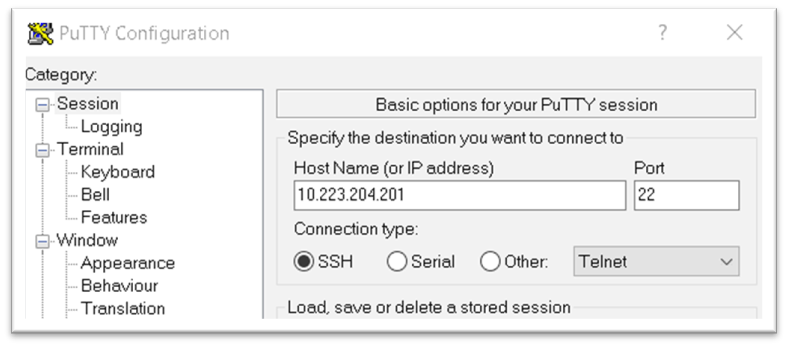
Smart\_Gases\_PRO\_v30

IP adresa: 10.223.204.201

Port: 7800



Slika 1. Otvaranje sesije kroz PuTTy alat

LORAWAN NETWORK SERVER

<https://eu1.loriot.io/>

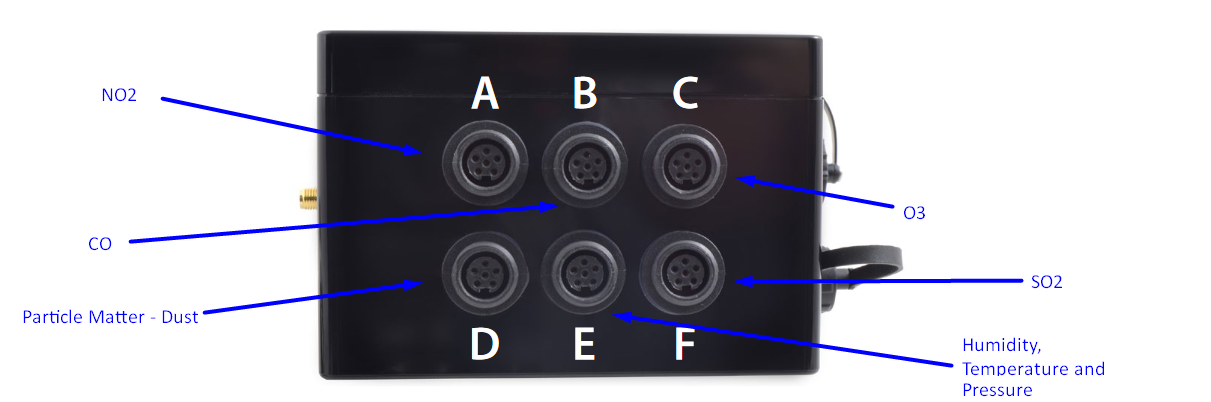
user: [nikola.zagorec@versoaltima.com](mailto:nikola.zagorec@versoaltima.com)

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](https://www.cooking-hacks.com/documentation/tutorials/waspmote/index.html%23waspmote_ps_2_8_2)

[Vodič za programiranje senzora](https://development.libelium.com/gases_pro_sensor_guide/)

[Vodič za rad u programu Waspmote](https://development.libelium.com/waspmote-programming-guide/)

