

Textul și imaginile din acest document sunt licențiate

Attribution-NonCommercial-NoDerivs
CC BY-NC-ND



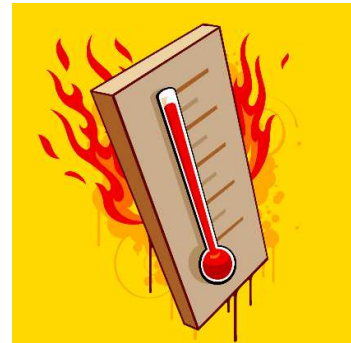
Codul sursă din acest document este licențiat

Public-Domain

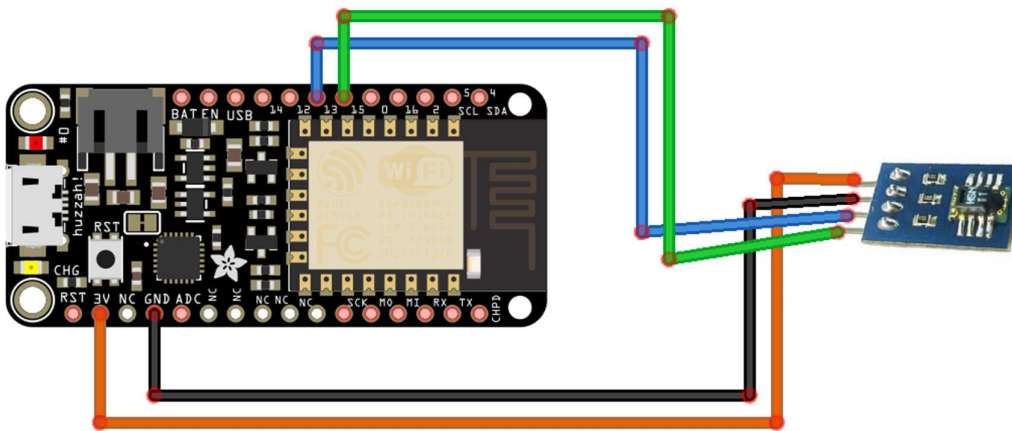
Ești liber să distribui acest document prin orice mijloace consideri (email, publicare pe website / blog, tipărire, sau orice alt mijloc), atât timp cât nu aduci nici un fel de modificări acestuia. Codul sursă din acest document poate fi utilizat în orice fel de scop, de natură comercială sau nu, fără nici un fel de limitări dar autorii nu își asumă nici o răspundere pentru pagubele pricinuite de implementările realizate de utilizatori. Schemele și codul sursă au un rol educativ și nu sunt gândite pentru a fi utilizate în mediu de producție (industrial, casnic sau comercial).

Hunting the Heat

Măsurarea temperaturii este o operație simplă de achiziție din punct de vedere al unui sistem electronic. Monitorizarea temperaturii însă ridică câteva provocări cu privire la înregistrarea datelor și vizualizarea acestora. În cadrul lecției de față vom da exemplu de două soluții de monitorizare a temperaturii prin intermediul unui serviciu de cloud.

