TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA
GETALDIĆEVA 4, ZAGREB

ZAVRŠNI STRUČNI RAD:

Meteorološka postaja s Arduinom

MENTOR:
Zlatko Nadarević, dipl. ing. el.

UČENIK: Anđelko Kućar
RAZRED: 4.F

Zagreb, travanj 2021.

Sadržaj

1. Uvod

Meteorologija je znanost koja proučava sastav i strukturu Zemljine atmosfere te promjene u njoj. Znanstveni razvoj meteorologije započeo je sredinom sedamnaestog stoljeća primjenom prvih meteoroloških mjernih instrumenata.

Meteorološke postaje mjere atmosferske uvijete te se na temelju tih podataka predviđaju vremenske prognoze. Profesionalne postaje sadrže veoma skupe i precizne mjerne instrumente. Ti instrumenti mjere: atmosferski tlak, vlagu zraka, temperaturu, količinu padalina, brzinu vjetra, smjer vjetra, jakost Sunčeve svjetlosti, kvalitetu zraka, svjetlosnu zagađenost, zvučnu zagađenost i dr.

U današnje vrijeme sve su popularnije meteorološke postaje kućne izrade. Mjerni instrumenti relativno dobre preciznosti i male cijene te platforme poput Arduina omogućuju izradu takvih projekata.

U ovom radu obrađeno je povezivanje senzora na mikroupravljač, prosljeđivanje prikupljenih podataka bežično na server, spremanje tih podataka u bazu podataka te prikaz podataka pomoću grafova na web stranici.

Cilj ovog rada je vizualno prikazati korisniku stanje osnovnih atmosferskih uvjeta poput temperature, vlage i tlaka te mu dati dobru predodžbu o kvalitetu zraka mjereći razinu sitnih čestica i CO_2 u zraku.

2. Opis tehnologije

2.1. Arduino

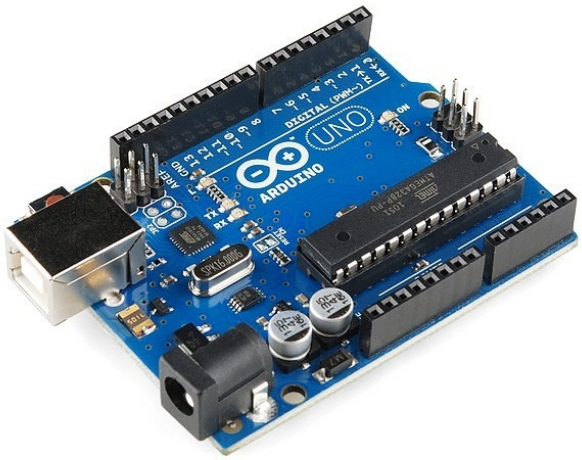
Arduino je otvorena računalna i softverska platforma koja omogućuje veoma jednostavno korištenje mikroupravljača za izradu jednostavnijih projekata, ali i onih veće kompleksnosti.

2.1.1. Arduino Uno

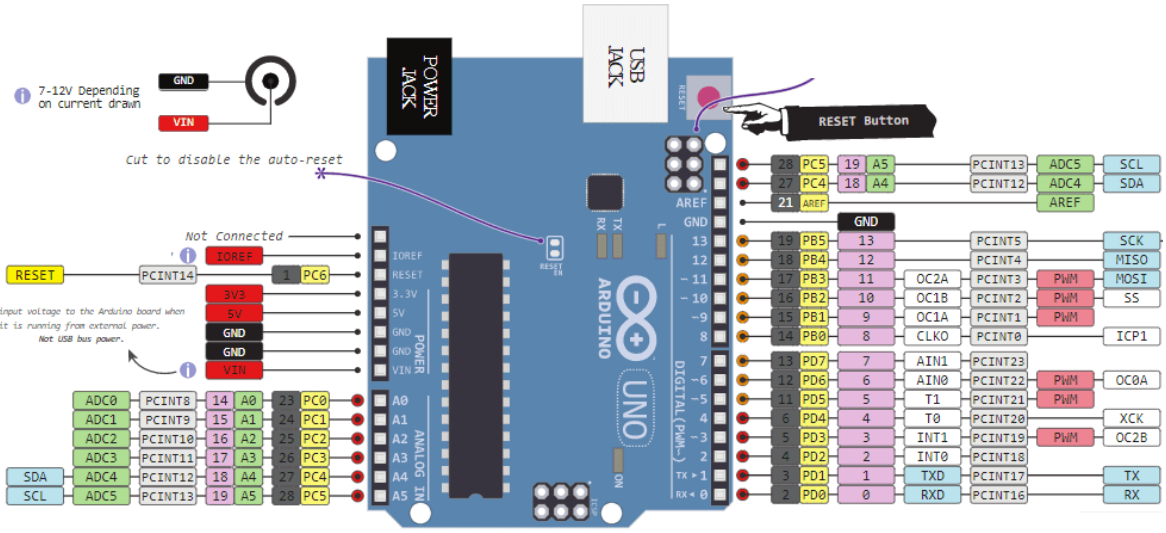
Arduino Uno je razvojna pločica bazirana na ATmega328p mikroupravljaču. Ima 14 digitalnih ulazno/izlaznih pinova i 6 analognih ulaznih pinova. Program se na mikroupravljač prebacuje preko USB-B priključka. Nakon što je program učitan na mikroupravljač, on može dalje uz vanjsko napajanje autonomno izvoditi zadani program.

Karakteristike:

- Mikroupravljač: ATmega328p
- Radni napon: 5V
- Ulazni napon: 7 – 12 V
- Flash memorija: 32 KB
- SRAM: 2 KB
- EEPROM: 1 KB
- Pinovi: 14 digitalnih ulazno/izlaznih, 6 analognih ulaznih



Slika 2.1. Arduino Uno



Slika 2.2. Arduino Uno pinovi i njihove funkcije

2.1.2. ArduinoIDE

ArduinoIDE je platforma za programiranje mikroupravljača na Arduino razvojnim pločicama. Platforma omogućuje pisanje vlastitog koda u C/C++ programskom jeziku, ali i korištenje već gotovih biblioteka koje su otvorenog koda. Zbog velike zajednice postoji i veliki broj biblioteka koje omogućuju brzo programiranje uređaja i veću preglednost unutar samog programskog koda.

