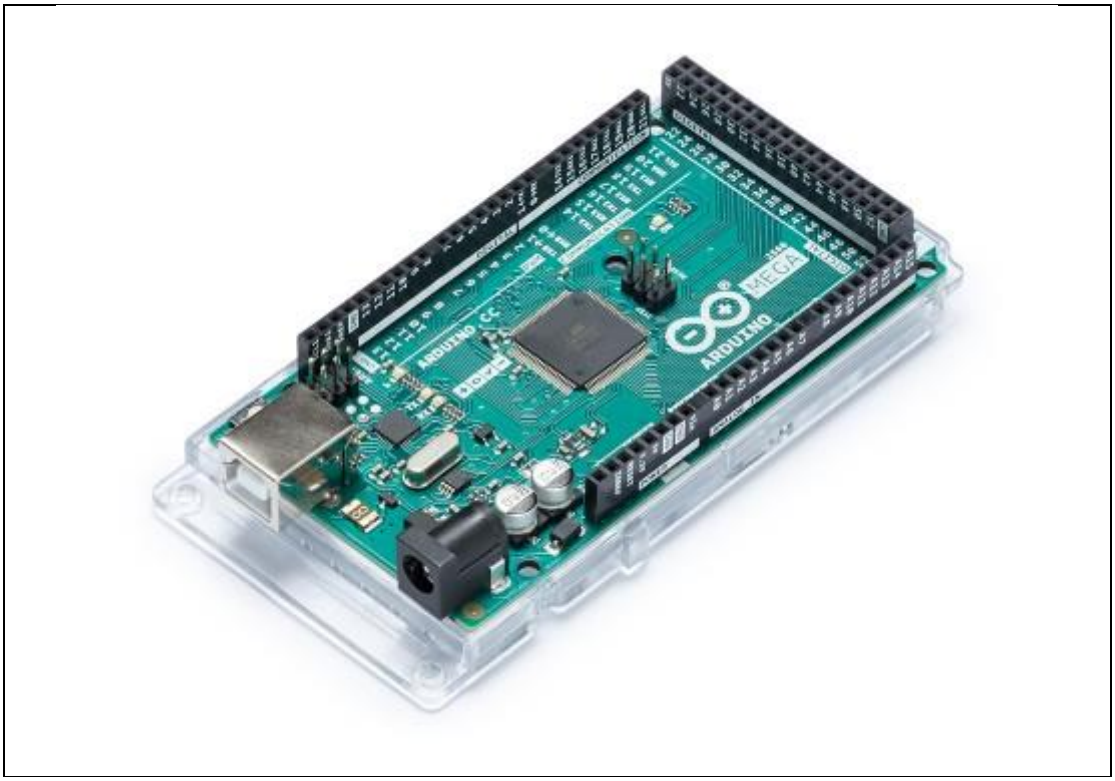
Σκοπός του παρόντος εγγράφου είναι η σύντομη περιγραφή – ανάλυση της λειτουργίας ενός απλού **μετεωρολογικού σταθμού IoT πλατφόρμας** αποτελούμενη από 1 Arduino με ένα αισθητήρα ψηφιακό θερμοκρασίας – υγρασίας DHT 22 και διαμεσολαβητή broker MQTT (HiveMQ) για publish των real – time μετρήσεων.

Στη συνέχεια θα κάνουμε συλλογή των αποτελεσμάτων μέσω της πλατφόρμας Node-RED που εκτελείται σε ένα Raspberry Pi ή Laptop μέσω subscribe στο broker και παρουσίαση γραφικά με διαγράμματα και ενδείξεις στην οθόνη των τιμών **θερμοκρασίας**, **υγρασίας** και **δείκτη θερμότητας σε oC** (Heat Index Value) σε πραγματικό χρόνο.

Dimitris Lokas

ARDUINO PLATFORM – DHT22 SENSOR

ARDUINO MEGA 2560

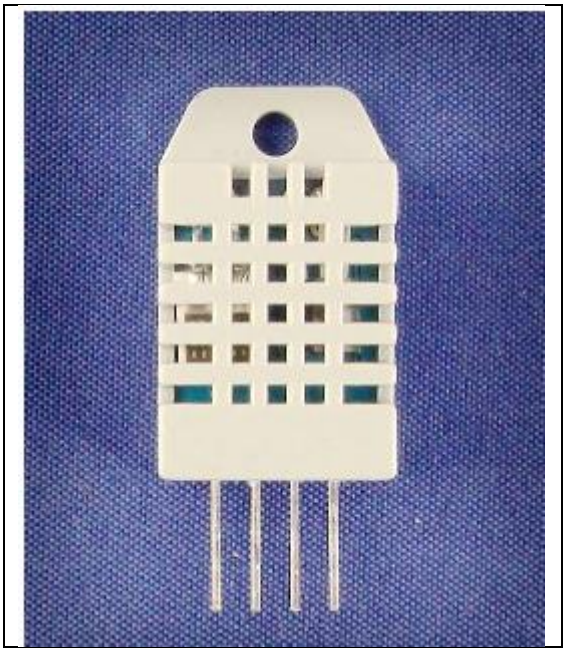


1.Τεχνικα Χαρακτηριστικά

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

DHT22 (AM2302)

Αισθητήρας σχετικής υγρασίας και θερμοκρασίας / μονάδας ψηφιακής εξόδου



1.**Χαρακτηριστικά - Εφαρμογή**:

- Πλήρης αντιστάθμιση θερμοκρασίας
- Σχετική υγρασία και μέτρηση θερμοκρασίας
- Ψηφιακό σήμα βαθμονομημένο
- Εξαιρετική μακροπρόθεσμη σταθερότητα
- Δεν χρειάζονται επιπλέον εξαρτήματα
- Μεγάλη απόσταση μετάδοσης
- Χαμηλή κατανάλωση ενέργειας
- 4 ακίδες συσκευασμένες και πλήρως εναλλάξιμες

2. **Περιγραφή**

Τα αισθητήρια στοιχεία του είναι συνδεδεμένα με ένα τσιπ 8 bit υπολογιστή. Κάθε αισθητήρας αυτού του μοντέλου βαθμονομείται σε ακριβή θάλαμο βαθμονόμησης και ο συντελεστής βαθμονόμησης αποθηκεύεται σε τύπο προγράμματος στη μνήμη OTP

Μικρό μέγεθος & χαμηλή κατανάλωση & μεγάλη απόσταση μετάδοσης (20m) επιτρέπουν στο DHT22 να ταιριάζει σε όλα τα είδη και σε σκληρές συνθήκες εφαρμογής.

