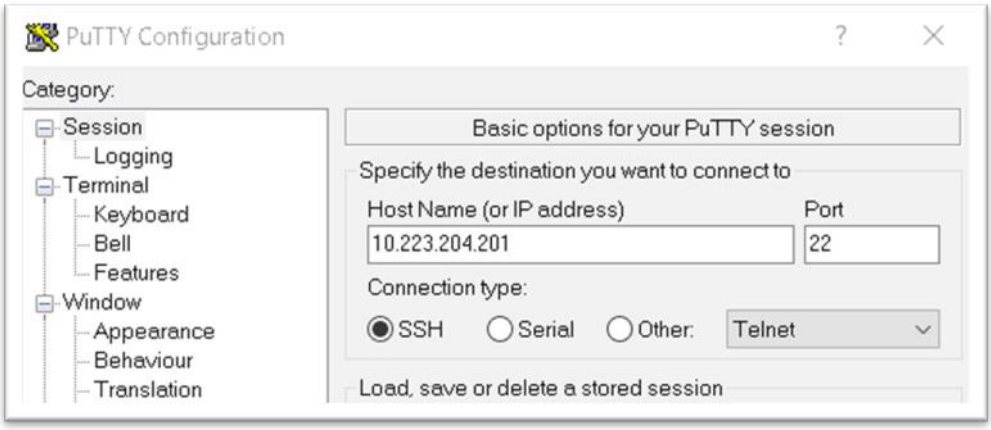
Smart_Gases_PRO_v30

IP adresa: 10.223.204.201

Port: 7800



Slika 1. Otvaranje sesije kroz PuTTy alat

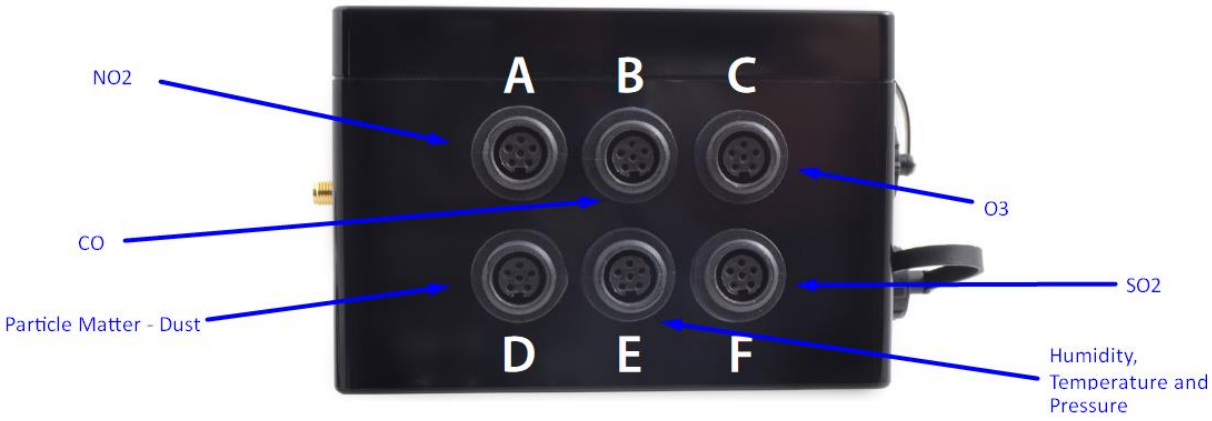
LORAWAN NETWORK SERVER

https://eu1.loriot.io/

user: nikola.zagorec@versoaltima.com
pass: pqGUZ4CeZ9PUf%

Libelium Smart Enviroment PRO

Slika 2. Libelium Smart Enviroment PRO senzor



Slika 3. Svi senzori na uređaju

Dodatne informacije

[Infomacije o uređaju i njegovim portovima]

[Vodič za programiranje senzora]

[Vodič za rad u programu Waspmote]

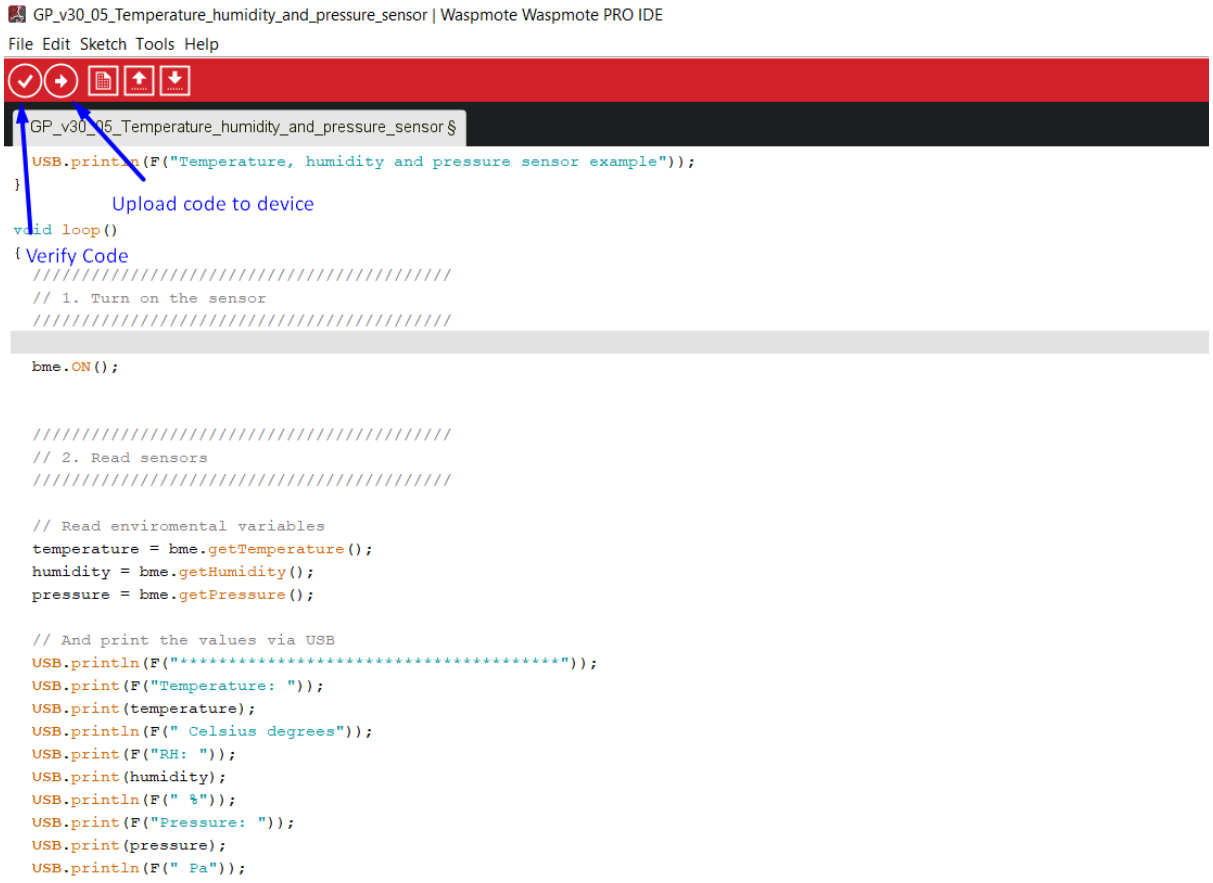
[Vodič za programiranje u Waspmoteu koristeći priložene module]