Svaki program napisan u ArduinoIDE sučelju mora imati dvije funkcije:

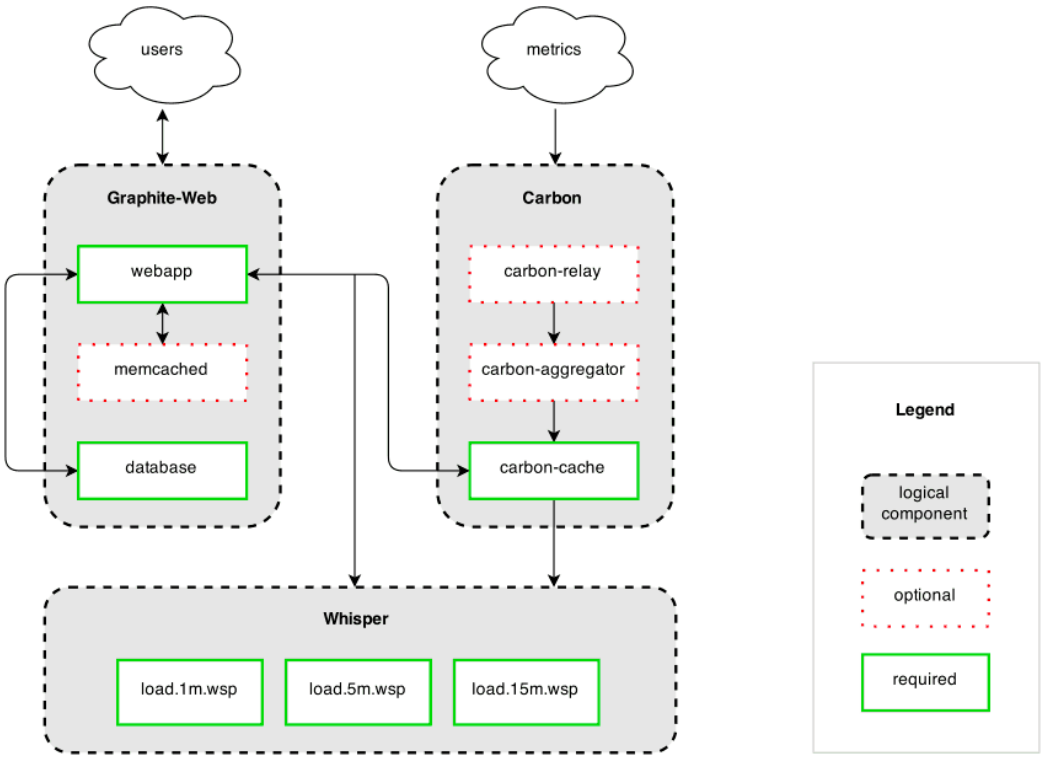
void setup() - funkcija se pokreće samo prilikom prvog ciklusa izvođenja programa i u nju se stavljaju naredbe za inicijalizaciju uređaja

void loop() - funkcija se pokreće sve dok je mikroupravljač spojen na napajanje

Za ostvarivanje kontakta s već isprogramiranim mikroupravljačem, koristi se Serial Monitor. Na Serial Monitoru se mogu ispisivati podatci koje mikroupravljač obrađuje, a preko njega se mogu i slati podatci na mikroupravljač. Veoma je koristan za razvijanje programskog koda jer ga se može koristiti umjesto ulazno/izlaznih uređaja poput LCD zaslona i tipkovnice što korisniku uvelike olakšava testiranje samog programa.

2.2. Graphite

Graphite je sustav za prihvat i spremanje podataka te prikazivanje istih. Sastoji se od tri podsustava: carbon, whisper i graphite-web.



Slika 2.3. Graphite struktura

2.2.1. Carbon

Servis velikih performansi koji služi za prihvat vremenski obilježenih podataka.

2.2.2. Whisper

Jednostavna baza podataka za spremanje vremenski obilježenih podataka.

2.2.3. Graphite-Web

Graphitovo korisničko sučelje i API (Application Programming Interface – služi kao poveznica između dvije aplikacije i omogućuje im da međusobno komuniciraju) za prikazivanje grafova i nadzornih ploča.

Graphite sam po sebi ne skuplja podatke. Podatci mu moraju biti dostavljeni od neke druge aplikacije u određenom formatu. Kad se podatci dostave, servis Carbon pokupi te podatke i proslijedi ih u Whisper bazu podataka na trajnu pohranu. Kad korisnik pristupi Graphite web sučelju, ono pošalje zahtjev za podatke Carbonu i Whisperu i po primitku podataka konstruira graf.

2.3. Grafana

Grafana je web aplikacija za analizu i interaktivnu vizualizaciju podataka. Aplikacija vadi podatke iz baze podataka i grafički ih prikazuje pomoću velikog broja interaktivnih grafova.

Grafana omogućuje krajnjem korisniku stvaranje nadzornih ploča, gdje korisnik može po želji stvarati grafove od odabranih podataka. Korisnik ima veliki broj opcija za prilagodbu izgleda pojedinog grafa. Ukoliko korisnik i dalje nije zadovoljan ponuđenim opcijama, ima mogućnost dodavanja dodatnih modula (Plugins).

Aplikacija je veoma intuitivna za koristiti. Krajnjem korisniku omogućuje namještanje postavki aplikacije pomoću grafičkog sučelja.

Slika 2.4. Grafana izgled sučelja

2.4. Flask

Flask je programski okvir za razvoj web aplikacija u Pythonu. Omogućuje korisniku primanje i slanje HTTP metoda putem Werkzunga te generiranje HTML datoteka pomoću predložaka koristeći Jinja2 programske okvire.

3. Opis rada

Rad se sastoji od fizičkog i programskog dijela. Fizički dio rada se sastoji od vanjskog dijela u kojem se nalaze senzori i unutarnjeg dijela u kojem se nalaze sklopovi za prikupljanje podataka, računanje stvarnog vremena, bežičnu komunikaciju i napajanje sustava. Programski dio rada sastoji se od dva dijela, programa koji se nalazi na samom uređaju i programa na serveru. Uređaj se brine za prikupljanje podataka, a server za obradu i prikaz.

3.1. Komponente

3.1.1. PMS5003

PMS5003 je Plantowerov senzor za mjerenje sitnih čestica u zraku. Količinu čestica mjeri pomoću laserske zrake.

Senzor mikroupravljaču šalje dvije skupine mjerenja. Šalje mu koncentraciju čestica u $\mu g/m^3$ (čestice veličine 1.0, 2.5 i 10 μm) i šalje mu broj čestica u 0.1L zraka (čestice veličine 0.3, 0.5, 1.0, 2.5, 5.0 i 10 μm).