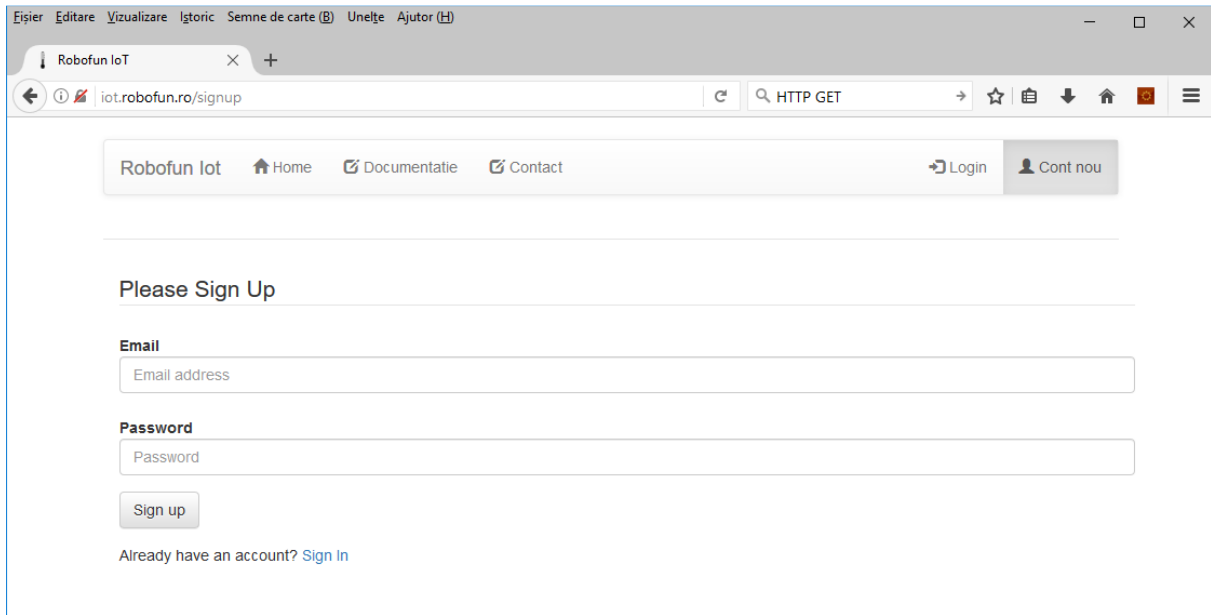


Pentru trimiterea datelor în cloud vom utiliza placa de dezvoltare Adafruit Feather HUZZAH (1) bazată pe microprocesorul WiFi ESP8266. Prima variantă de monitorizare va utiliza sensorul digital de temperatură și umiditate SHT11 (2) – o soluție de precizie recomandată când dorim să monitorizăm temperatura din aer (ambientală). Din punct de vedere al confortului termic monitorizarea temperaturii trebuie însoțită de monitorizarea umidității din aer. Schema de interconectare dintre placa de dezvoltare și sensor este următoarea:



Senzorul se va alimenta la 3.3V iar liniile de comunicație se vor conecta la pinii 13 (pinul DAT al senzorului) și 15 (pinul SCK al senzorului) ai plăci de dezvoltare. Pentru integrarea plăcii de dezvoltare în mediul Arduino IDE este necesară parcurgerea materialului „Adafruit Feather HUZZAH - WiFi with built-in battery charging for IoT on-the-go!” (3).

Ca serviciu de monitorizare în cloud vom utiliza serviciul Robofun IoT (4). Pentru utilizarea acestuia este necesară înregistrarea gratuită.



The screenshot shows a web browser window with the URL `iot.robofun.ro/signup`. The page has a navigation bar with links for Home, Documentatie, and Contact, along with Login and Cont nou buttons. The main content area is titled "Please Sign Up" and contains two input fields: "Email" (labeled "Email address") and "Password" (labeled "Password"). Below these fields is a "Sign up" button and a link that says "Already have an account? Sign in".

După înregistrare și conectare este necesară definirea a doi noi senzori (*Adauga senzor*) pentru a putea înregistra cele două valori măsurate (temperatură și umiditate).



The screenshot shows a web browser window with the URL `iot.robofun.ro/senzori`. A modal dialog box titled "Senzor" is open, featuring a single text input field and "Cancel" and "OK" buttons at the bottom. In the background, a green button labeled "Adauga senzor" is visible on the page.

După definirea celor doi senzori este necesar să copiem cheile de autentificare (*Token*) pentru a le utiliza în program.



The screenshot shows a web interface for generating a token. It has a red header bar with the word "Token". Below the header, there is a paragraph of text explaining that the token is a unique string of characters used for GET requests. A text box displays a token: `2f..vf..q2ef..0..2L..7q9C..j1c~9`. At the bottom, there is a red button labeled "Genereaza un nou token pentru acest senzor".

Programul a fost dezvoltat și testat utilizând Arduino IDE 1.8.3 cu extensia esp8266 2.3.0 instalată și biblioteca SHT1x (5) commit be7042c. În cadrul programului trebuie personalizate datele de conectare WiFi (*ssid* și *pass*) precum și cheile de autentificare oferite de procesul de înregistrare a senzorilor pe platforma Robofun IoT (4) (*SENSOR_TOKEN1* și *SENSOR_TOKEN2*).