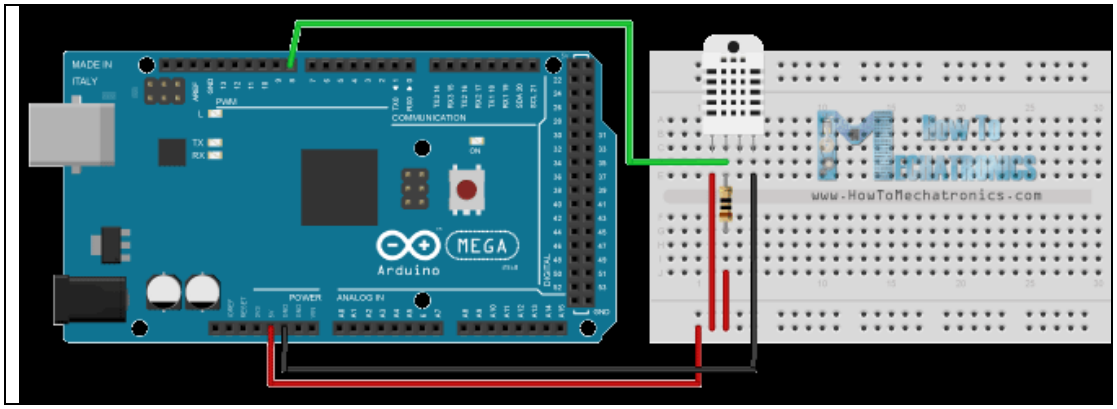
Βιβλιοθήκη λειτουργίας στο Arduino IDE <dht.h>

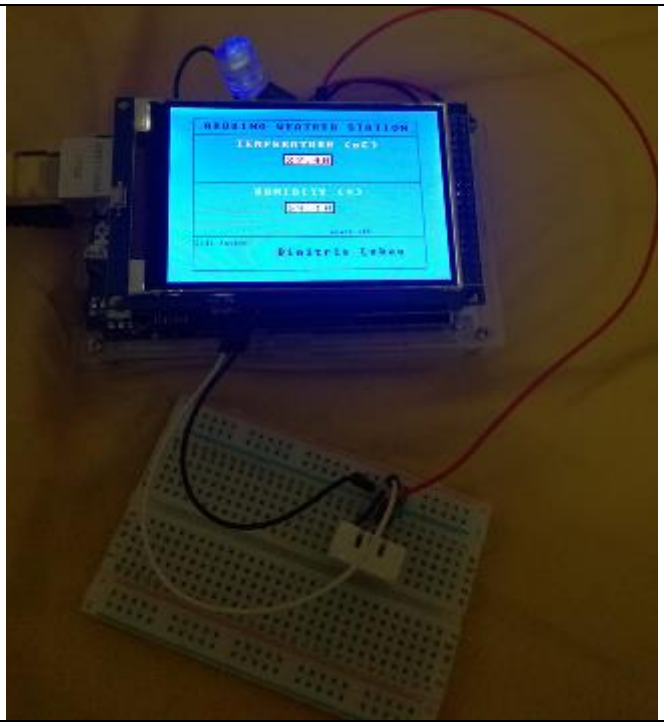
3.**Τεχνικα Χαρακτηριστικά**

Τροφοδοσία	3.3-6V
DC Ψηφιακό σήμα σήματος εξόδου μέσω ενός μόνο διαύλου Στοιχείο αισθητήρα Πυκνωτής πολυμερούς	
Ευρος Λειτουργίας	
Υγρασία 0-100% RH	θερμοκρασία -40 ~ 80Celsius
Ακρίβεια	
Υγρασία + -2% RH (Μέγιστη + -5% RH)	θερμοκρασία <+ - 0.5Celsius

Ανάλυση ή Ευαισθησία	
Υγρασία0,1% RH	θερμοκρασία 0.1Celsius
Επαναληψιμότητα	
Υγρασία + -1% RH	Θερμοκρασία + -0,2Celsius
Υστέρηση υγρασίας	+-0.3%RH
Μακροπρόθεσμη Σταθερότητα	+-0.5%RH/year
Περίοδος ανίχνευσης	≈ 2sec
Διαστάσεις	14*18*5.5mm

ΣΥΝΔΕΣΜΟΛΟΓΙΑ





Κώδικας σε Arduino IDE

arduino_weather_station_final-nodeRED | Arduino 1.8.5

Αρχείο Επεξεργασία Σχέδιο Εργαλεία Βοήθεια

arduino_weather_station_final-nodeRED §

```
//
//                                    SIMPLE DHT22 WEATHER STATION WITH 3.2 TFT IPS SCREEN READOUTS,ALARM BUZZER AND NODE RED INTERGRATION
//
//
//                                    Arduino Code Author : Dimitris Lokas Msc IoT and Electrical Engineering
//                                                                                          May 2018 Rev 1.2
//
//
//
#include <UTFT.h>                     //http://www.rinkydinkelectronics.com/resource/UTFT/UTFT.pdf
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#include <DHT.h>
```

Βιβλιοθήκες

```
#define YAXIS 0                       //Axis of the TFT sceen
#define XAXIS 1
#define DHTPIN 2                      // Temp and humidity sensor
#define DHTTYPE DHT22                 //Sensor type

UTFT myGLCD(CTE40,38,39,40,41);       // Initialize display
DHT dht(DHTPIN, DHTTYPE);
unsigned long readTime;
extern uint8_t BigFont[];             //Setting fonts for the LCD http://www.rinkydinkelectronics.com/r_fonts.php
extern uint8_t SmallFont[];
extern uint8_t SevenSegNumFont[];

const int buzzer = 12;                //buzzer to arduino pin 12
const int ledPin6 = 6;
const int ledPin7 = 7;                //Remote Led switch pin 7
```

Αρχικοποίηση

```
// Update these with values suitable for your network.
byte mac[] = { 0xAA, 0xBB, 0xCC, 0x22, 0x33, 0x81 };
IPAddress ip(192,168,1,20);
IPAddress server(3,121,19,92);
// IPAddress server([illegible]);
char message_buff[100]; // this buffers our incoming messages so we can do something on certain commands
```