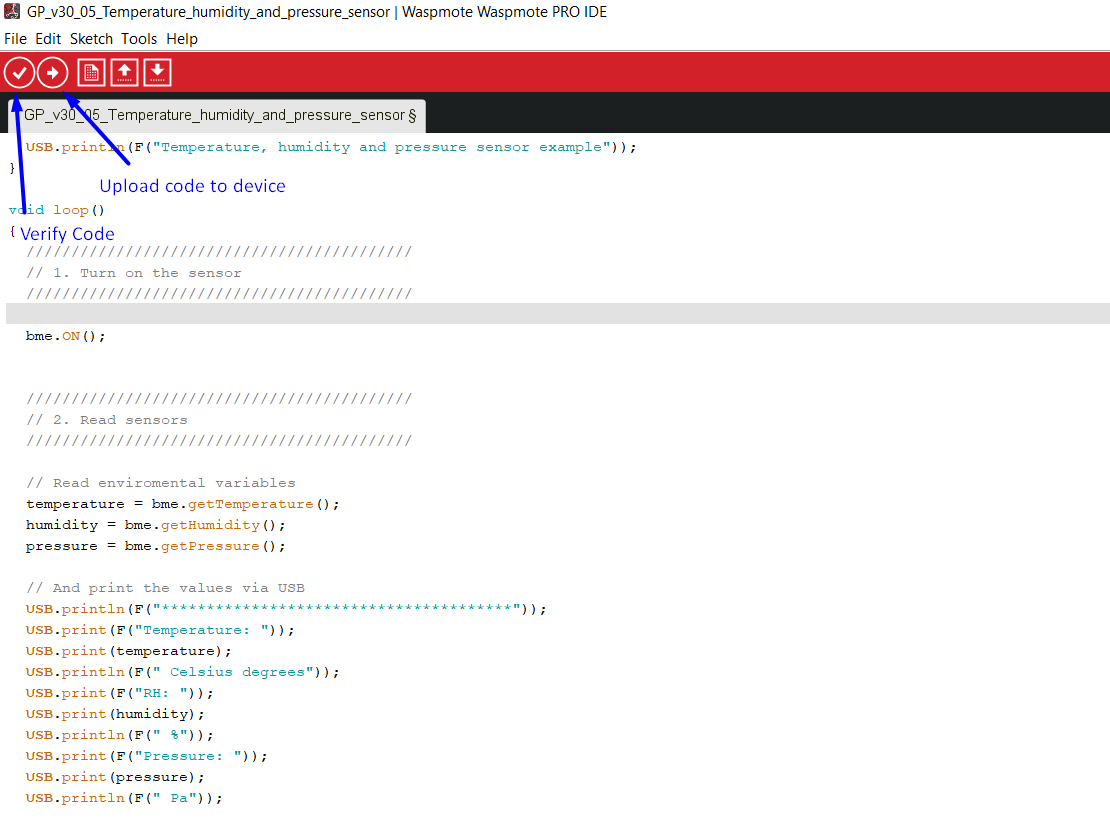
[Vodič za programiranje u Waspmoteu koristeći priložene module](https://development.libelium.com/utilities-programming-guide/)

[Primjeri kodova u programu Waspmote](https://development.libelium.com/waspmote/code-examples/)

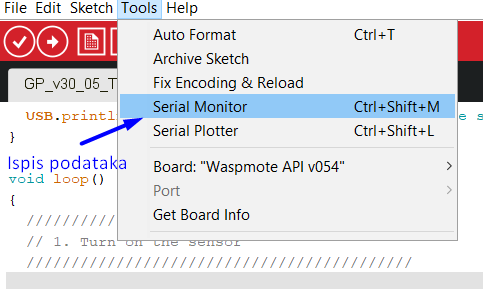
# Korištenje programa WaspMote

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

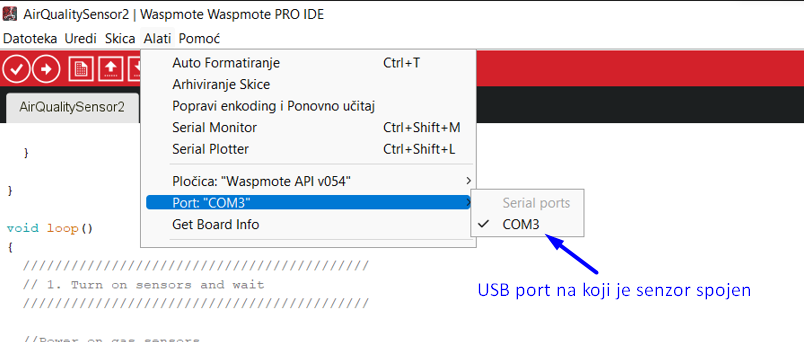
Firmware je baziran na C++-u, a i zaglavlja te moduli koje smo koristili su pisani u C++-u.