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

char ssid[] = "...";
char pass[] = "...";
WiFiClient client;

#include <SHT1x.h>
#define dataPin 13
#define clockPin 15
SHT1x sht1x(dataPin, clockPin);

void setup() {
  Serial.begin(9600);
  delay(10);
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
```

```

    Serial.println(WiFi.localIP());
}

unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 60L * 1000L;

void loop() {
    if (millis() - lastConnectionTime > postingInterval) {
        IoTpublish();
    }
}

void IoTpublish() {
    String SENSOR_TOKEN1="...";
    String SENSOR_TOKEN2="...";
    float temp_c;
    float humidity;

    temp_c = sht1x.readTemperatureC();
    humidity = sht1x.readHumidity();

    Serial.print("Temperature: ");
    Serial.print(temp_c, DEC);
    Serial.print("C ");
    Serial.print(" Humidity: ");
    Serial.print(humidity);
    Serial.println("%");

    HTTPClient http;
    String data =
        String("http://iot.robofun.ro/api/v1/input/") +
        SENSOR_TOKEN1 + "/" + String(temp_c, DEC);

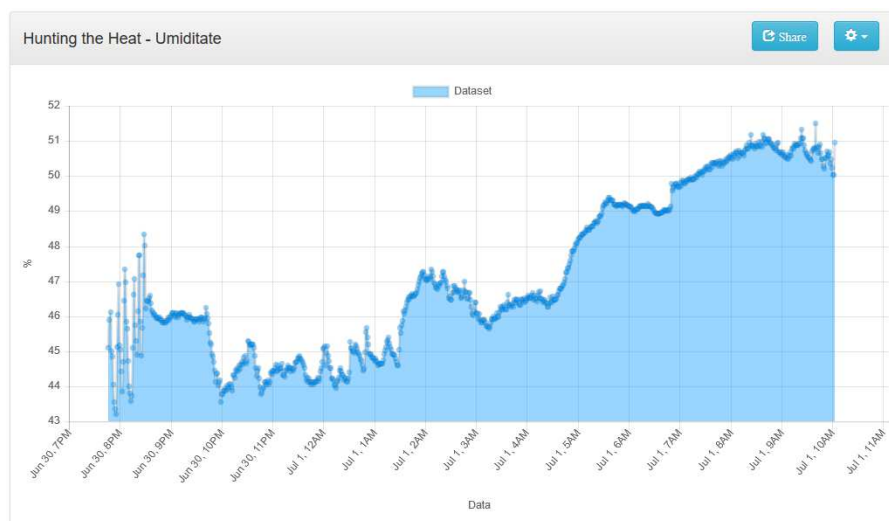
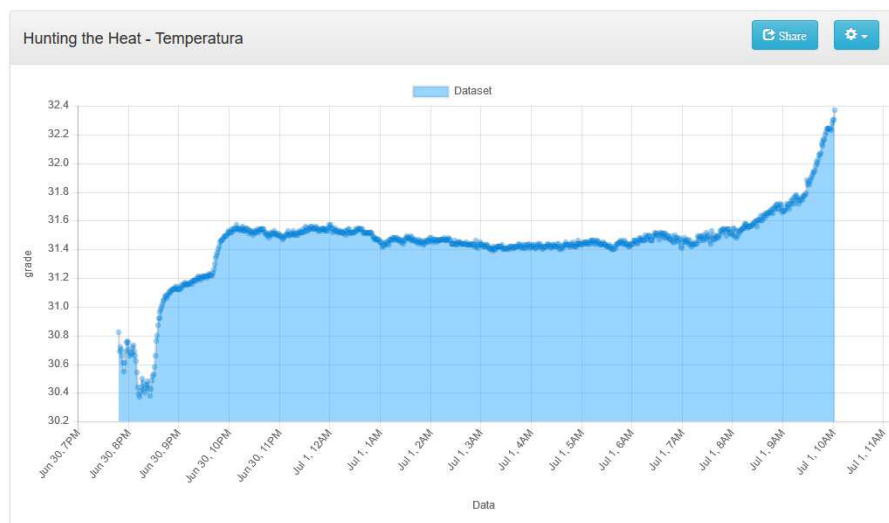
```