[Primjeri kodova u programu Waspmote]

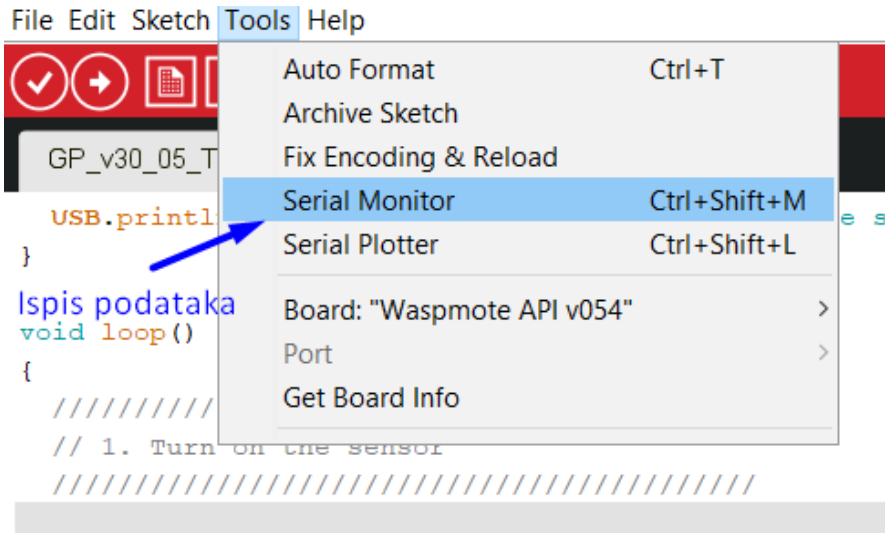
Korištenje programa WaspMote

Program WaspMote koristi se za pisanje firmware koda za uređaj „Libelium Smart Enviroment PRO“.

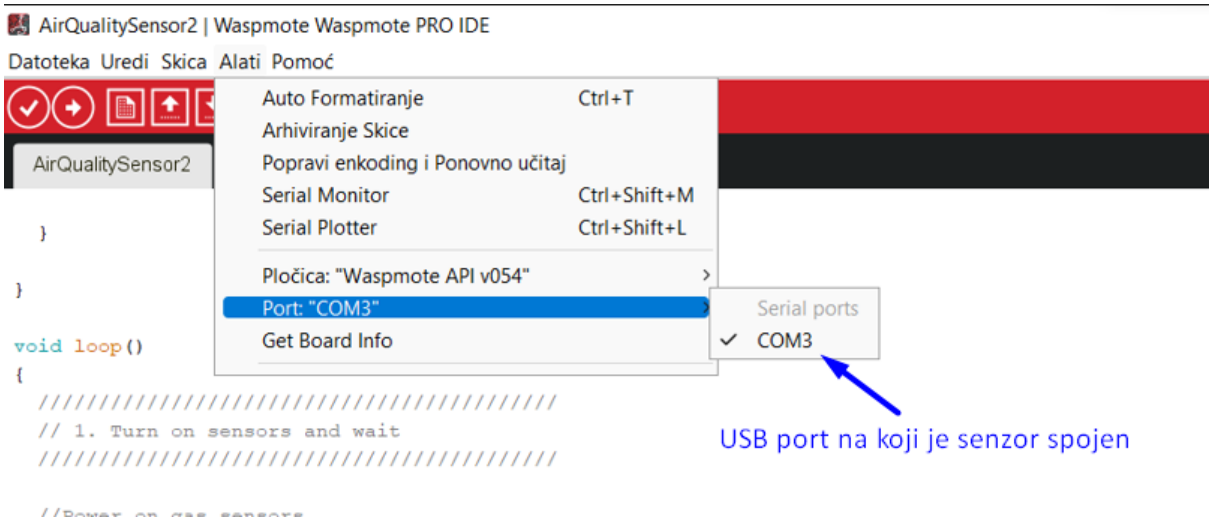
Firmware je baziran na C++-u, a i zaglavlja te moduli koje smo koristili su pisani u C++-u.



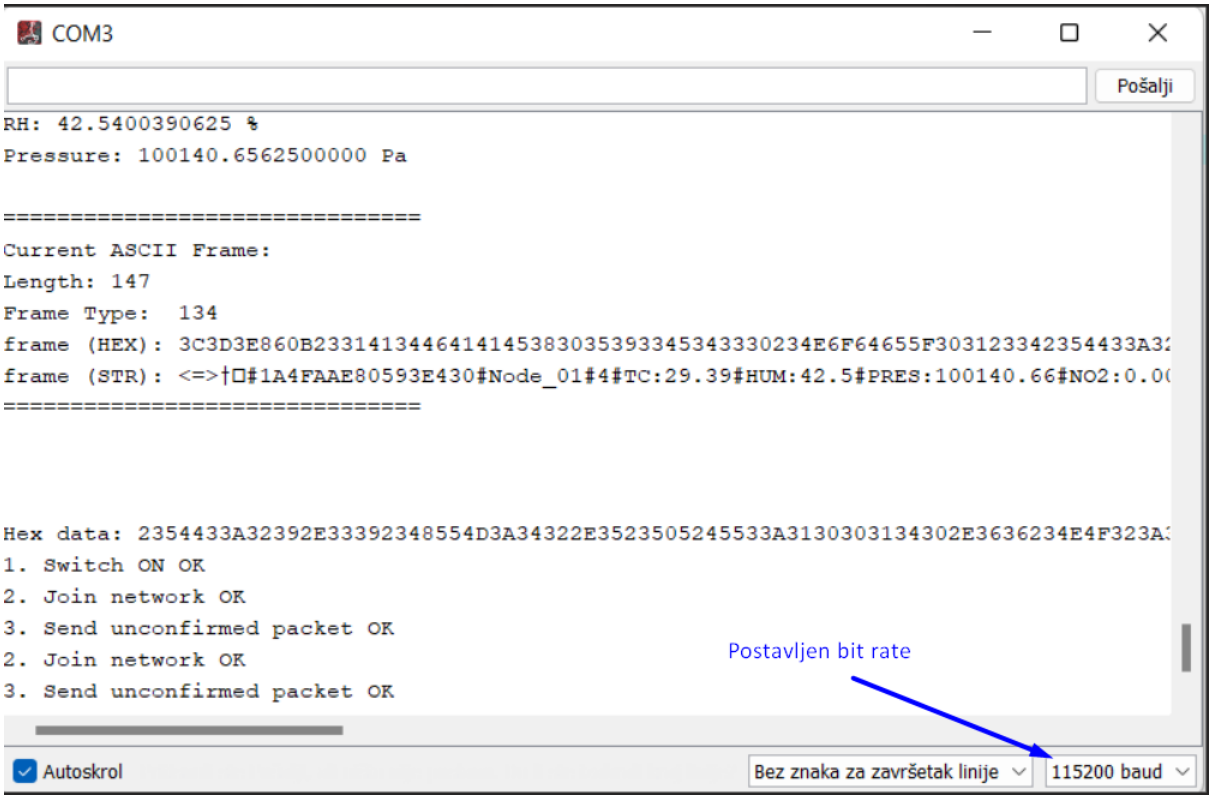
Slika 4. Funkcije programa, compile i upload