IP arduino Shield
IP HiveMQ

```
char message_buff[100]; // this buffers our incoming messages so we can do something on certain commands

EthernetClient ethClient;
PubSubClient client(ethClient);

void callback(char* topic, byte* message, unsigned int length) {
Serial.print("Message arrived on topic: ");
Serial.print(topic);
Serial.print(". Message: ");
String messageTemp;
for (int i = 0; i < length; i++) {
Serial.print((char)message[i]);
messageTemp += (char)message[i];
}
Serial.println();
// Feel free to add more if statements to control more Pins with MQTT
// If a message is received on the topic home/livingroom/arduino/ledPin6
// It's a value between 0 and 255 to adjust the LED brightness
if(String(topic)=="hellion/LEDANLG"){
Serial.print("Changing Digital Pin 6 Brithness to ");
Serial.print(messageTemp);
analogWrite(ledPin6, messageTemp.toInt());
}
// If a message is received on the topic home/livingroom/arduino/ledPin7,
//you check if the message is either 1 or 0. Turns the Arduino Digital Pin according to the message
if(String(topic)=="hellion/LED"){
Serial.print("Changing Digital Pin 7 to ");
if(messageTemp == "1"){
digitalWrite(ledPin7, HIGH);
Serial.print("On");
}
else if(messageTemp == "0"){
digitalWrite(ledPin7, LOW);
Serial.print("Off");
}
}
```

```
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
if (client.connect("arduinoClient")) {
Serial.println("connected");
// Subscribe or resubscribe to a topic
// You can subscribe to more topics (to control more LEDs in this example)
client.subscribe("hellion/LED");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
// Wait 5 seconds before retrying
delay(5000);
}
}
}
```

Σειριακή θύρα στοιχεία επικοινωνιας αποτελέσματα

```
}

void setup()
{
pinMode(ledPin6, OUTPUT);
pinMode(ledPin7, OUTPUT);

Serial.begin(57600);
myGLCD.InitLCD();
myGLCD.clrScr();
myGLCD.setColor(255,255,255);
myGLCD.fillScr(VGA_YELLOW);
myGLCD.setFont(BigFont);
myGLCD.setBackColor(VGA_YELLOW);
myGLCD.print("ARDUINO WEATHER STATION", CENTER, 150);

pinMode(buzzer, OUTPUT);                // Set buzzer - pin 12 as an output
dht.begin();                            //Start the temp and humidity sensor
tone(buzzer, 2000);
delay(500);         // ...for 0.5 sec
noTone(buzzer);     // Stop sound...
delay(500);         // ...for 0.5 sec

client.setServer(server,1883); // can be replaced by: client.setServer(server, 1883);
client.setCallback(callback);

Ethernet.begin(mac,ip);

dht.begin();
// Allow the hardware to sort itself out
delay(1500);
Serial.println(Ethernet.localIP());
readTime = 0;
}
```

```
  delay(6000);
// Read Temperature as Celcius and Humidity
float h = dht.readHumidity();
float t = dht.readTemperature();
myGLCD.setFont(BigFont);
   myGLCD.fillScr(VGA_YELLOW);
   myGLCD.drawRect(197,86,282,106);
   myGLCD.drawRect(198,87,283,107);
   myGLCD.drawRect(197,176,282,196);
   myGLCD.drawRect(198,177,283,197);
   myGLCD.drawRoundRect(40,20,440,300);
   myGLCD.drawRoundRect(40,40,440,50);
   myGLCD.drawRoundRect(40,40,440,140);
   myGLCD.drawRoundRect(40,40,440,240);
   myGLCD.setBackColor(VGA_YELLOW);
   myGLCD.print("ARDUINO WEATHER STATION",  CENTER,  25);
   myGLCD.setColor(0,155,155);
   myGLCD.print("TEMPERATURE (oC)",  CENTER,  60);
   myGLCD.setColor(255,255,255);
   myGLCD.setBackColor(0,155,155);
   myGLCD.printNumF(t,2,CENTER,90);
   myGLCD.setBackColor(VGA_YELLOW);
   myGLCD.setColor(255,0,0);
   myGLCD.print("HUMIDITY (%)",  CENTER,  150);
   myGLCD.setColor(255,255,255);
   myGLCD.setBackColor(255,0,0);
   myGLCD.printNumF(h,2,CENTER,180);
   myGLCD.setBackColor(VGA_YELLOW);
   myGLCD.setFont(SmallFont);
   myGLCD.print("alarm off",275,225);
   myGLCD.print("Code Author:",45,245);
   myGLCD.setFont(BigFont);
   myGLCD.print("Dimitris Lokas",180,260);
if (!client.connected()) {
```

Προγραμματισμός Οθόνης

```
void sensorRead(){
readTime = millis();
// Reading temperature or humidity takes about 250 milliseconds!
// Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).
if (isnan(h) || isnan(t) || isnan(f)) {
Serial.println("Failed to read from DHT sensor!");
return;
}

// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);

char buffer[14];
dtostrf(t,5,2,buffer);
client.publish("hellion/temperature",buffer);
//Serial.println(buffer);
dtostrf(h,5,2,buffer);
client.publish("hellion/humidity",buffer);
dtostrf(hic,5,2,buffer);
client.publish("hellion/heatindex",buffer);

//client.publish("inTopic/humidity",sprintf(buf, "%f", h));
Serial.print("Humidity: ");
Serial.print(h);
```

Μεταβλητές Float
t : θερμοκρασίας σε oC
f : θερμοκρασίας σε oF
h : υγρασίας
hif : Heat Index σε oF
hic : Heat Index σε oC

Publish στο MQTT Broker στο Topic "hellion/temperature "hellion/humidity "hellion/heatindex

```
Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Heat index: ");
Serial.print(hic);
Serial.print(" *C ");
Serial.print(hif);
Serial.println(" *F");
}
```

Εμφάνιση Αποτελεσμάτων στην σειριακή Θύρα

MQTT - BROKER

(Message Queue Telemetry Transport 1999): Το MQTT πρωτόκολλο είναι ένα πιστοποιημένο κατά το ISO standard (ISO/IEC PRF 20922) πρωτόκολλο επικοινωνίας το οποίο αναπτύχθηκε από τους Andy Stanford-Clark (IBM) & Arlen Nipper (Eurotech) το 1999 για την παρακολούθηση πετρελαϊκού αγωγού.

Πλεονεκτεί ως προς την κατανάλωση ισχύος (battery savings) και την χρήση του bandwidth Καταλληλότερο για M2M IoT.

Η Αρχιτεκτονική του βασίζεται στην λειτουργία του Publish – Subscribe:

Publish

Δημοσίευση μηνυμάτων ενός συγκεκριμένου topic στον broker.

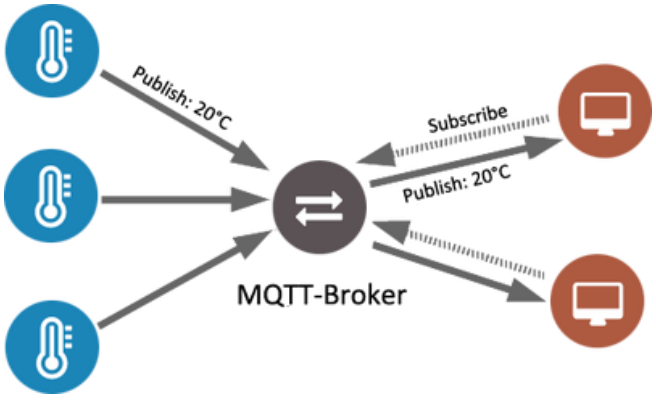
Subscribe

Εγγραφή σε συγκεκριμένο topic στον message broker.