```

http.begin(data);
int httpCode = http.GET();
http.end();
data = String("http://iot.robofun.ro/api/v1/input/") +
        SENSOR_TOKEN2 + "/" + String(humidity, DEC);
http.begin(data);
httpCode = http.GET();
http.end();
lastConnectionTime = millis();
}

```

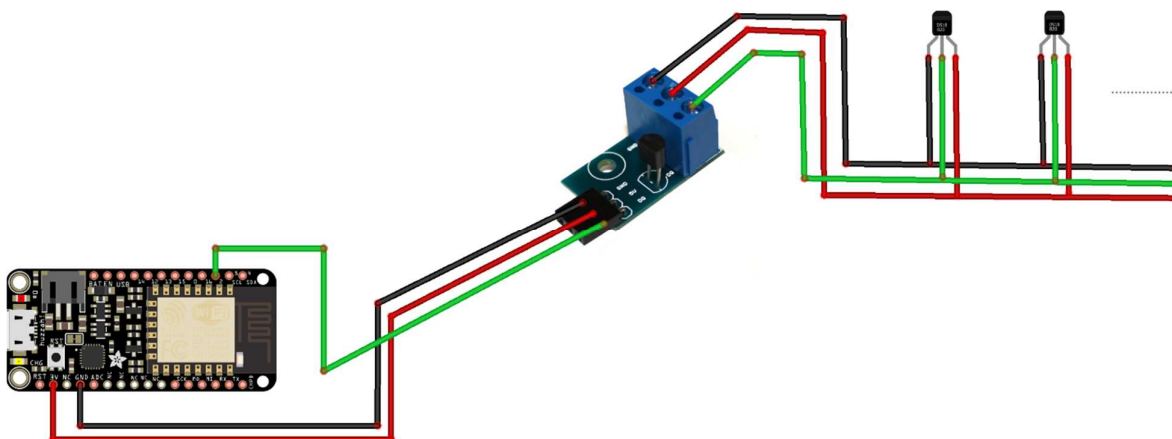
După încărcarea programului și punerea în funcțiune a sistemului putem monitoriza on-line valorile măsurate:



Cea de a doua variantă propusă pentru monitorizarea temperaturii este specifică monitorizării temperaturii materialelor și spațiilor de echipamente. Pentru măsurarea temperaturii vom utiliza senzorul digital DS18B20 ce permite realizarea unei rețele de senzori pe o magistrală cu un singur fir (1-Wire (6)) facilitând astfel supravegherea temperaturii în mai multe puncte. Vom utiliza un modul brick DS18B20 (7) ca prim element în rețeaua de senzori și mai mulți senzori DS18B20+ (8) pentru diverse puncte de supraveghere.

Soluția bazată pe comunicația 1-Wire a fost aleasă pentru distanța mare la care pot fi plasați senzorii față de placa de dezvoltare (aproximativ 200 de metri) și a unui număr mare de senzori care pot fi conectați simultan (având în vedere faptul că fiecare senzor are o adresă proprie din fabricație pe 64 de biți nu există o limitare de protocol privind numărul de senzori dintr-o rețea). A se vedea și: „Guidelines for Reliable Long Line 1-Wire Networks” (9).

Prin intermediul rețelei de senzori DS18B20 sistemul va putea monitoriza din punct de vedere al măsurării temperaturii un număr ”nelimitat” de zone. Conectarea cu placa de dezvoltare implică conectarea componentei brick în felul următor: pinul GND al modului la pinul de GND al plăcii de dezvoltare, pinul 5V la pinii de 3V al plăcii de dezvoltare (senzorii DS18B20 acceptă alimentare de la 3V la 5.5V) și pinul DQ al modului la pinul 3 al plăcii de dezvoltare. Următorii senzori atașați în rețea se vor conecta la modulul brick prin intermediul conectorului cu șurub respectând semnificația pinilor.



Programul a fost dezvoltat și testat utilizând Arduino IDE 1.8.3 cu extensia esp8266 2.3.0 instalată și bibliotecile OneWire 2.3.3 și DallasTemperature 3.7.6. În cadrul programului trebuie personalizate datele de conectare WiFi (*ssid* și *pass*) precum și cheile de autentificare oferite de procesul de înregistrare a senzorilor pe platforma Robofun IoT (4) (vectorul *SENSOR_TOKEN*).

```
#include <ESP8266WiFi.h>
```

```

#include <ESP8266HTTPClient.h>
char ssid[] = "...";
char pass[] = "...";
WiFiClient client;

#include <DallasTemperature.h>
#include <OneWire.h>
#define ONE_WIRE_BUS 2
#define MAX_ATTACHED_DS18B20 10
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
int numSensors=2;

void setup() {
    sensors.begin();
    delay(100);
    Serial.begin(9600);
    delay(100);
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 60L * 1000L;

void loop() {
    if (millis() - lastConnectionTime > postingInterval) {
        IoTpublish();
    }
}

void IoTpublish() {
    String SENSOR_TOKEN[] = {"...", "..."};
    HTTPClient http;

```