Slika 5. Ispis podataka u konzolu



Slika 6. USB port na koji je senzor spojen



Slika 7. Bit rate u komandnoj liniji

Kod programskog proizvoda

```
char DEVICE_EUI[]  = "0004A30B00212A2C"; ---> LoRaWAN EUI na uređaju

char APP_EUI[] = "0000000000000001"; ----> odabir pri Enrollanju uređaja na Loriotu

char APP_KEY[] = "00000000000000000000000000000000"; ---> “public key”
```

- Postavljanje varijable za ostvarivanje komunikacije

```
#include <WaspPM.h>  //Za PM – particle sensor

bmeGasesSensor  bme;   //Temperature, humidity, pressure sensor

Gas NO2(SOCKET_A);   //Gas sensor
Gas CO(SOCKET_B);    //Gas sensor
Gas O3(SOCKET_C);    //Gas sensor
Gas SO2(SOCKET_F);   //Gas sensor
```

- Dohvaćanje interfacea za senzore

```
USB.ON();
NO2.ON();
CO.ON();
O3.ON();
SO2.ON();
PM.ON();
```

- Paljenje senzora

```
#include <WaspFrame.h>

frame.setID(node_ID);
frame.createFrame(ASCII);

frame.addSensor(SENSOR_GASES_PRO_TC, temperature);
frame.addSensor(SENSOR_GASES_PRO_HUM, humidity);
frame.addSensor(SENSOR_GASES_PRO_PRES, pressure);

frame.addSensor(SENSOR_GASES_PRO_NO2, concentration_NO2);
frame.addSensor(SENSOR_GASES_PRO_CO, concentration_CO);
frame.addSensor(SENSOR_GASES_PRO_O3, concentration_O3);
frame.addSensor(SENSOR_GASES_PRO_SO2, concentration_SO2);
frame.addSensor(SENSOR_GASES_PRO_PM1, PM._PM1);
frame.addSensor(SENSOR_GASES_PRO_PM2_5, PM._PM2_5);
frame.addSensor(SENSOR_GASES_PRO_PM10, PM._PM10);
frame.addSensor(SENSOR_BAT, PWR.getBatteryLevel());

data_len = frame.showFrame(Array);
```

- Inicijalizacija i stvaranje frame-a
- Dodavanje podataka u frame preko:

```
frame.addSensor(Sensor_id, variable)
```

- Dohvaćanje podataka od frame-a preko frame.showFrame(Array) i dovačanje dužine frame-a preko povratne vrijednosti funkcije

```
uint16_t showFrame(uint8_t Array[]);
```

```
uint16_t WaspFrame::showFrame(uint8_t Array[])
{
```

```
    USB.secureBegin();

    printString( "frame (STR): ",  0);
    for( uint16_t i= 0; i < length ; i++ )
    {
        printByte( buffer[i],  0);
        Array[i] = buffer[i];

    }

    USB.secureEnd();
    return length;
}
```

- Funkcija showFrame prima jedan argument tipa uint8_t array u koji zapisujemo podatke iz buffera.
- Funkcija vraća vrijednost length (duljina buffera) koja se sprema u varijablu data_len.

```
  char copy[data_len];
  for(int i = 0;i < data_len;i++){
    copy[i] = (char)Array[i];
  }
```

- Pretvorba uint8_t Array-a u polje copy tipa char array.

```
  char data2[data_len];
  int brojac = 0;
  int lijevi;
  int desni;
  int brojac_zapisa = 0;


  for(int i = 0; i < data_len;i++){
    char current = copy[i];
    if(current == '#' && brojac < 4){
      brojac++;
      lijevi = i;
      continue;
    }

    if(current == '#' && brojac > 3){
      desni = i;
      for(int k = lijevi; k < desni;k++){
```

```
        data2[brojac_zapisa++] = copy[k];
      }
      lijevi = desni;

    }

  }
  data2[brojac_zapisa] = '\0';
```

- Skraćivanje i brisanje nepotrebnih podataka iz zapisa (sanitizing of string).
- Podatci unutar zapisa su odvojeni znakom '#'.
- For petlja traži indeks četvrte pojave znaka '#' unutar zapisa i pohranjuje njegov indeks u varijablu lijevi.
- Nakon toga se pohranjuje indeks prvog sljedećeg znaka '#' u zapisu i prepisuje se sadržaj između dva indeksa pohranjenih u varijablama lijevi i desni.
- Nakon toga se sadržaj varijable desni pohranjuje u varijablu lijevi te se ponovno traži sljedeći znak '#' u zapisu.

```
void string2hexString(char* input, char* output)
{
    int loop;
    int i;
    i=0;
    loop=0;

    while(input[loop] != '\0')
    {
        sprintf((char*)(output+i),"%02X", input[loop]);
        loop+=1;
        i+=2;
    }
    //insert NULL at the end of the output string
    output[i++] = '\0';
}
```