Broker

Πρόγραμμα διαμεσολαβητή για τη διαχείριση μηνυμάτων μεταξύ ετερογενών messaging πρωτοκόλλων.

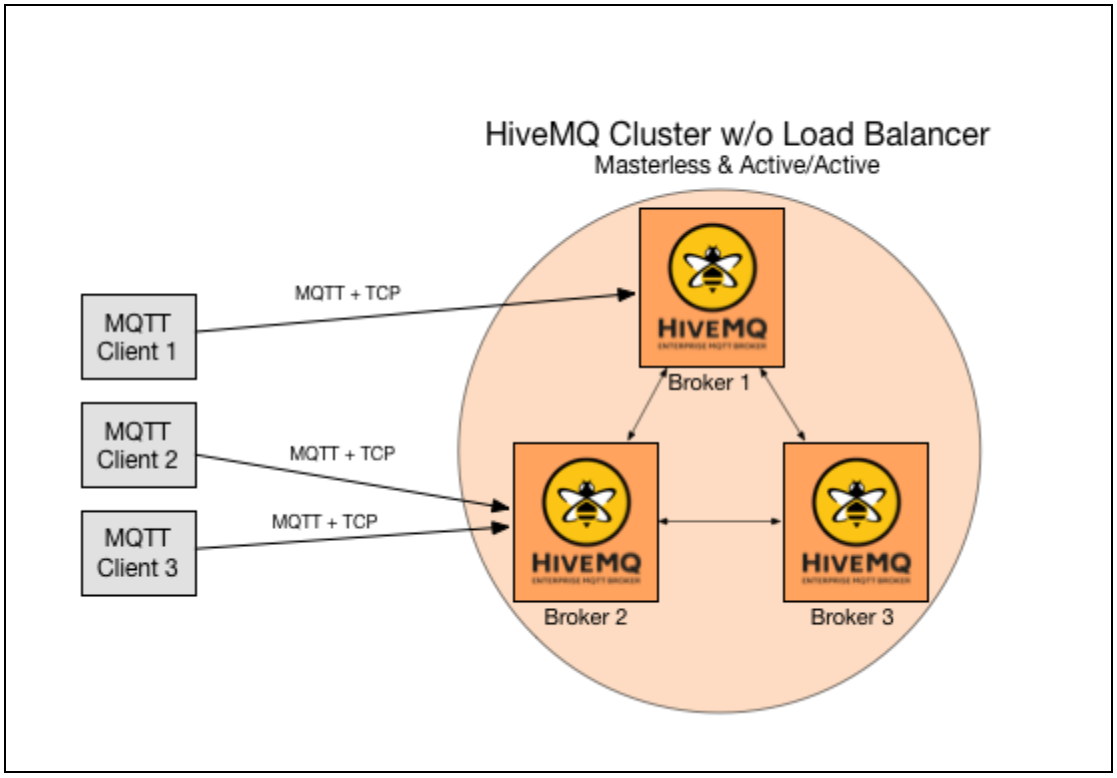
- *Επικαιροποίηση.*
- *Μετασχηματισμός.*
- *Δρομολόγηση.*



HiveMQ

Στο παρόν project θα γίνει χρήση της πλατφόρμας ***HiveMQ***, Το HiveMQ είναι η πλατφόρμα ανταλλαγής μηνυμάτων βασισμένη σε MQTT για γρήγορη, αποτελεσματική και αξιόπιστη μεταφορά δεδομένων προς και από τις συνδεδεμένες συσκευές IoT και τα συστήματα επιχειρήσεων.

Το πρόγραμμα αυτό εκτελείται σε JAVA VM σε ένα σμήνος υπολογιστών (3 ή 4) Server Nodes.



Η σύνδεση γίνεται μεσώ προγραμματισμού στην διεύθυνση host broker.hivemq.com στην πόρτα 1883.

Ο διαμεσολαβητής broker είναι υπεύθυνος για τη λήψη όλων των μηνυμάτων, το φιλτράρισμα των μηνυμάτων, τον προσδιορισμό του χρήστη που έχει εγγραφεί σε κάθε μήνυμα και την αποστολή του μηνύματος σε αυτούς τους εγγεγραμμένους πελάτες. Ο broker διατηρεί επίσης τις συνεδρίες όλων των ενεργών πελατών(clients), συμπεριλαμβανομένων των συνδρομών και των μηνυμάτων που χάθηκαν(missed messages). Μια άλλη ευθύνη του broker

είναι η πιστοποίηση και η εξουσιοδότηση των πελατών. Συνήθως, ο broker διευκολύνει την προσαρμοσμένη πιστοποίηση ταυτότητας, την εξουσιοδότηση και την ενσωμάτωση σε συστήματα backend. Η ενσωμάτωση είναι ιδιαίτερα σημαντική επειδή ο broker είναι συχνά το στοιχείο που είναι άμεσα εκτεθειμένο στο διαδίκτυο, χειρίζεται πολλούς πελάτες και χρειάζεται να διαβιβάσει μηνύματα σε συστήματα ανάλυσης και επεξεργασίας στα επόμενα στάδια.

Άρα στο παρόν project μέσω Arduino IDE με τις εντολές

char buffer[14];

dtostrf(t,5,2,buffer);

client.publish("hellion/temperature",buffer);

//Serial.println(buffer);

dtostrf(h,5,2,buffer);

client.publish("hellion/humidity",buffer);

dtostrf(hic,5,2,buffer);

client.publish("hellion/heatindex",buffer);

κάναμε Publish στα Topic hellion/temperature , hellion/humidity , hellion/heatindex τις τιμές των αντίστοιχων αισθητήρων. Στη συνέχεια θα κάνουμε χρήση της πλατφόρμας NODE-RED για συλλογή των τιμών αυτών από τον broker και εμφάνιση σε διαγράμματα κλπ. στην οθόνη.