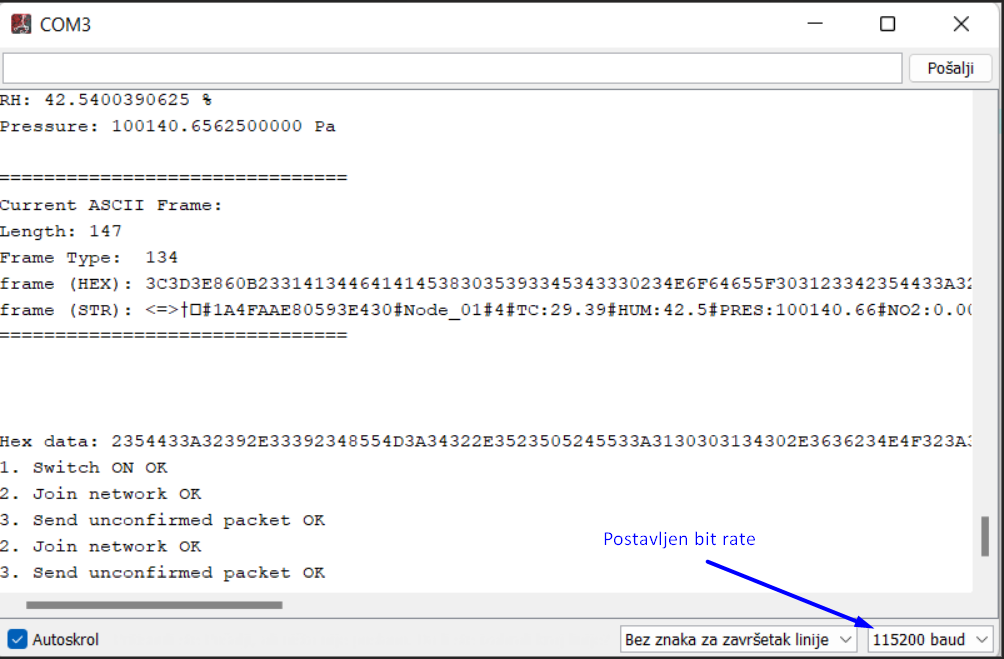
Slika 4. Funkcije programa, compile i upload



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();