- Funkcija string2hexString() prima 2 ulazna argumenta tipa char array.
- While petljom iteriramo kroz polje dok ne naiđemo na terminator '\0'.
- Svakim prolazom kroz petlju pretvara se trenutni znak iz ASCII zapisa u heksadecimalni zapis.
- Na kraj zapisa dodaje se terminator '\0'.

```
char hex_str[(data_len*2)+1];
string2hexString(data2, hex_str);

 int counter = 0;

while(hex_str[counter] != '\0'){
  counter++;
 }

 char final_hex[counter];

  counter = 0;
  while(hex_str[counter] != '\0'){
    final_hex[counter] = hex_str[counter];
    counter++;
 }
```

- Poziv funkcije `string2hexString(data2, hex_str);` za pretvorbu dobivenog polja charactera data2[] u polje hex znakova hex_str[]
- Maksimalno potrebni prostor za pretvorbu stringa u hex_string iznosi 2*length(stringa) +1
- Budući da je moguće da nije cijelo polje hex_str[] popunjen broji se duljina popunjenog prostora hex_str[] do nul_terminatora
- Prostor koji je iskorišten u hex_str[] prebacujemo u final_hex[] polje
- Time smo postigli smanjenje informacija koje je potrebno poslati preko LoraWan-a

```
  int j=0;
  int flag = 0;
  if(counter>100){
    for(int i=0;i<counter;i++){
      if(j<100){
        flag = 0;
        data[j]=final_hex[i];
        j++;
      }
      else{
        flag = 1;
        data[j] = '\0';
        int counter2 = 0;
        while(data[counter2] != '\0'){
          counter2++;
       }
        SendData(PORT, data, counter2);
        j=0;
        for (int z=0;z<100;z++){
```

```
            data[z]='\0';
          }
          data[j++] = final_hex[i];
        }
      }
      if(flag == 0){
          SendData(PORT,data,j);
      }
    }else{
      int counter2 = 0;
      while(data[counter2] != '\0'){
        counter2++;
     }
     for(int i = 0; i<counter2; i++){
          USB.print(data[i]);
        }
      USB.println();
      SendData(PORT,data, counter2);
    }
```

- Razdvajanje poruke na više manjih dijelova ako je potrebno i slanje svakog dijela podatka preko LoraWan-a
- Svaki dio poruke koji se šalje može imati maksimalnu veličinu od 100 znakova
- U char polje data2 sprema se 100 znakova iz poruke unutar final_hex polja
- Nakon spremanja 100 znakova polje data2 se resetira i popunjuje terminatorom '\0'
- Zastavica služi za određivanje duljine trenutnog dijela poruke, ako poruka ima manje od 100 znakova zastavica ima vrijednost 0, a ako ima točno 100 znakova onda ima vrijednost 1
- Zastavica je potrebna kako bi se poslao zadnji dio poruke koji ima manje od 100 znakova

```
void SendData(int PORT, char data[], int data_length){

   error = LoRaWAN.joinABP();

  if( error == 0 )
  {
    USB.println(F("2. Join network OK"));
  error = LoRaWAN.sendUnconfirmed( PORT, data);

    if( error == 0 )
    {
      USB.println(F("3. Send unconfirmed packet OK"));
      if (LoRaWAN._dataReceived == true)
      {
        USB.print(F("   There's data on port number "));
```

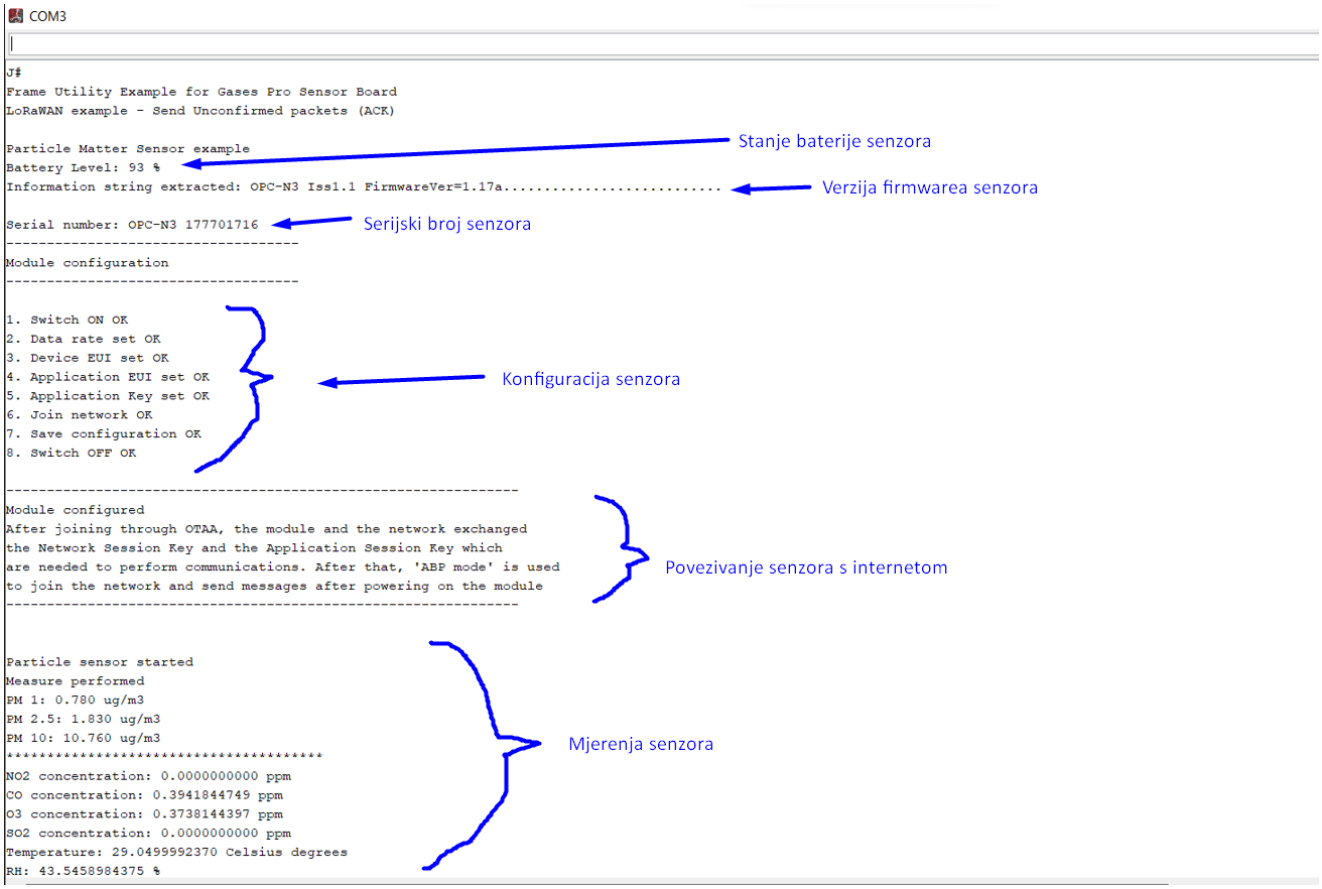
```
        USB.print(LoRaWAN._port,DEC);
        USB.print(F(".\r\n   Data: "));
        USB.println(LoRaWAN._data);
      }
    }
    else
    {
      USB.print(F("3. Send unconfirmed packet error = "));
      USB.println(error, DEC);
    }
  }
  else
  {
    USB.print(F("2. Join network error = "));
    USB.println(error, DEC);
  }
}
```