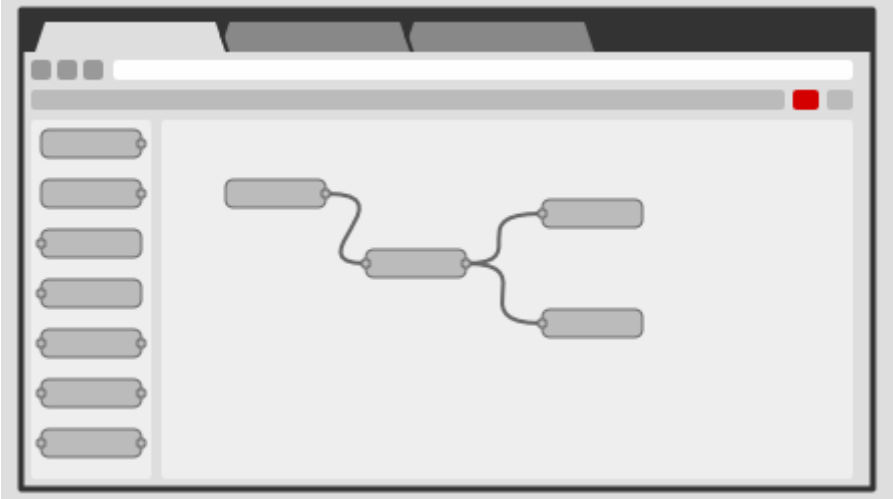
NODE-RED

Το Node-RED είναι ένα αναπτυξιακό εργαλείο που βασίζεται στην ροή για οπτικό προγραμματισμό και αναπτύχθηκε αρχικά από την IBM για τη διασύνδεση συσκευών υλικού, APIs και online υπηρεσιών ως μέρος του Διαδικτύου των Πραγμάτων.

Παρέχει έναν editor που βασίζεται σε πρόγραμμα περιήγησης, ο οποίος διευκολύνει τη σύνδεση των ροών χρησιμοποιώντας το ευρύ φάσμα των κόμβων (nodes) της παλέτας που μπορούν να αναπτυχθούν στο χρόνο εκτέλεσης με ένα μόνο κλικ.

Οι λειτουργίες JavaScript μπορούν να δημιουργηθούν μέσα στο πρόγραμμα επεξεργασίας χρησιμοποιώντας έναν editor εμπλουτισμένου κειμένου.

Μια ενσωματωμένη βιβλιοθήκη επιτρέπει να αποθηκεύονται χρήσιμες λειτουργίες, πρότυπα ή ροές για επαναχρησιμοποίηση.



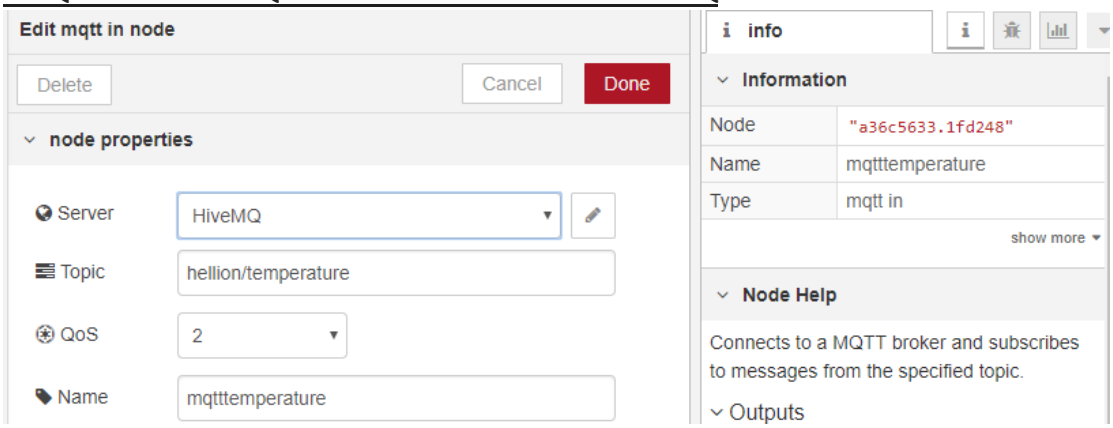
Ο ελαφρύς χρόνος εκτέλεσης είναι βασισμένος στο Node.js, εκμεταλλευόμενος πλήρως το πρότυπο που βασίζεται σε γεγονότα(events) και δεν μπλοκάρει. Αυτό το καθιστά ιδανικό για να εκτελείται στην άκρη του δικτύου σε χαμηλού κόστους υλικό, όπως το Raspberry Pi καθώς και στο νέφος (Cloud).

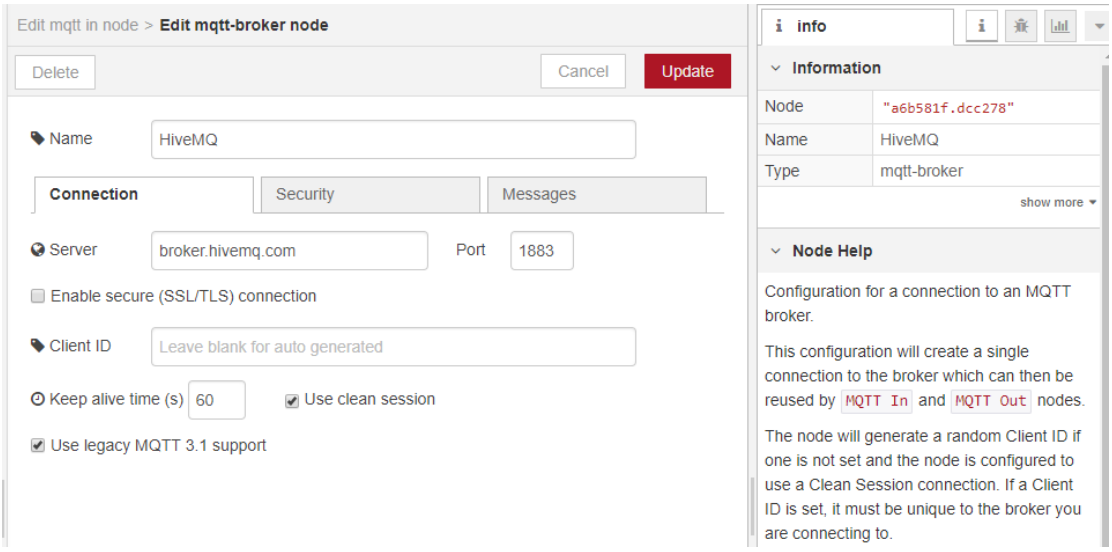
Οι ροές που δημιουργούνται στο Node-RED αποθηκεύονται χρησιμοποιώντας το JSON, το οποίο μπορεί εύκολα να εισαχθεί και να εξαχθεί για κοινή χρήση με άλλους.