Karakteristike:

- Veličine mjerenih čestica: 0.3, 0.5, 1.0, 2.5, 5.0, 10 μm
- Radni napon: 4.5 – 5.5 V
- Radna temperatura: -10 – 60 °C
- Razlučivost: 1 $\mu g/m^3$

Slika 3.1. PMS5003 senzor

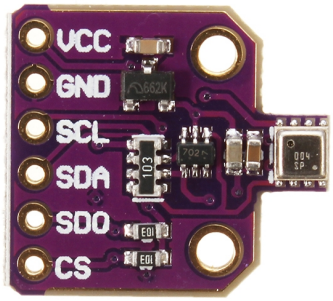
3.1.2. BME680

BME680 je modul koji mjeri temperaturu, vlagu i tlak. Dodatno može mjeriti i razinu određenih plinova u zraku, što korisniku može dati dobru pretpostavku o kvaliteti zraka u zatvorenom prostoru u kojem se senzor nalazi. Kvalitetu zraka izražava u ohmima te je potreban poseban softver za pretvoriti tu veličinu u neku korisnu jedincu kvalitete zraka. Taj softver nije moguće pokretati na Arduino Unu, već je potreban jači mikroupravljač. Zato se ta vrijednost ne koristi u ovom radu.

Modul je dobio naziv po Bosch BME680 senzoru koji je glavni dio modula. Komunikacija između modula i mikroupravljača može biti ostvarena I2C ili SPI komunikacijom.

Karakteristike BME680 senzora:

- Raspon mjerenja temperature: - 40 °C – 85 °C
  - Preciznost: ± 1.0 °C
- Raspon mjerenja vlage u zraku: 0 % – 100 %
  - Preciznost: ± 3 %
- Raspon mjerenja atmosferskog tlaka: 300 – 1100 hPa
  - Preciznost: ± 1 hPa
- Radni napon: 1.7 – 3.6 V



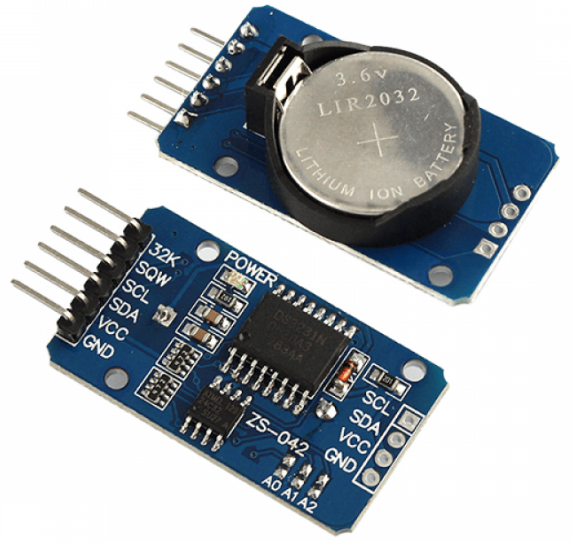
Slika 3.2. BME680 senzor

3.1.3. DS3231

Arduino Uno nema ugrađenu funkcionalnost praćenja stvarnog vremena te je zbog toga potrebno koristiti poseban modul. DS3231 precizno mjeri vrijeme od trenutka kad mu je postavljeno početno vrijeme, najčešće uzeto tijekom sastavljanja programskog koda. Modul sadrži bateriju s pomoću koje može duže vrijeme bez vanjskog napajanja nastaviti precizno mjeriti vrijeme. Da bi ostvario precizno računanje vremena, modul koristi linearni temperaturni senzor preciznosti ± 3 °C pomoću kojeg kalibrira svoja mjerenja.

Karakteristike:

- Radna temperatura: - 45°C – 80 °C
- Preciznost: ± 2 minute po godini
- Radni napon: 2.3 – 5.5 V
- Komunikacija: I2C



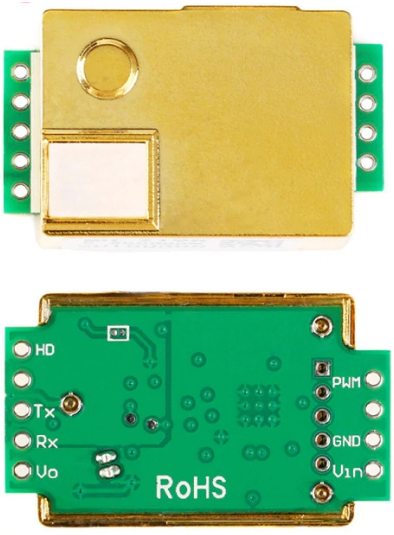
Slika 3.3. DS3231

3.1.4. MH-Z19B

MH-Z19B je senzor koji detektira prisutnost CO_2 čestica u zraku. Glavna komponenta senzora je infracrvena zraka koja se odašilje u komoru s uzorkom zraka prema detektoru. Paralelno se nalazi još jedna komora s referentnim plinom. Plin koji se nalazi u komori s uzorkom uzrokuje djelomičnu apsorpciju određenih valnih duljina infracrvene zrake. Detektor ispred sebe ima optički filter koji uklanja sve svjetlosne zrake osim one koju CO_2 može apsorbirati. Što je veća koncentracija CO_2 u komori, to je slabija zraka koja dođe do detektora.

Karakteristike:

- Raspon mjerenja: 0 – 10000 ppm
- Radna temperatura: 0 – 50 °C
- Vrijeme zagrijavanja: 3 minute
- Radni napon: 4.5 – 5.5 V



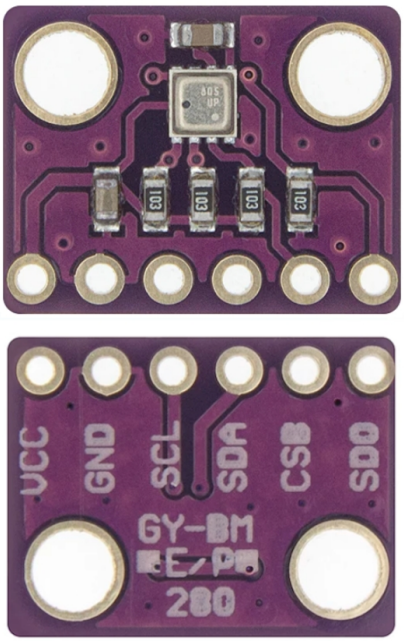
Slika 3.4. MH-Z19B

3.1.5. BME280

BME280 je modul čija je glavna komponenta Boschov senzor BME280 po kojem je modul dobio i ime. Senzor mjeri temperaturu, vlagu i tlak s velikom preciznošću.