- Funkcija SendData() prima 3 argumenta tipa int, char polje i int
- Služi za slanje podataka preko LoraWan-a
- LoraWan.joinABP() funkcija služi za spajanje na server
- LoraWan.sendUnconfirmed() funkcija šalje Port i podatak na server kao ulazne argumente te ako se podatci uspješno pošalju ispisuju se informacije o portu i poruci poslanog podatka

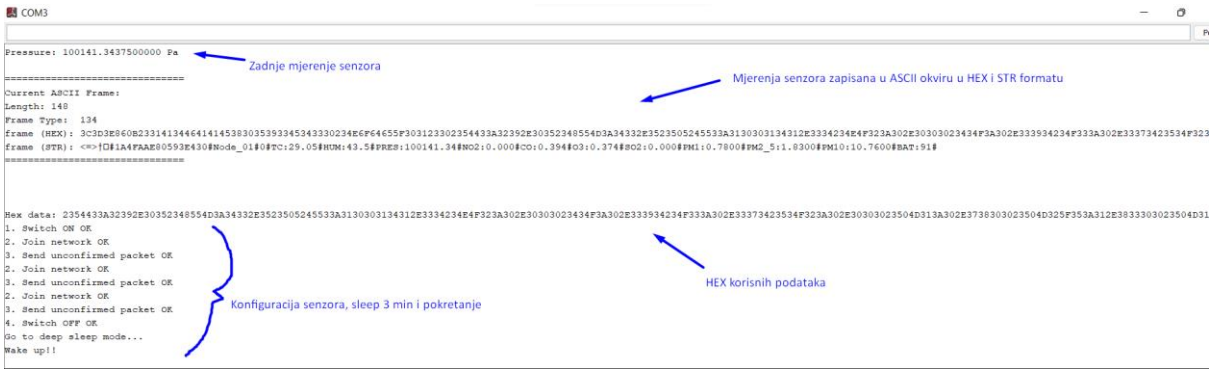
```
  bme.OFF();
  O3.OFF();
  SO2.OFF();
  NO2.OFF();
  CO.OFF();
  PM.OFF();
```

```
  USB.println(F("Go to deep sleep mode..."));
  PWR.deepSleep("00:00:03:00", RTC_OFFSET, RTC_ALM1_MODE1, ALL_OFF);
  USB.ON();
```

- Gašenje senzora te ponovno paljenje nakon 3 minute



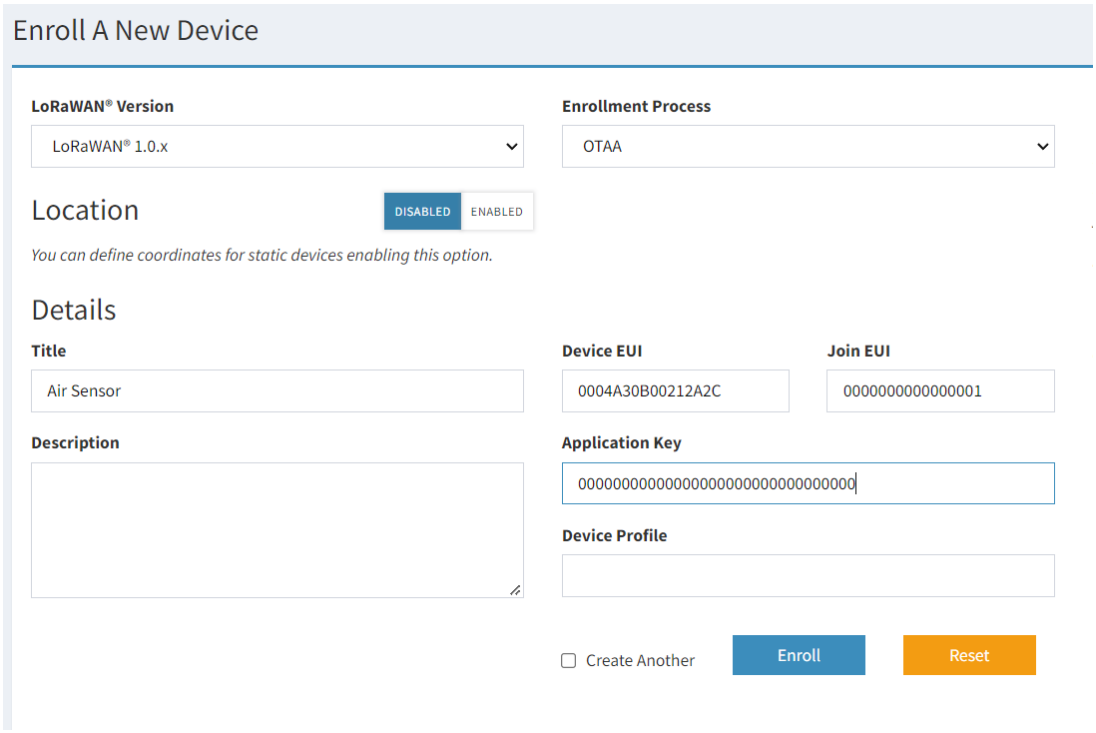
Slika 8. Ispis očitanih podataka senzora