Εκτελούμε το αρχείο node red.js και από την κονσόλα έχουμε εκτέλεση της εφαρμογής στην διεύθυνση 127.0.0.1:1880

Στο παρών project κάναμε χρήση των nodes

MQTT IN ➔MQTT subscribe στο HiveMQ

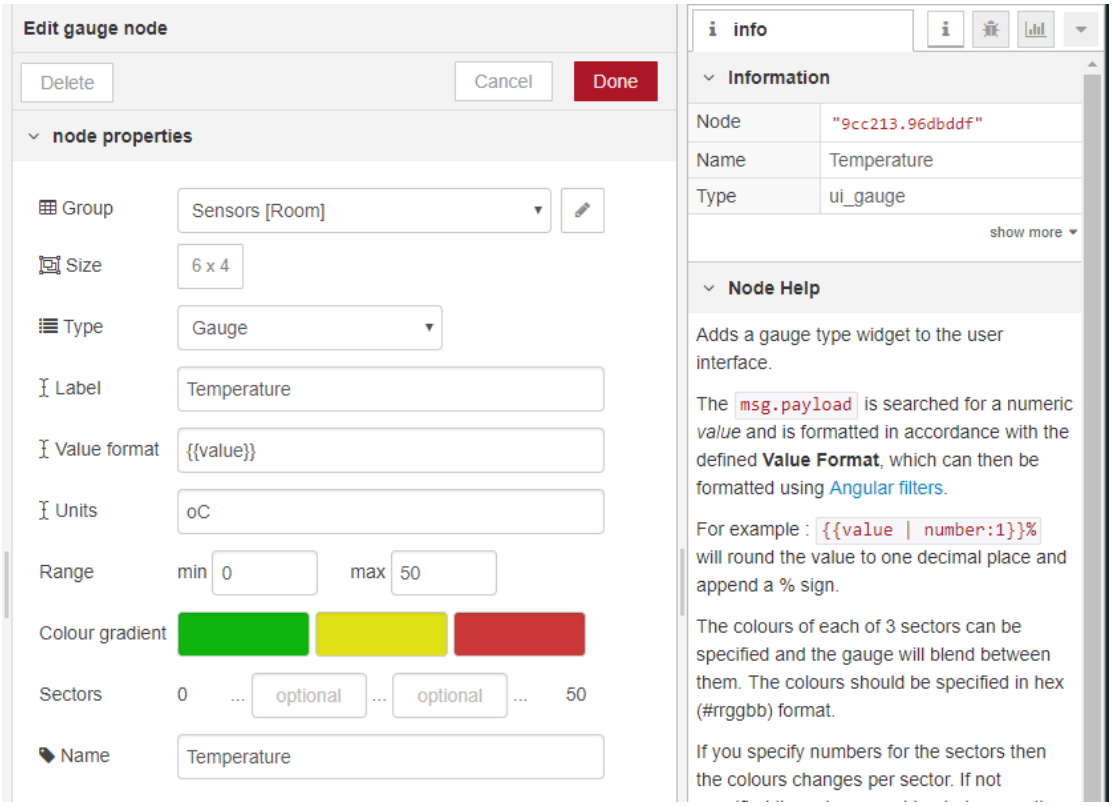




Function

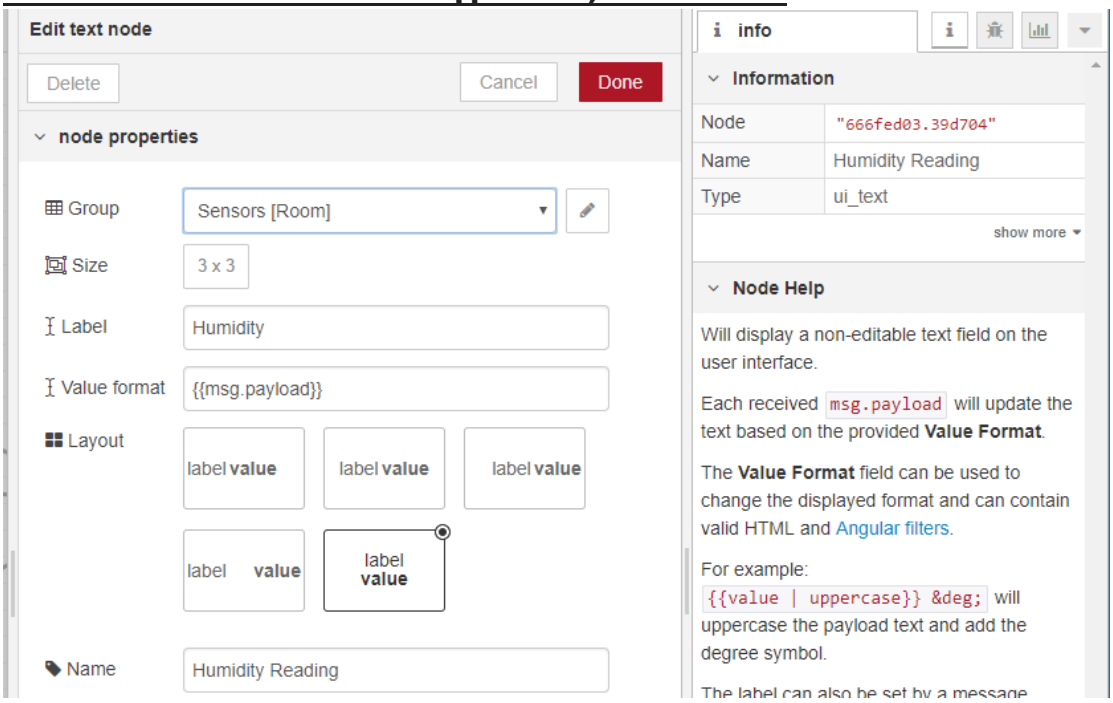
Dashboard text ➔ Για θερμοκρασία , υγρασία , heatindex

Dashboard gauge➔ Για θερμοκρασία



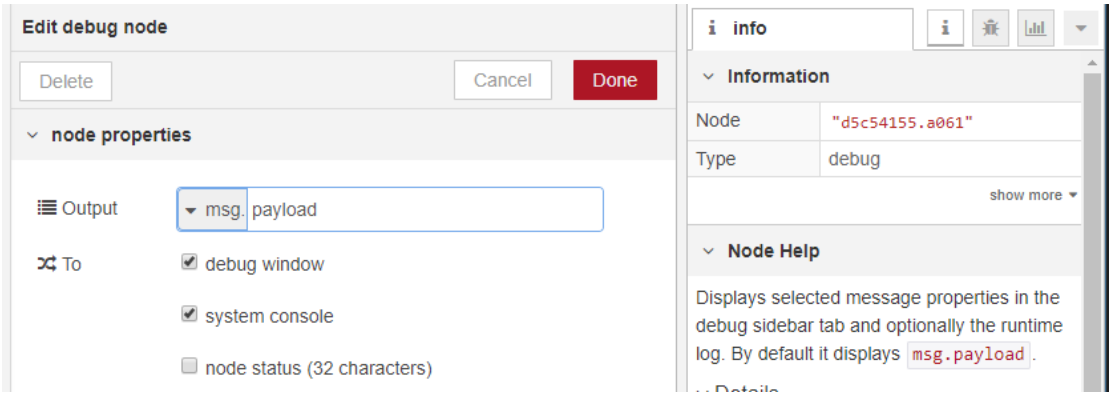
Επιλέγουμε διαστάσεις , χρώματα ανά πεδίο θερμοκρασιακών τιμών και labels.