Karakteristike:

- Raspon mjerenja temperature: - 40 °C – 85 °C
  - Preciznost: ± 1.0 °C
- Raspon mjerenja vlage u zraku: 0 % – 100 %
  - Preciznost: ± 3 %
- Raspon mjerenja atmosferskog tlaka: 300 – 1100 hPa
  - Preciznost: ± 3 hPa
- Radni napon: 1.7 – 3.6 V
- Komunikacija: I2C i SPI



Slika 3.5. BME280 senzor

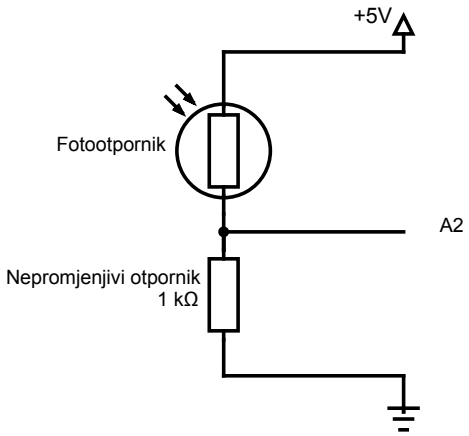
3.1.6. Fotootpornik

Fotootpornik je senzor kojim se detektira svjetlost. Otpor mu se mijenja ovisno o količini svjetlosti koja dopire do njega. U mraku ima najveći otpor, dok na najvećoj svjetlosti ima najmanji otpor. Preciznost mu nije jača strana, ali je jeftin i veoma izdržljiv. Iako se ne može koristiti za precizno određivanje svjetline izražene u luksima ili milikandelama, dobar je za prikaz grube vrijednosti kojom korisnik može dobiti dobru pretpostavku o jačini svjetlosti. Korisnik iz mjerenja može iščitati koje je doba dana i je li oblačno vrijeme ili nije.

Fotootpornik pretvara jakost svjetlosti u otpor. Pomoću Arduino Una nije moguće mjeriti otpor, ali je zato moguće mjeriti razinu napona. Fotootporniku se u seriju doda otpornik nepromjenjive veličine. Fotootpornik i nepromjenjivi otpornik se zajedno ponašaju kao potenciometar. Jedan pin fotootpornika se spaja na +5V, drugi se spaja na nepromjenjivi otpornik i analogni ulaz mikroupravljača. Mikroupravljač očitava razinu napona na analognom ulazu i analogno-digitalnim dekoderom ga pretvara u digitalnu vrijednost veličine cijelog broja od 0 do 1023.

Karakteristike:

- Radni napon: 5 V
- Otpor u potpunom mraku: 50 kΩ
- Otpor pri najvećoj svjetlosti: 500 Ω
- Očitana veličina na mikroupravljaču: 0 - 1023



Slika 3.6. Shema spajanja

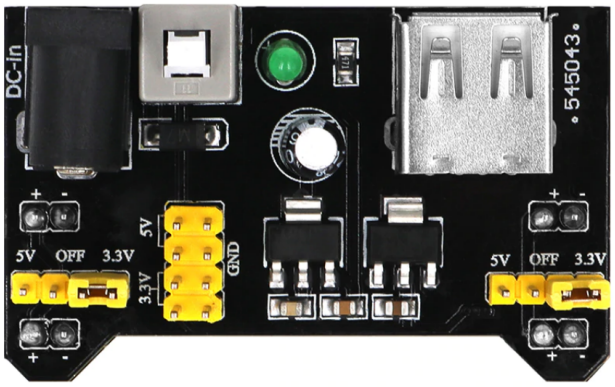
Slika 3.7. Fotootpornik

3.1.7. Modul za napajanje

Modul za napajanje služi za napajanje svih komponenti električnom energijom. Napon veličine 7 – 12 V pretvara u napon veličine 5 V i 3.3 V. Modul se može napajati preko cilindričnog ili USB priključka. Ima 2 pina na kojima je izlaz 5V i 2 pina na kojima je izlaz 3.3V. Dodatno ima još dva puta po dva pina na kojima je moguće odabrati izlazni napon između 5V i 3.3V. Modul ima i prekidač kojim se može paliti/gasiti cijeli uređaj.

Karakteristike:

- Ulazni napon: 7 – 12 V
- Izlazni napon: 5 i 3.3 V
- Pinovi:
  - 5 V: 2 komada
  - 3.3 V: 2 komada
  - 3.3 ili 5 V (po izboru): 2x2 komada



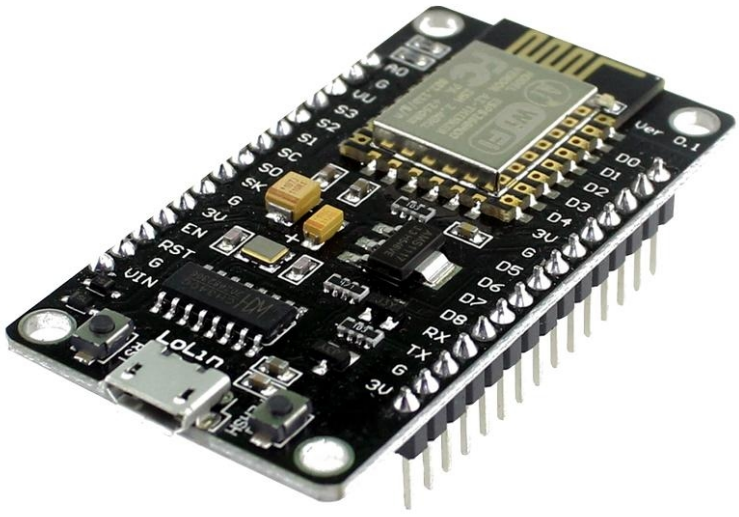
Slika 3.8. Modul za napajanje

3.1.8. NodeMCU

NodeMCU je mikroupravljač baziran na ESP8266 Wi-Fi modulu. Ne proizvodi ga tvrtka Arduino, ali ga je moguće programirati u ArduinoIDE programu. Zbog svoje niske cijene i dobrih karakteristika, veoma je popularan u IoT (Internet of Thing – Internet stvari) zajednici. Za razliku od Arduino Una, NodeMCU ima Wi-Fi modul na sebi što ga čini idealnom komponentom za sustave koji moraju biti povezani s internetom. Najveća mana ove platforme je broj analognih pinova. NodeMCU ima samo jedan analogni ulazni pin zbog čega nije pogodan za sustave koji imaju više od jedne komponente s analognim izlazom.