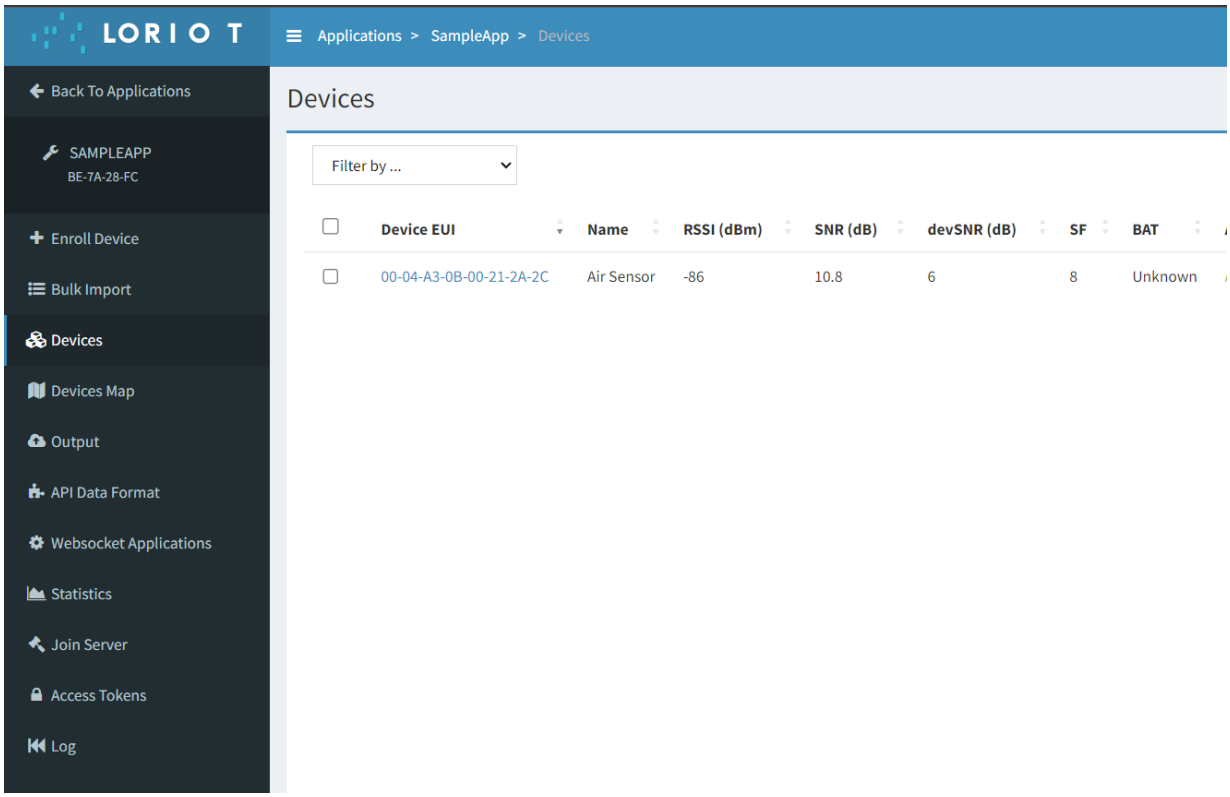
Slika 9. Nastavak ispisa očitanih podataka senzora

Slanje snimljenih podataka sa senzora na Loriot (LoRaWAN)

Podatke snimljene na senzoru slali smo na gateway od Loriot networka. Dobili smo Loriot račun sa već registriranim gateway-om, te je na nama bilo samo da registriramo (odnosno enrollamo) novi device točnije naš senzor. Bitno je bilo da su Device EUI te Join EUI jedinstveni.

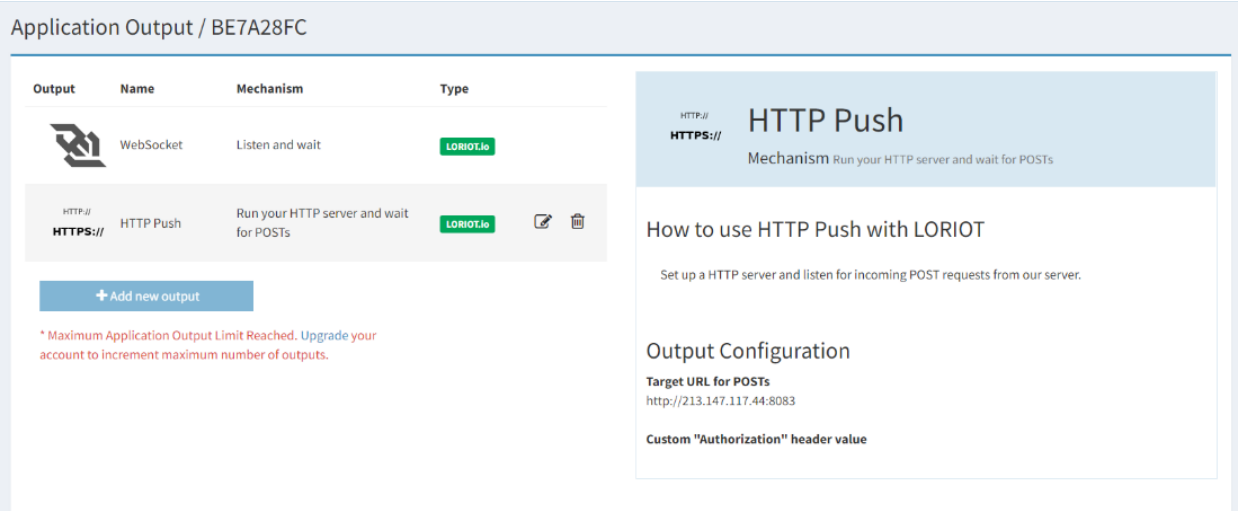


Slika 10. Enroll našeg senzora



Slika 11. Registrirani device

Zatim smo morali definirati komunikaciju sa virtualnim uređajem na kojem će se obrađivati podaci. Komunikaciju smo ostvarili HTTP protokolom gdje ćemo POST zahtjevima slati podatke. Podatci će se slati na javnu adresu našeg virtualnog uređaja na port 8083.



Slika 12. HTTP Push na Loriot platformi

curl -A POST http://213.147.117.44:8083

sudo tcpdump -i ens192 -v -s 0 -A -w test.pcap

Obrada podataka

Na virtualnom uređaju obađujemo primljene podatke skriptom ‘server.py’. Ona sluša za dolazne POST zahtjeve na portu 8083 filtrira ih te ih spaja. Spojeni podaci se konačno ‘izvlače’ pomoću funkcije extractData() koja će ih pretvarati iz heksadekadskih u ASCII vrijednosti te ih spremati u mapu.

```
def extractData(data):
    data = "".join(data.split())
    byte_array = bytearray.fromhex(data)
    data = str(byte_array.decode())
    data = data.split('#')
    del data[0]
    data_map = {}
    for item in data:
        passed = False
        name = ""
        value = ""
        for letter in item:
            if letter == ':':
                passed = True
                continue
            if passed:
                value += letter
            else:
                name += letter

        data_map[name] = value
```

Slika 13. Funkcija extractData()

Uz snimljene podatke, u mapu će se ubacivati evidentirani podatak u trenutku dolaska primljenih podataka (timestamp). Zatim će se svi podaci iz mape ubacivati u bazu podataka.