Dashboard chart ➔Για υγρασία , heatindex

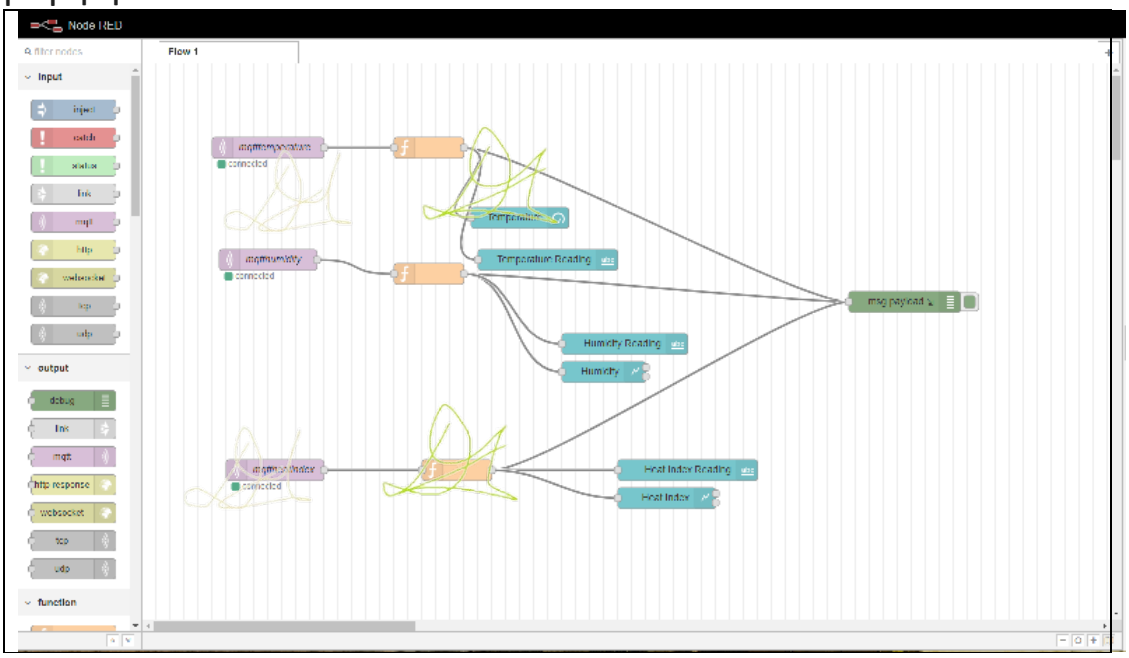


Επιλέγουμε διαστάσεις και διάταξη των τιμών label-value .

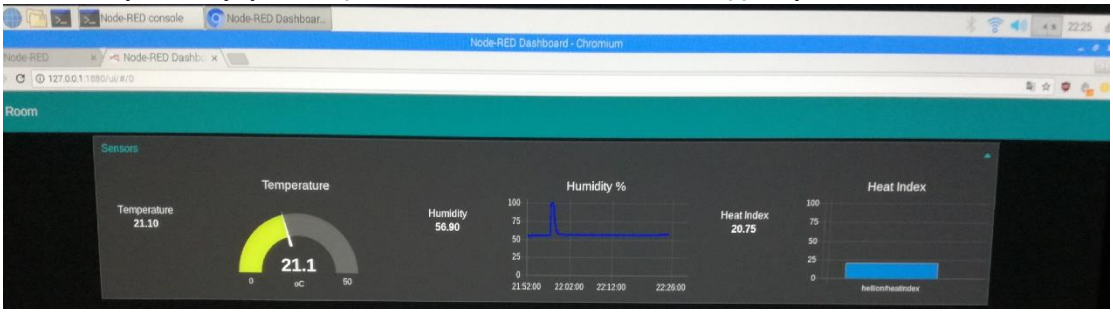
Output Debug ➔ msg.payload



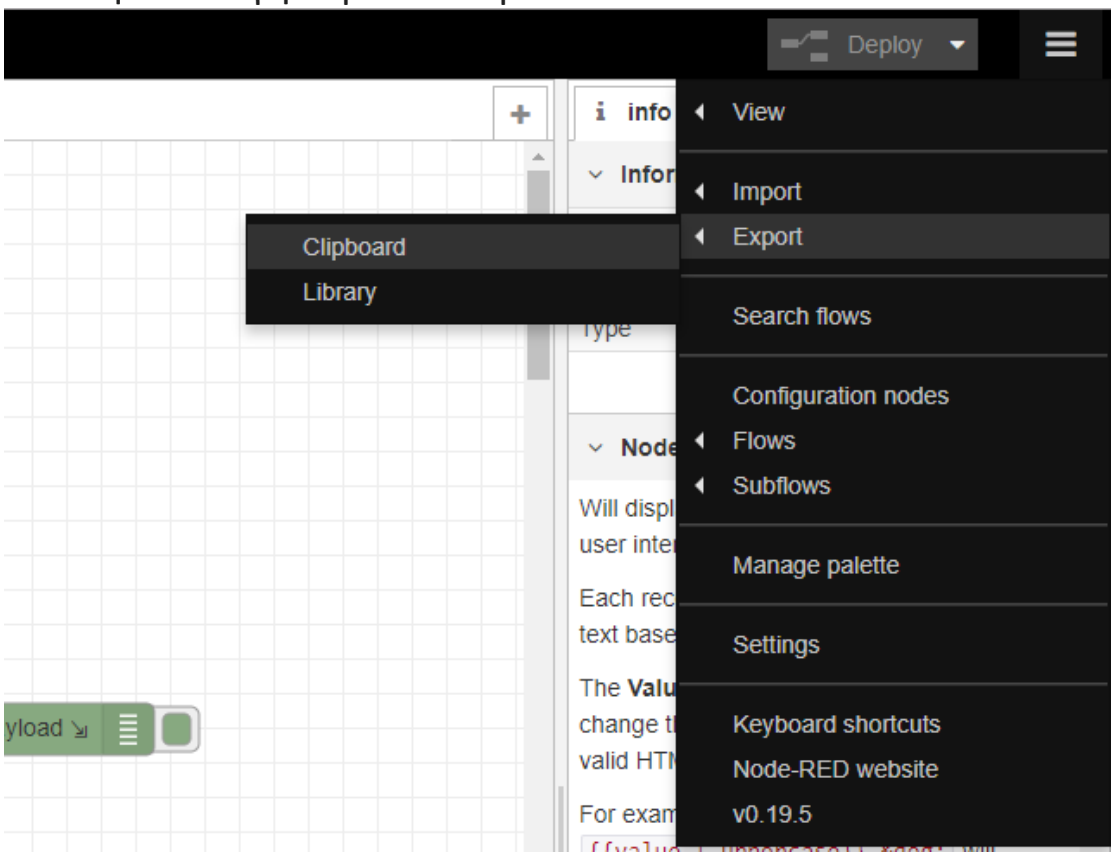
Επιλέγουμε εμφάνιση των τιμών debug σε παράθυρο αλλά και στην κονσόλα εκτέλεση της εφαρμογής.
Το συνολικό διάγραμμα ροής (**Flow Chart**) έχει την τελική μορφη.