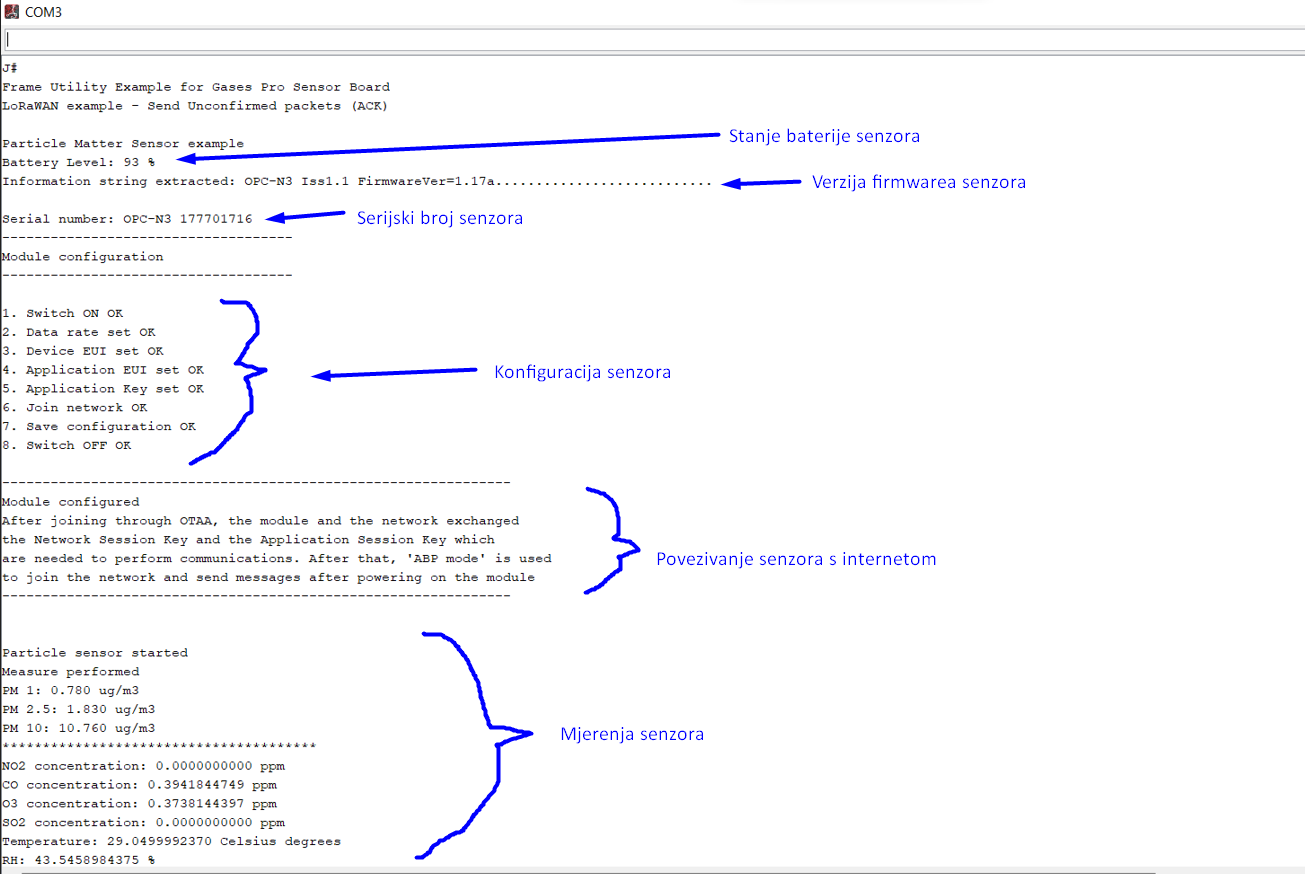
  PM.OFF();

  USB.println(F("Go to deep sleep mode..."));

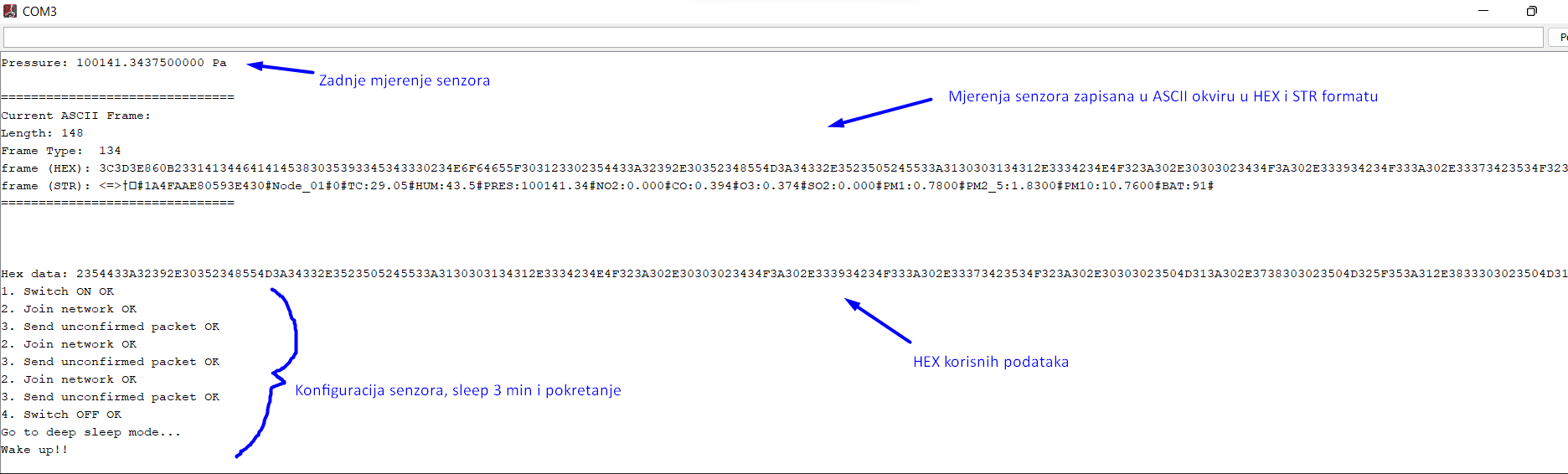
  PWR.deepSleep("00:00:03:00", RTC\_OFFSET, RTC\_ALM1\_MODE1, ALL\_OFF);