Kreiranje te povezivanje baze podataka s programom

Kreiranje baze podataka:

```
CREATE TABLE airsensor (
temperature         varchar(50),
humidity            varchar(50),
pressure            varchar(50),
concentration_NO2   varchar(50),
concentration_CO    varchar(50),
concentration_O3    varchar(50),
concentration_SO2   varchar(50),
PM_PM1              varchar(50),
PM_PM2_5            varchar(50),
```

```
PM_PM10             varchar(50),
battery             varchar(50),
tstamp              timestamp
);
```

Povezivanje baze podataka:

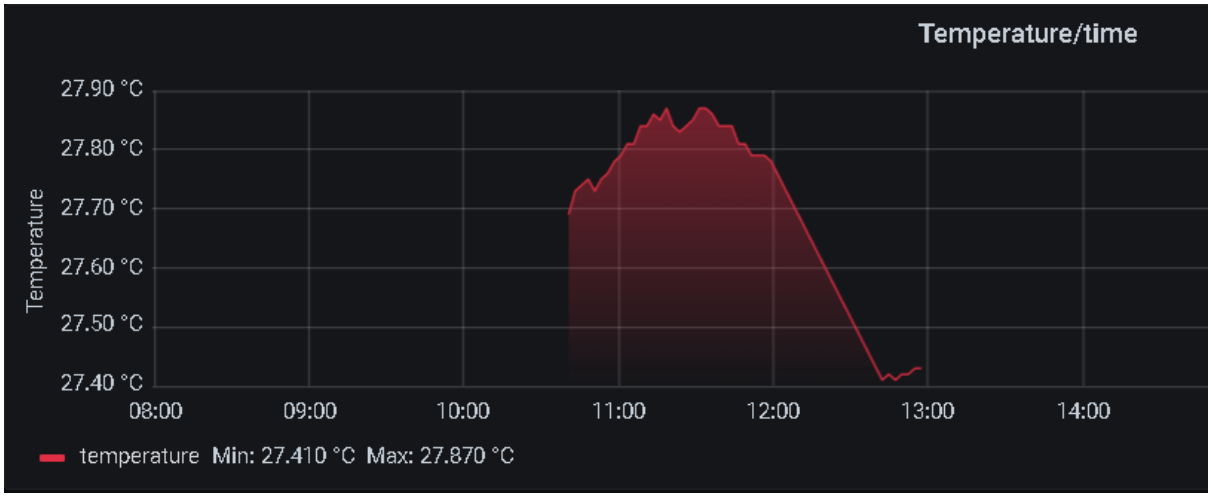
```
def sendToDB(data):
    connection = psycopg2.connect("dbname=postgres user=postgres
password=123")

    cursor = connection.cursor()

    cursor.execute("""INSERT INTO airsensor (temperature, humidity, pressure,
concentration_NO2, concentration_CO, concentration_O3, concentration_SO2,
PM_PM1, PM_PM2_5, PM_PM10, battery, TSTAMP)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)""", (data["TC"],
data["HUM"], data["PRES"], data["NO2"], data["CO"], data["O3"], data["SO2"],
data["PM1"], data["PM2_5"], data["PM10"], data["BAT"], data["TSTAMP"]))

    connection.commit()
```

Grafana



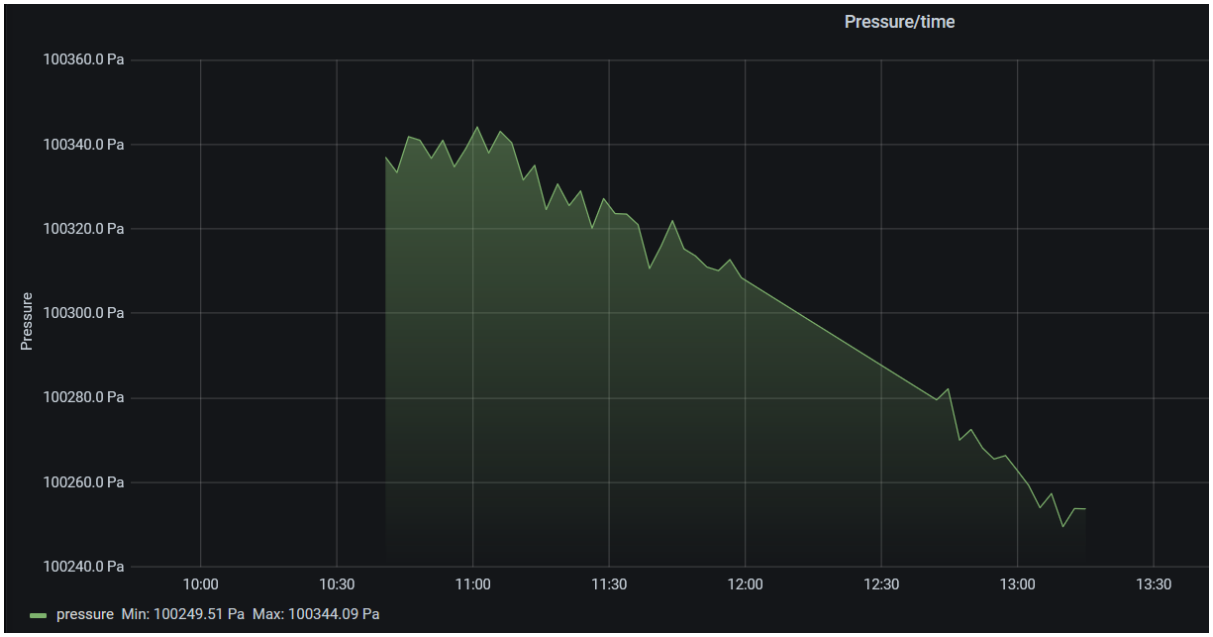
Slika 14. Temperature/time graf

```
SELECT
  tstamp as time, temperature::FLOAT
```

```
FROM
airsensor;
```