Στο raspberry pi στην 127.0.0.1:1880/ui έχουμε το UI.



Από την επιλογή export➔clipboard



JSON FLOW CHART EXPORT

[{"id":"c7029f1b.ab6b","type":"mqtt
in","z":"f7137b41.4cf3c8","name":"mqtttemperature","topic":"
hellion/temperature","qos":"2","broker":"59460da6.016b44","
x":170,"y":180,"wires":[["399e6ce9.375644"]]},{"id":"399e6ce9
.375644","type":"function","z":"f7137b41.4cf3c8","name":"","f
unc":"\nreturn
msg;","outputs":1,"noerr":0,"x":400,"y":180,"wires":[["b20d22
65.a8bfa","f03223fe.d0d45","e103d86b.f1c318"]]},{"id":"f0322
3fe.d0d45","type":"debug","z":"f7137b41.4cf3c8","name":"","a
ctive":true,"tosidebar":true,"console":true,"tostatus":false,"co
mplete":"payload","x":1080,"y":400,"wires":[]},{"id":"b20d2265
.a8bfa","type":"ui_gauge","z":"f7137b41.4cf3c8","name":"Tem
perature","group":"61a7b138.666c","order":2,"width":"6","hei
ght":"4","gtype":"gage","title":"Temperature","label":"oC","for
mat":"{{value}}","min":0,"max":"50","colors":["#0db50d","#e0e
017","#ca3838"],"seg1":"","seg2":"","x":530,"y":280,"wires":[]},
{"id":"2f6f1eaa.98d292","type":"mqtt
in","z":"f7137b41.4cf3c8","name":"mqtthumidity","topic":"hell
ion/humidity","qos":"2","broker":"59460da6.016b44","x":170,"
y":340,"wires":[["95e17043.ddaf4"]]},{"id":"95e17043.ddaf4","t
ype":"function","z":"f7137b41.4cf3c8","name":"","func":"\nret
urn
msg;","outputs":1,"noerr":0,"x":400,"y":360,"wires":[["70c86f4
e.03587","f03223fe.d0d45","8b17b8ca.0081e8"]]},{"id":"70c86
f4e.03587","type":"ui_chart","z":"f7137b41.4cf3c8","name":"H
umidity
","group":"61a7b138.666c","order":4,"width":"6","height":"4",
"label":"Humidity
%","chartType":"line","legend":"false","xformat":"HH:mm:ss","i
nterpolate":"linear","nodata":"","dot":false,"ymin":"0","ymax":
"100","removeOlder":1,"removeOlderPoints":"","removeOlder

Unit":"3600","cutout":0,"useOneColor":false,"colors":["#1f34b4","#aec7e8","#ff7f0e","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,"x":650,"y":500,"wires":[[],[]]},{"id":"fecc8bee.0355e8","type":"function","z":"f7137b41.4cf3c8","name":"","func":"\nreturn msg;","outputs":1,"noerr":0,"x":440,"y":640,"wires":[["f03223fe.d0d45","ebb8c7b4.e47018","26bdfc42.4a48f4"]]},{"id":"a1d50d10.5070d","type":"mqtt in","z":"f7137b41.4cf3c8","name":"mqttheatindex","topic":"hellion/heatindex","qos":"2","broker":"59460da6.016b44","x":180,"y":640,"wires":[["fecc8bee.0355e8"]]},{"id":"e103d86b.f1c318","type":"ui_text","z":"f7137b41.4cf3c8","group":"61a7b138.666c","order":1,"width":"3","height":"3","name":"Temperature Reading","label":"Temperature","format":"{{msg.payload}}","layout":"col-center","x":570,"y":340,"wires":[]},{"id":"8b17b8ca.0081e8","type":"ui_text","z":"f7137b41.4cf3c8","group":"61a7b138.666c","order":3,"width":"3","height":"3","name":"Humidity Reading","label":"Humidity","format":"{{msg.payload}}","layout":"col-center","x":680,"y":460,"wires":[]},{"id":"26bdfc42.4a48f4","type":"ui_chart","z":"f7137b41.4cf3c8","name":"Heat Index","group":"61a7b138.666c","order":6,"width":"6","height":"4","label":"Heat Index","chartType":"bar","legend":"false","xformat":"HH:mm:ss","interpolate":"linear","nodata":"","dot":false,"ymin":"0","ymax":"100","removeOlder":1,"removeOlderPoints":"","removeOlderUnit":"3600","cutout":0,"useOneColor":false,"colors":["#1f77b4","#aec7e8","#e4a413","#2ca02c","#98df8a","#d62728","#ff9896","#9467bd","#c5b0d5"],"useOldStyle":false,"x":740,"y":680,"wires":[[],[]]},{"id":"ebb8c7b4.e47018","type":"ui_text","z":"f7137b41.4cf3c8","group":"61a7b138.666c","order":5,"width":"3","height":"3","name":"Heat Index Reading","label":"Heat

Index","format":"{{msg.payload}}","layout":"col-
center","x":770,"y":640,"wires":[]},{"id":"59460da6.016b44","t
ype":"mqtt-
broker","z":"","name":"HiveMQ","broker":"broker.hivemq.com
","port":"1883","clientid":"","usetls":false,"compatmode":true,
"keepalive":"60","cleansession":true,"birthTopic":"","birthQos"
:"0","birthPayload":"","closeTopic":"","closeQos":"0","closePayl
oad":"","willTopic":"","willQos":"0","willPayload":""},{"id":"61a
7b138.666c","type":"ui_group","z":"","name":"Sensors","tab":"
83ac1c.7c6913e8","disp":true,"width":"27","collapse":true},{"i
d":"83ac1c.7c6913e8","type":"ui_tab","z":"","name":"Room","i
con":"dashboard","order":2}]

Dimitris Lokas