  USB.ON();

* Gašenje senzora te ponovno paljenje nakon 3 minute



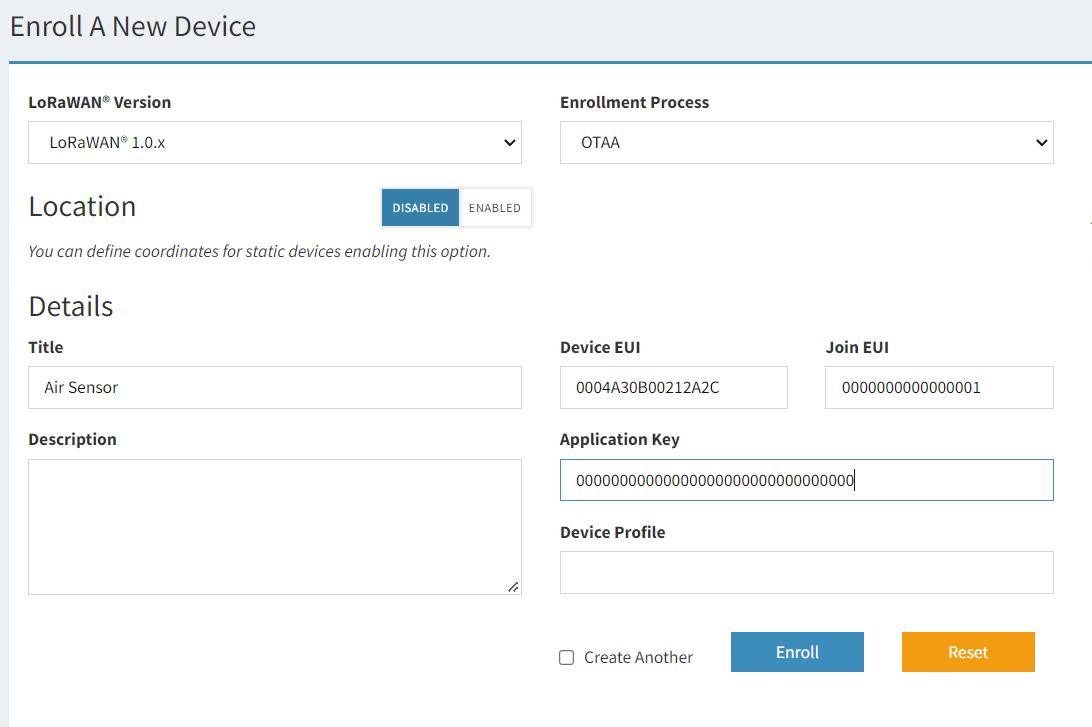
Slika 8. Ispis očitanih podataka senzora