Karakteristike:

- Radni napon: 3.3 V
- Ulazni napon: 7 – 12 V
- Flash memorija: 4 MB
- SRAM: 64 KB
- EEPROM: 512 KB
- Wi-Fi antena: 2.4 GHz
  - Standard: IEEE 802.11 b/g/n
- Pinovi:
  - 16 digitalnih ulazno/izlaznih
  - 1 analogni ulazni



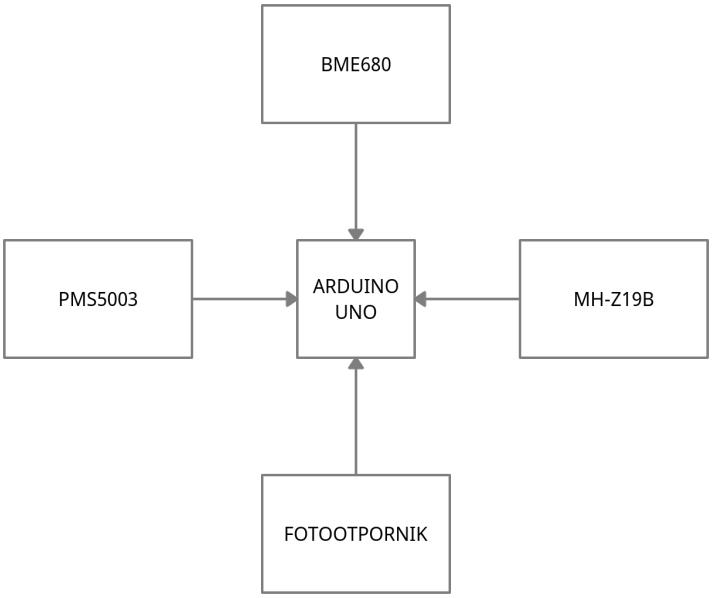
Slika 3.9. NodeMCU v3

3.2. Uređaj

Ovaj rad je takvog tipa da istovremeno mora biti izložen atmosferskim uvjetima, a i ne smije. Senzori moraju biti djelomično izloženi kako bi mogli mjeriti atmosferske uvijete, ali oni su na kraju krajeva ipak elektroničke komponente i ne smiju ostati nezaštićeni pred direktnim vremenskim nepogodama. Zbog toga se kućište dizajnira od plastičnih posudica oblika naopako okrenutih tanjura nasloženih jedan na drugog. Taj oblik dopušta cirkulaciju zraka kroz kućište, ali ne dopušta da voda dođe u kućište. Na taj način senzori mogu mjeriti atmosferske uvijete, a da pritom nisu direktno izloženi.

Mikroupravljač za razliku od senzora ne mora biti izložen vremenskim uvjetima. Da bi se osigurali čim bolji uvjeti rada mikroupravljača, on se stavlja u posebnu, zatvorenu kutiju. U kutiji se osim mikroupravljača nalaze i drugi moduli koji moraju biti čim bolje zaštićeni od atmosferskih uvjeta. Gornje kućište, u kojem se nalaze senzori, je metalnim vijcima spojeno s donjim kućištem. Iz gornjeg kućišta se sa senzora spuštaju žičice u donje kućište gdje se one spajaju na mikroupravljač i na modul za napajanje.

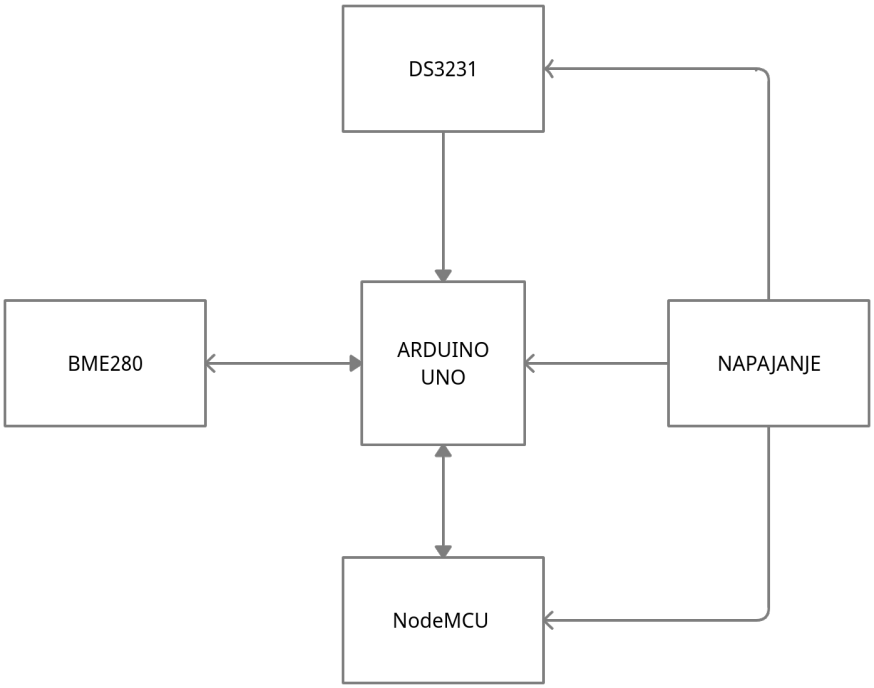
3.2.1. Vanjski dio uređaja



Slika 3.10. Blok shema vanjskog dijela uređaja

Vanjski dio uređaja sadrži sve senzore koji su potrebni za mjerenje atmosferskih uvjeta. Senzor BME680 mjeri temperaturu, vlagu zraka i atmosferski pritisak. MH-Z19B mjeri razinu CO_2 u zraku, dok PMS5003 mjeri koncentraciju sitnih čestica. Fotootpornik služi za mjerenje jačine svjetlosti. Nije namijenjen za precizno mjerenje, već da korisniku da određenu predodžbu o jačini svjetlosti.
Svi su senzori spojeni na unutarnji dio uređaja, odnosno na Arduino Uno koji očitava vrijednosti sa senzora.