```

Serial.print("Requesting temperatures...");
sensors.requestTemperatures();
Serial.println("DONE");
for (int i=0; i<numSensors && i<MAX_ATTACHED_DS18B20;
      i++) {
    float temperature = sensors.getTempCByIndex(i);
    Serial.print("Temperature for the device ");
    Serial.print(i); Serial.print(" is: ");
    Serial.println(temperature);
    String data =
        String("http://iot.robofun.ro/api/v1/input/") +
        SENSOR_TOKEN[i] + "/" + String(temperature, DEC);
    http.begin(data);
    int httpCode = http.GET();
    if(httpCode > 0) {
        Serial.printf("[HTTP] GET... code: %d\n", httpCode);
        if(httpCode == HTTP_CODE_OK) {
            String payload = http.getString();
            Serial.println(payload);
        }
        else {
            Serial.printf("[HTTP] GET... failed, error: %s\n",
                http.errorToString(httpCode).c_str());
        }
    }
    http.end();
}
lastConnectionTime = millis();
}

```

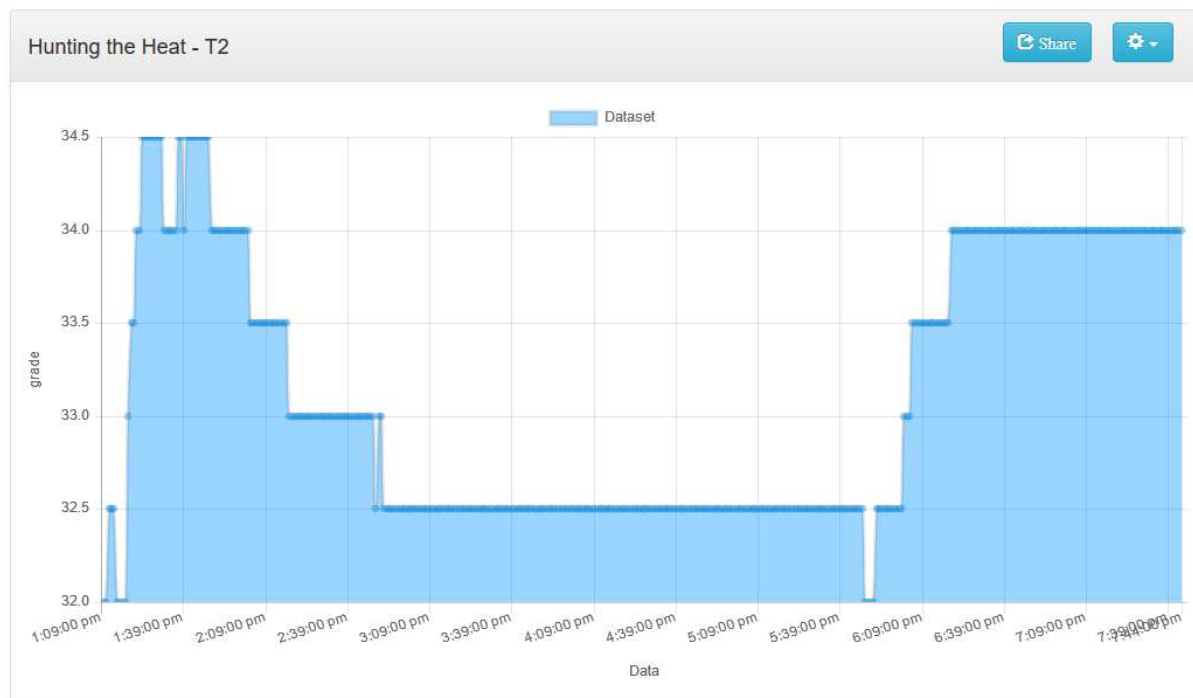
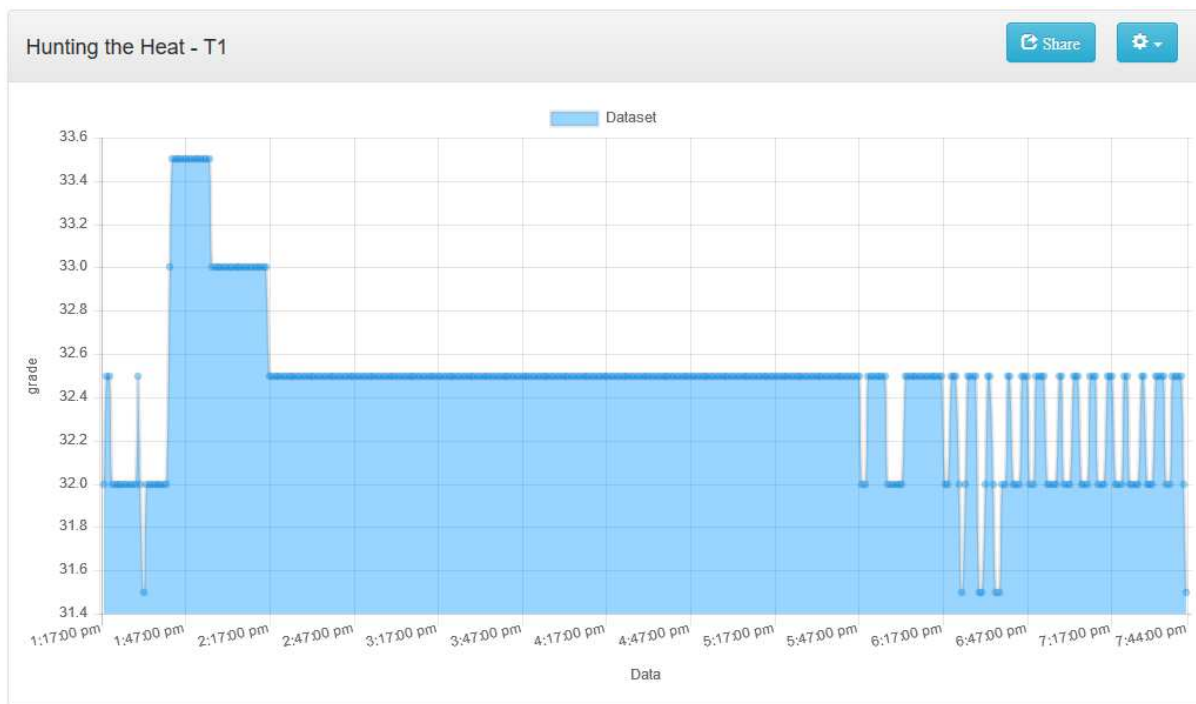
Programul va prezenta în consolă desfășurarea procesului de achiziții și înregistrare cloud (exemplificare cu doi senzori conectați):