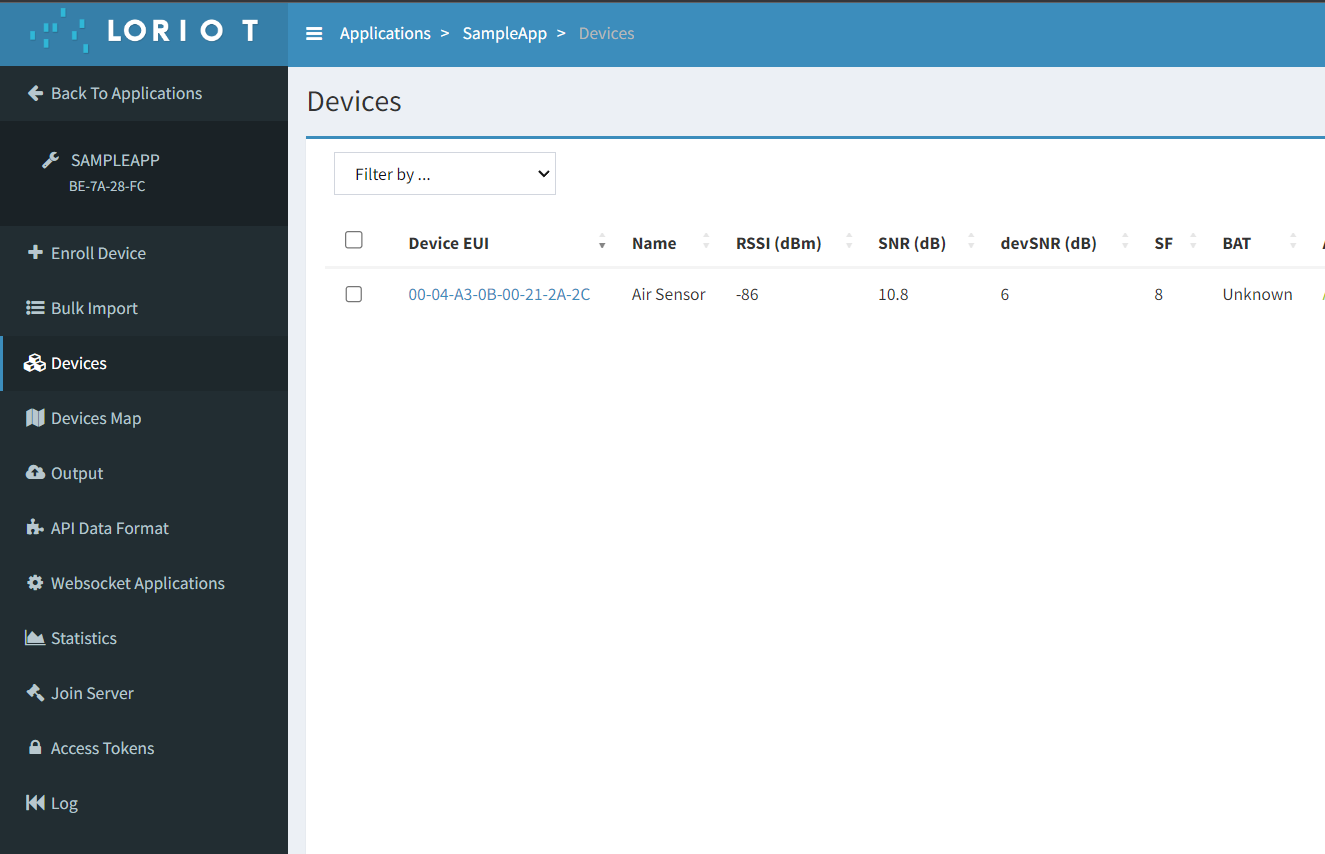
Slika 9. Nastavak ispisa očitanih podataka senzora