3.2.2. Unutarnji dio uređaja



Slika 3.11. Blok shema unutarnjeg dijela uređaja

Unutarnji dio uređaja sadrži mikroupravljač Arduino Uno koji prikuplja podatke s vanjskog dijela uređaja, odnosno s vanjskih senzora. Osim podataka s vanjskih senzora, Uno prikuplja i podatke s DS3231 modula, koji služi za računanje stvarnog vremena. On je veoma bitan za kasniju obradu podataka jer svakoj izmjerenoj vrijednosti pridruži i vrijeme njenog mjerenja. Uno prikuplja podatke i s BME280 senzora koji mjeri temperaturu, vlagu i atmosferski pritisak. Njegova glavna svrha je prikupljanje podataka o radnim uvjetima

unutar samog kućišta. Ti se podatci koriste isključivo za nadgledanje radnih uvjeta unutarnjeg dijela uređaja. U unutarnjem dijelu uređaja se nalazi i modul za napajanje. Na njega su spojene sve komponente uključujući i mikroupravljač.

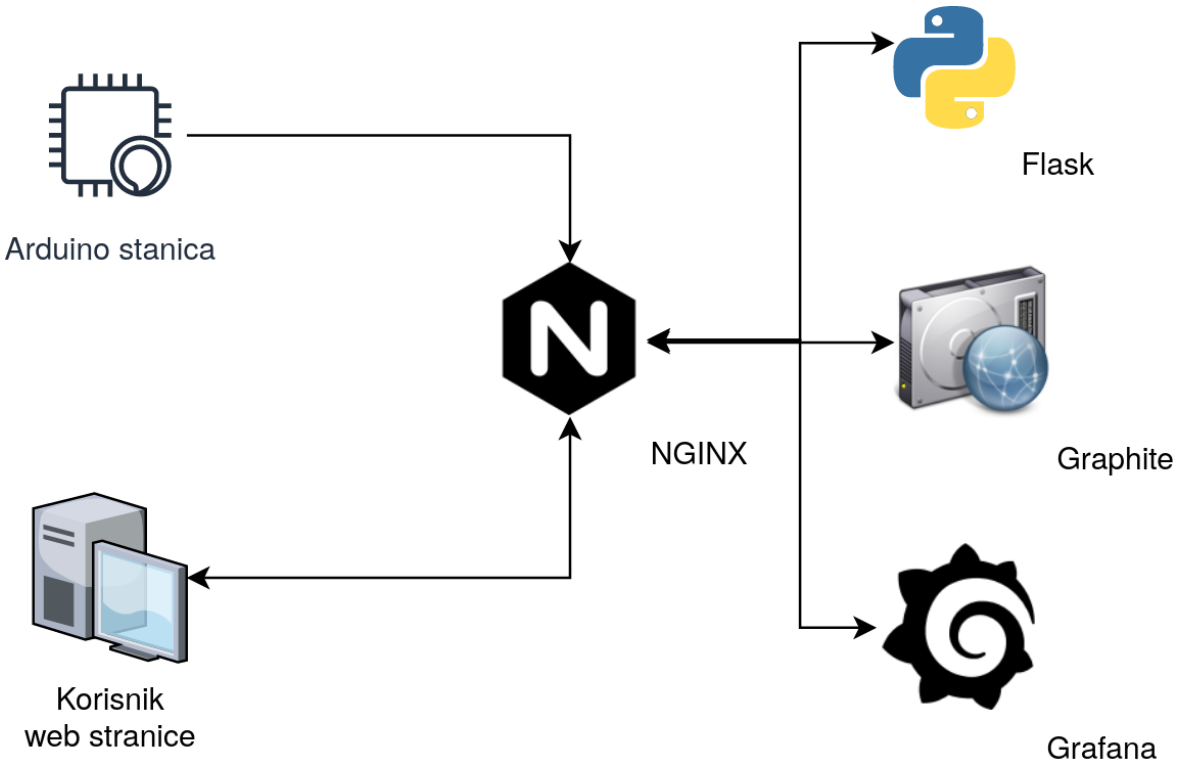
Nakon što Arduino Uno prikupi podatke sa svih uređaja, on ih sve zajedno proslijedi na NodeMCU. NodeMCU ostvari povezanost s internetom i proslijedi podatke na određenu ip adresu. Ukoliko ne može ostvariti povezanost s internetom, NodeMCU akumulira podatke sve dok se povezanost ponovno ne ostvari. Podatci se spremaju u kružnu memoriju veličine 20 polja. Nakon što se spremnik napuni, najstarija vrijednost se prebriše najnovijom. Ostvarivanjem povezanost s internetom, podatci se vade iz memorije po FIFO (First In, First Out) metodi. Odnosno najstariji zapisani podatak se prvi vadi iz memorije i prosljeđuje dalje.

3.3. Server

Uređaj prikupljene podatke šalje na IP adresu servera na port 5000 putem POST zahtjeva. Na portu 5000 nalazi se Flask koji zaprima POST zahtjev i iz njega vadi podatke. Program zatim dobivene podatke formatira i šalje ih na lokalni port 2003, gdje se nalazi Graphite aplikacija. Graphite zatim sprema te podatke u svoju bazu podataka. Na Graphite se spaja web aplikacija Grafana koja potom grafički prikazuje podatke koji se nalaze u Graphiteovoj bazi podataka.

Grafana se s Graphitom povezuje tako da se kroz Grafanino korisničko sučelje, koje je grafičkog tipa, kao izvor podataka odabere aplikacija Graphite i postavi odgovarajuća URL adresa. Nakon toga je moguće kreirati grafove u Grafani koja automatski prikuplja podatke iz Graphitove baze podataka.

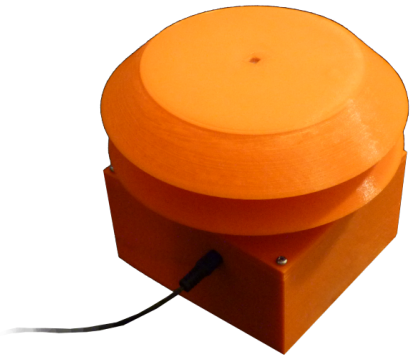
Krajnji korisnik kad ode na IP adresu servera, vidi grafički prikaz svih prikupljenih podataka. Korisničko sučelje je interaktivno. Korisnik može odabrati vremensko razdoblje za koje će se prikazivati podatci. Korisnik s administratorskim računom može izmjenjivati izgled stranice te odabrati koji će se podatci prikazivati i na koji način.