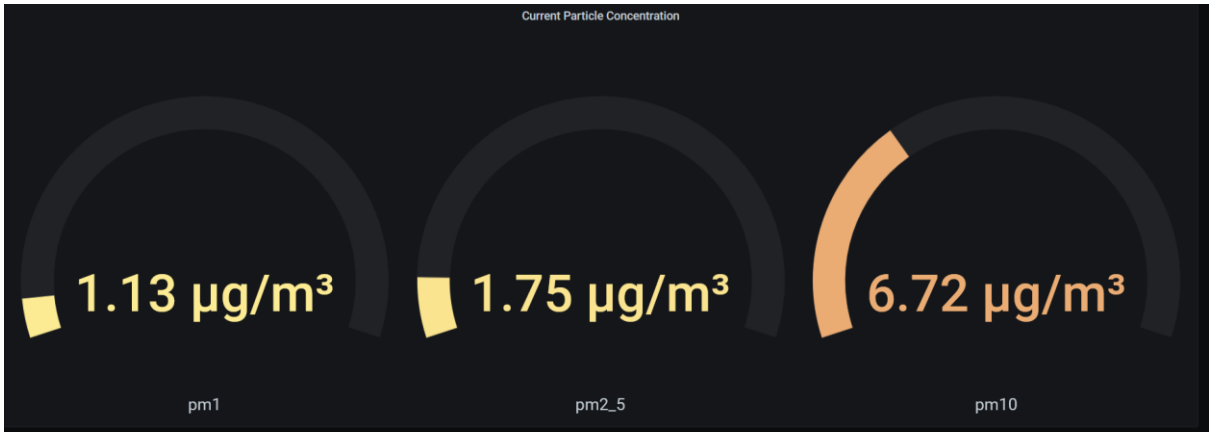
Slika 15. Humidity/time graf

```
SELECT
  tstamp as time, humidity::FLOAT
FROM
  airsensor;
```



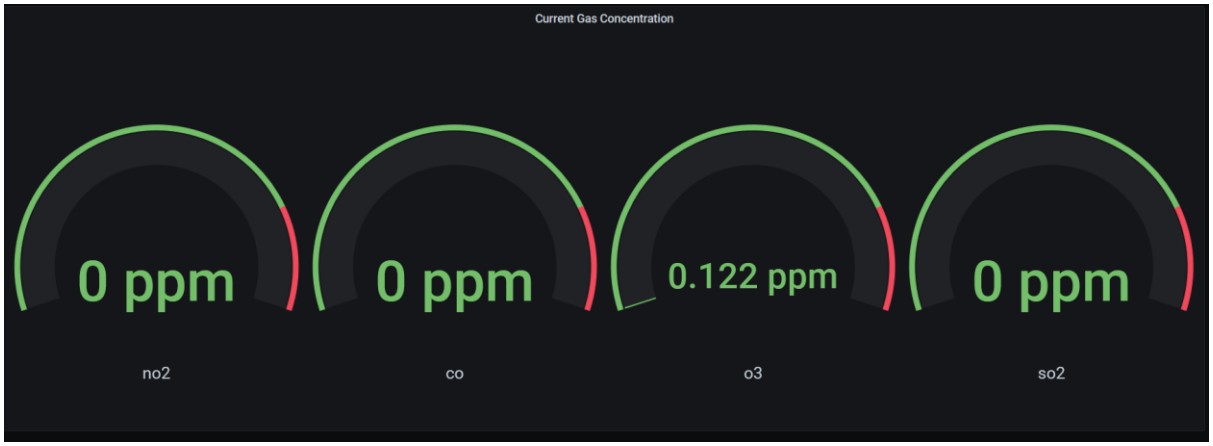
Slika 16. Pressure/time graf

```
SELECT
  tstamp as time, pressure::FLOAT
FROM
  airsensor;
```



Slika 17. Trenutna koncentracija čestica

```
SELECT
pm_pm1::float as pm1,pm_pm2_5::float as pm2_5,pm_pm10::float as pm10
from airsensor;
```



Slika 18. Trenutna koncentracija plinova

```
SELECT
  concentration_no2 as no2,concentration_co as co,concentration_o3 as
o3,concentration_so2 as so2
FROM airsensor;
```

Dodatci

Server.py kod

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import logging
import json
import datetime
import psycopg2

####
# python3 server.py <PORT>
# python3 server.py 8083
####

counter = 0
data_global = ""
data_array = []

def sendToDB(data):
    connection = psycopg2.connect("dbname=postgres user=postgres
password=123")
```

```
    cursor = connection.cursor()

    cursor.execute("""INSERT INTO airsensor (temperature, humidity, pressure,
concentration_NO2, concentration_CO, concentration_O3, concentration_SO2,
PM_PM1, PM_PM2_5, PM_PM10, battery, TSTAMP)
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)""", (data["TC"],
data["HUM"], data["PRES"], data["NO2"], data["CO"], data["O3"], data["SO2"],
data["PM1"], data["PM2_5"], data["PM10"], data["BAT"], data["TSTAMP"]))

    connection.commit()

def extractData(data):
    data = "".join(data.split())
    byte_array = bytearray.fromhex(data)
    data = str(byte_array.decode())
    data = data.split('#')
    del data[0]
    data_map = {}
    for item in data:
        passed = False
        name = ""
        value = ""
        for letter in item:
            if letter == ':':
                passed = True
                continue
            if passed:
                value += letter
            else:
                name += letter

        data_map[name] = value

    vrijeme = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    data_map["TSTAMP"] = vrijeme
    sendToDB(data_map)



class S(BaseHTTPRequestHandler):
    def _set_response(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()


    def do_POST(self):
        content_length = int(self.headers['Content-Length']) # <--- Gets the
size of data
```

```
        post_data = self.rfile.read(content_length) # <--- Gets the data
itself

        self._set_response()
        self.wfile.write("POST request for {}".format(self.path).encode('utf-
8'))

        json_data = json.loads(post_data.decode('utf-8'))
        data = json_data["data"]

        global counter, data_array, data_global

        if counter != 3 and data not in data_array:
            data_global += data
            counter += 1
            data_array.append(data)
            print('NUMBER OF DATA: ' + str(counter) + ' , data:'+ data)
        elif counter == 3:
            extractData(data_global)
            data_global = ""
            counter = 0
            data_array = []


def run(server_class=HTTPServer, handler_class=S, port=8083):
    logging.basicConfig(level=logging.INFO)
    server_address = ('0.0.0.0', port)
    httpd = server_class(server_address, handler_class)
    logging.info('Starting httpd...port: %d\n', port)


    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
    logging.info('Stopping httpd...\n')

if __name__ == '__main__':
    from sys import argv

    if len(argv) == 2:  #  pocni na novom unosu za port ako ima dodatnog unosa
        run(port=int(argv[1]))
    else:
        run()  #zapocni ako nema dodatnog inputa
```