```

COM23
1;
Requesting temperatures...DONE
Temperature for the device 0 is: 32.00
[HTTP] GET... code: 200
1;
Temperature for the device 1 is: 34.50
[HTTP] GET... code: 200
1;
Requesting temperatures...DONE
Temperature for the device 0 is: 32.00
[HTTP] GET... code: 200
1;
Temperature for the device 1 is: 34.50
[HTTP] GET... code: 200
1;
Requesting temperatures...DONE
Temperature for the device 0 is: 32.00
[HTTP] GET... code: 200
1;
Temperature for the device 1 is: 34.50
[HTTP] GET... code: 200
1;

```

dar monitorizarea se va face online:



Referințe on-line

(1) Feather HUZZAH cu ESP8266 WiFi

https://www.robofun.ro/iot/adafruit-feather-huzzah-with-esp8266-wifi?utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(2) Senzor de temperatura si umiditate SHT11

https://www.robofun.ro/senzori/vreme/senzor-temperatura-umiditate-sht11?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(3) Adafruit Feather HUZZAH - WiFi with built-in battery charging for IoT on-the-go!

<https://learn.adafruit.com/adafruit-feather-huzzah-esp8266>

(4) Robofun IoT

<http://iot.robofun.ro/>

(5) Arduino library to support SHT1x-series (SHT10, SHT11, SHT15) temperature / humidity sensors from Sensirion

<https://github.com/practicalarduino/SHT1x>

(6) 1-Wire - Maxim

<https://www.maximintegrated.com/en/products/digital/one-wire.html>

(7) Senzor Temperatura Inlantuibil Brick (DS18B20) - Motherboard

https://www.robofun.ro/senzori/vreme/senzor-temperatura-inlantuibil-brick-ds18b20-motherboard?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(8) Senzor de Temperatura DS18B20+

https://www.robofun.ro/senzori/vreme/senzor-temperatura-ds18b20?utm_source=newsletter&utm_medium=email&utm_content=productLink&utm_campaign=CURS_EMAIL

(9) Guidelines for Reliable Long Line 1-Wire Networks

<http://www.maximintegrated.com/en/app-notes/index.mvp/id/148>