Slika 3.12. Arhitekturalni diagram sustava

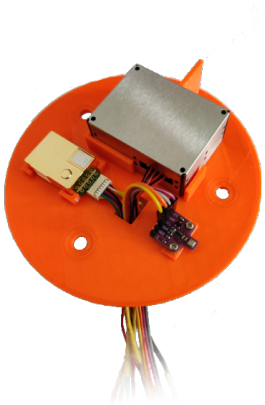
3.4. Izgled gotovog rada

3.4.1. Izgled uređaja



Vanjski izgled uređaja

Izgled unutarnjeg dijela uređaja iznutra



Izgled vanjskog dijela uređaja iznutra

3.4.2. Izgled web stranice

4. Dijelovi programskog koda

4.1. Arduino Uno

```
SoftwareSerial NodeMCU(2,3);
```

SoftwareSerial je softverska imitacija HardwareSeriala. Omogućuje da pojedini pinovi mikroupravljača služe za serijsku komunikaciju između mikroupravljača i neke druge komponente. U ovom slučaju spaja Arduino Uno i NodeMCU. Komunikacija je obostrana. Vrijednost 2 predstavlja ulazni pin D2 na Arduino Unu, a vrijednost 3 izlazni pin D3 koji se spaja na ulazni pin NodeMCU-a.

```
NodeMCU.begin(115200);
```

Naredba započinje komunikaciju mikroupravljača s NodeMCU modulom. Argument 115200 definira brzinu kojom će se prebacivati podatci između modula.

```
NodeMCU.end();
```

Naredba završava komunikaciju između mikroupravljača i modula.

```
StaticJsonDocument<300> doc;
…
serializeJson(doc, NodeMCU);
```

StaticJsonDocument stvori prazni dokument veličine 300 bajta naziva doc. Zatim se u dokument ubace željene vrijednosti koristeći format: „ doc[”ime”] = vrijednost ”. Nakon ubacivanja svih željenih vrijednosti, pokreće se naredba serializeJson();. Naredba kao argument uzima naziv datoteke i odredišnu lokaciju. U ovom slučaju uzima datoteku doc i šalje je na NodeMCU. Naredba u ovom slučaju služi samo za slanje podataka na drugi modul.

```
doc["wsid"] = "hr.10000.0";
```

Ova naredba sprema identifikacijski broj stanice u datoteku. Oznaka wsid predstavlja weather station id. Oznaka hr predstavlja oznaku države u kojoj se stanica nalazi, broj

10000 znači da se stanica nalazi u gradu Zagrebu i 0 predstavlja broj stanice na tom području.

```
doc["time"] = now.unixtime();
```

U datoteku se sprema vrijednost vremena u Unix vremenskom formatu. To je vrijednost vremena koja se neprestano broji još od početka 1970. godine.

```
// PMS5003
  unsigned long pm25_millis = millis();
  bool aqi_reading = true;
  pm25_Serial.begin(9600);
  PM25_AQI_Data data;
  do{
    delay(500);
    if(millis() >= pm25_millis+5000){
      aqi_reading = false;
      break;
    }
  }while(!aqi.read(&data));
  pm25_Serial.end();
```

Ovaj dio koda služi za mjerenje sitnih čestica. Prvo se započne komunikacija sa senzorom pomoću begin naredbe. Potom se pokušava dohvatiti podatke sa senzora. Zahtjev se šalje sve dok senzor ne pošalje podatke ili dok ne istekne vrijeme odbrojavanja veličine 5 sekundi.

```
 if(aqi_reading){
    doc["pm10"] = data.pm10_env;
    doc["pm25"] = data.pm25_env;
    doc["pm100"] = data.pm100_env;
    doc["pm_p03"] = data.particles_03um;
    doc["pm_p05"] = data.particles_05um;
    doc["pm_p10"] = data.particles_10um;
    doc["pm_p25"] = data.particles_25um;
    doc["pm_p50"] = data.particles_50um;
    doc["pm_p100"] = data.particles_100um;
  }
```

Ovaj dio programa zapisuje prethodno prikupljene podatke o sitnim česticama u datoteku, ukoliko je uvjet zadovoljen. Uvjet je zadovoljen tek kad senzor ispravno uspije obaviti mjerenje.

```
delay(30000);
```

Ovo je funkcija koja dok se izvršava ne dopušta niti jednoj drugoj naredbi da se izvršava. Odnosno izvršavanje programa je zaustavljeno na određeno vrijeme. U ovom programu argument je postavljen na 30000, što znači da će cijeli program biti pauziran na 30 sekundi. To znači da će uređaj svakih pola minute prikupiti podatke i poslati ih na server.

4.2. NodeMCU

```
#define ssid "ImeRutera"
#define password "SifraRutera"
```

Ovdje se upisuju podatci o mreži na koju će se modul spojiti. U navodnike gornje linije mora biti upisan naziv lokalne mreže (rutera), a u liniji ispod lozinka lokalne mreže.

```
#define serverName "http://65.21.1.49:5000/"
```

Ova linija koda definira IP adresu na koju će se slati prikupljeni podatci.

```
#define DATA_SIZE 600
#define QUEUE_SIZE 20
```

DATA_SIZE definira veličinu podataka pristiglih s Arduino Una. Veličina je izražena u bajtima. Vrijednost je navedena na početku programa da bi se kasnije mogla lakše promijeniti kad se uređaj bude nadograđivao s dodatnim senzorima.

QUEUE_SIZE definira broj polja veličine DATA_SIZE. Koristi se kod akumulatora gdje definira koliko će se podataka privremeno spremiti ukoliko se ne može ostvariti povezanost s internetom. Trenutno je definirana veličina od 20 polja, što znači da ukoliko pukne povezanost s internetom, podatci će se 20 puta spremiti prije nego što nastane prvi gubitak podataka. Ako se svaki program izvršava 30 sekundi, to znači da uređaj može biti odspojen 10 minuta, bez gubitaka podataka.