# 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.

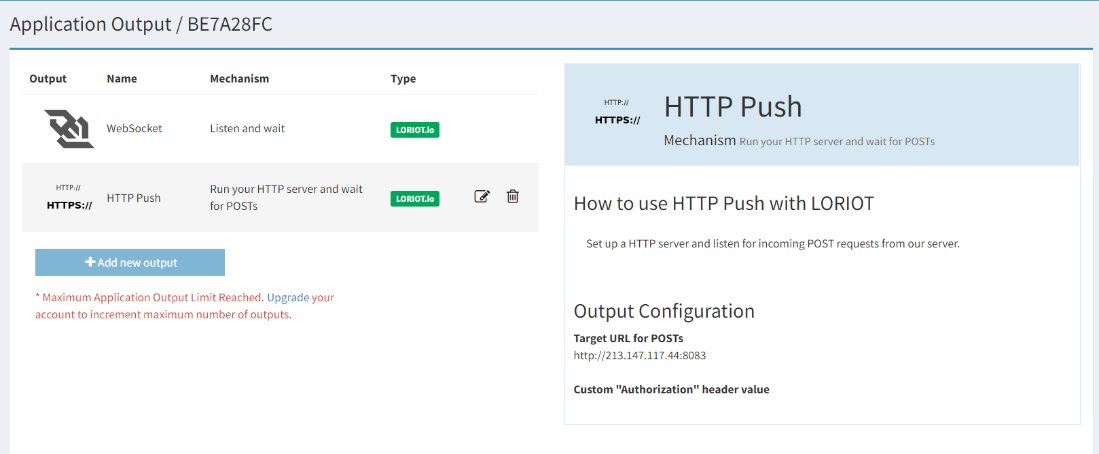


Slika 10. Enroll našeg senzora



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.



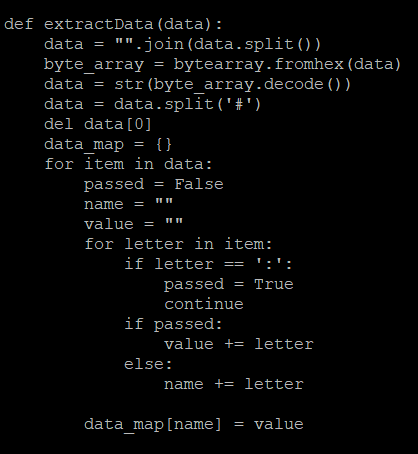
Slika 12. HTTP Push na Loriot platformi

*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.



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")

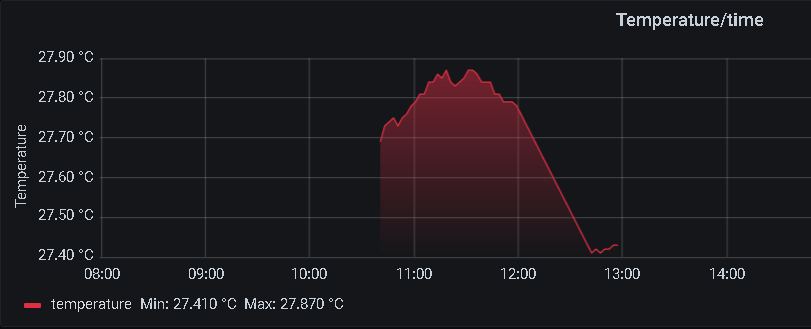
    cursor = connection.cursor()

    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;*



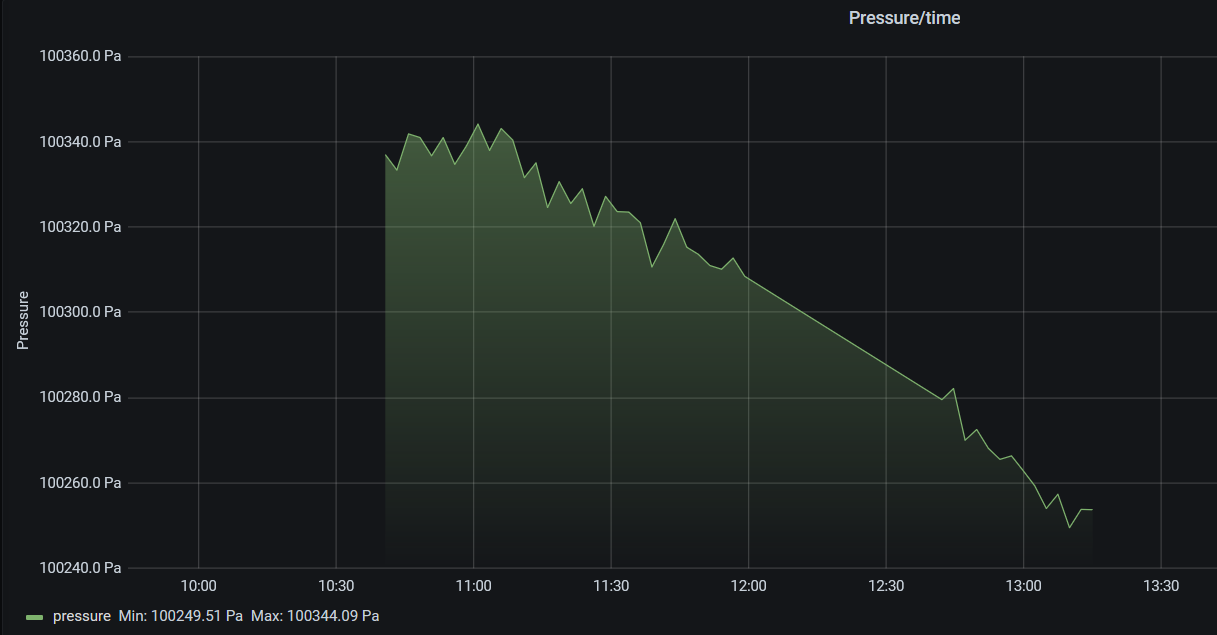
Slika 15. Humidity/time graf

*SELECT*

*tstamp as time, humidity::FLOAT*

*FROM*

*airsensor;*



Slika 16. Pressure/time graf

*SELECT*

*tstamp as time, pressure::FLOAT*

*FROM*

*airsensor;*

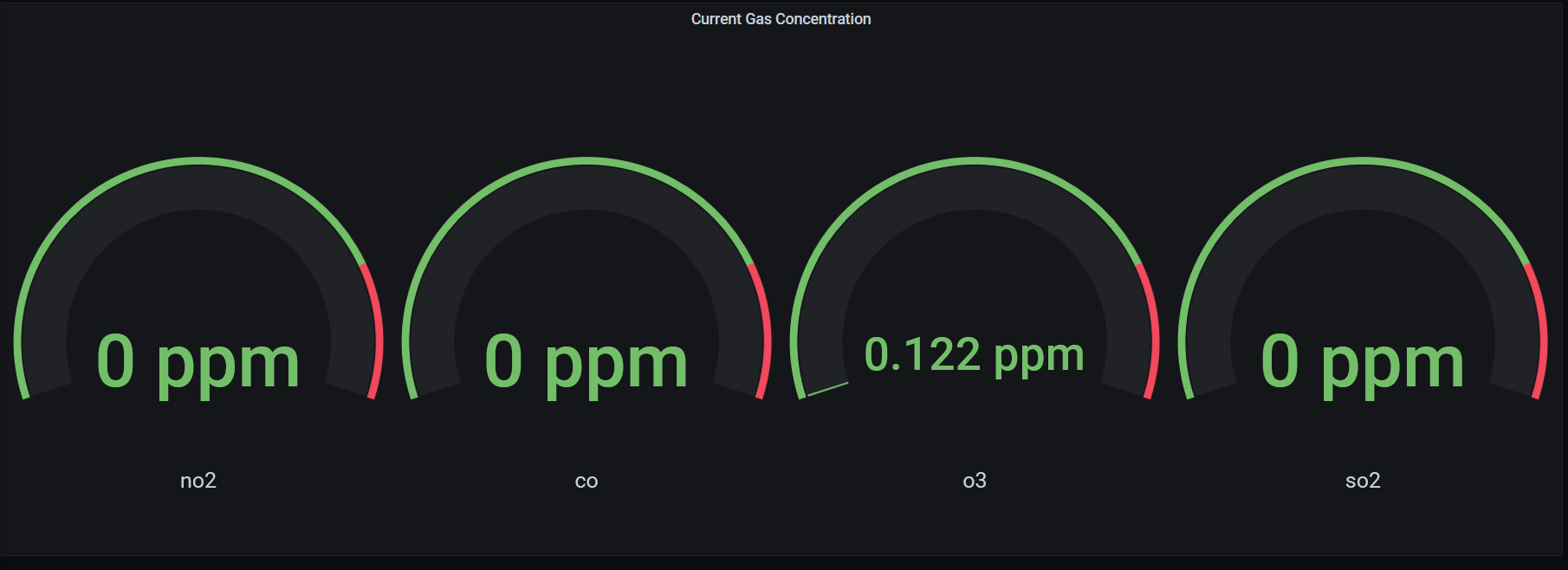


Slika 17. Trenutna koncentracija čestica

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