4.3. Server

```
import json
import os

from flask import Flask, request
```

Ovaj dio koda pribavlja module u program. Json modul služi za pretvaranje određenog tipa podataka u Json tip podataka. Os modul omogućuje korištenje određenih sistemskih naredbi unutar samog programa.

```
@app.route('/', methods=['GET', 'POST'])
def home():
    data = request.data
    json_data = json.loads(data)
    send_dict_to_graphite(json_data)
    return "OK", 200
```

Za ovaj dio programa se brine Flask. On prihvati podatke koje uređaj pošalje i ti se podatci zatim spreme u varijablu “data”. Pomoću json modula se ti podatci preoblikuju i zatim proslijede u funkciju na daljnju obradu.

```
def send_dict_to_graphite(in_dict):
    for key,value in in_dict.items():
        if key != 'wsid' and key != 'time':
            metric = f"{in_dict['wsid']}.{key}"
            timestamp = int(in_dict['time'])
            cmd = f'echo "{metric} {value} {timestamp} " | nc -q0 65.21.1.49 2003'
            os.system(cmd)
```

U funkciji se u for petlji obrađuje jedna po jedna vrijednost. Jedine vrijednosti koje se ne diraju su “wsid” i “time”. Ostale vrijednosti predstavljaju očitane vrijednosti sa senzora i svakoj se vrijednosti doda vrijednost “wsid”. Odnosno doda se identifikacijski broj stanice s koje je podatak prikupljen. To omogućuje da se s više uređaja spremaju podatci u bazu podataka, bez da dođe do miješanja podataka. Pokretanjem naredbe os.system(cmd), podatci se pošalju u bazu podataka.

5. Zaključak

Izrada ovog rada bio je veoma zanimljiv proces. Dugotrajan i zahtjevan, ali zanimljiv. Iako sam potrošio više desetaka sati na njegovu izradu, sada kada je gotov, mislim da se je isplatilo.

Rad objedinjuje hardver i softver, što osobno smatram da je veoma zanimljivo. Gotovi proizvod je uređaj koji se samo spoji na izvor napajanja i on automatski počne prikupljati podatke i oni se počnu prikazivati u obliku grafova. Zvuči veoma jednostavno. To je i bio cilj ovog rada. Napraviti sustav koji krajnjem korisniku omogućuje čim jednostavnije korištenje.

Osim gotove meteorološke postaje koja će mi svakako biti korisna, završetkom ovog rada sam dobio i neka nova znanja. Značajno sam produbio svoje znanje vezano uz Arduino platformu. Iako i dalje ne znam ni približno sve, imam dovoljno predznanja u ovom području da uz kraće pretraživanje interneta mogu veoma lako pronaći odgovore na ono što mi je potrebno. Osim Arduino platforme, dotaknuo sam se malo i Python programskog jezika prilikom izrade programa za server. Za potrebe ovog rada sam se bavio i izradom 3D modela. Kako je bilo potrebno dizajnirati kućište po mjeri, bio sam prisiljen naučiti koristiti FreeCAD alat za izradu 3D modela. Također sam naučio i ponešto o printanju 3D modela.

Što se tiče poboljšanja sustava, mislim da ima mjesta za poboljšanje. Svakako bih nadodao komponente koje mjere brzinu vjetra, smjer vjetra i količinu padalina, što na žalost nije uspjelo ući u sadržaj ovog rada. Također bih dodao senzor koji preciznije mjeri svjetlost i senzor koji bi mjerio razinu buke.

Sve u svemu, ovo je bio jedan koristan i zanimljiv projekt u kojem sam koristio sve svoje znanje stečeno za vrijeme školovanja u ovoj školi. Nadam se da ću i u budućnosti imati prilike raditi na sličnim projektima

6. Literatura

1. Arduino Uno (pristupljeno: 23.4.2021.)

   https://www.farnell.com/datasheets/1682209.pdf

2. Graphite vs. Grafana: Build the Best Monitoring Architecture for Your Application (pristupljeno: 25.4.2021.)

   https://www.overops.com/blog/graphite-vs-grafana-build-the-best-monitoring-architecture-for-your-application/

3. Graphite (pristupljeno: 27.4.2021.)

   https://graphiteapp.org/

4. Plantower PM5003 Air Quaility Sensor (pristupljeno: 24.4.2021.)

   https://nettigo.eu/products/plantower-pms5003-air-quality-sensor

5. Digital universal particle concentration sensor (pristupljeno: 27.4.2021.)

   https://cdn-shop.adafruit.com/product-files/3686/plantower-pms5003-manual_v2-3.pdf

6. PM2.5 Air Quality Sensor (pristupljeno: 23.4.2021.)

   https://learn.adafruit.com/pm25-air-quality-sensor

7. Guide for BME680 Environmental Sensor with Arduino (Gas, Temperature, Humidity, Pressure) (pristupljeno: 27.4.2021.)

   https://randomnerdtutorials.com/bme680-sensor-arduino-gas-temperature-humidity-pressure/

8. DS3231 RTC Module (pristupljeno: 27.4.2021.)

   https://components101.com/modules/ds3231-rtc-module-pinout-circuit-datasheet

9. Intelligent Infrared CO2 Module (pristupljeno: 24.4.2021.)

   https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19b-co2-ver1_0.pdf

10. BME280 Sensor with Arduino Tutorial (pristupljeno 27.4.2021.)

https://www.learnrobotics.org/blog/bme280-arduino-tutorial/

11. Interface BME280 Temperature, Humidity & Pressure Sensor with Arduino (pristupljeno: 26.4.2021.)

https://lastminuteengineers.com/bme280-arduino-tutorial/

12. Photocells (pristupljeno: 25.4.2021.)

https://learn.adafruit.com/photocells

13. Breadboard Power Supply Module (pristupljeno: 27.4.2021.)

https://www.aliexpress.com/item/32725717757.html

14. NodeMCU ESP8266 (pristupljeno: 25.4.201.)

https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

6.1. Popis slika

Slika 2.1. preuzeta s: https://www.crazypi.com/arduino-uno-r3-microcontroller

Slika 2.2. preuzeta s: https://arduino-forth.com/article/arduino_comprendreConnecteurs

Slika 2.3. preuzeta s: https://www.overops.com/blog/graphite-vs-grafana-build-the-best-monitoring-architecture-for-your-application/

Slika 3.1. preuzeta s: https://aqicn.org/sensor/pms5003-7003/vn/

Slika 3.2. preuzeta s: https://hallroad.org/images/detailed/12/CJMCU-680_BME680_BOSCH_Temperature_And_Humidity_Pressure_Sensor_Ultra-small_Pressure_Height_Development_Board_1.JPG

Slika 3.3. preuzeta s: https://xenyltechbd.com/wp-content/uploads/2020/01/DS3231-RTC-2.png

Slika 3.4. preuzeta s: https://ae01.alicdn.com/kf/HTB10Ow3TrvpK1RjSZFqq6AXUVXaq/MH-Z19-Infrared-CO2-Sensor-Module-MH-Z19B-Carbon-Dioxide-Gas-Sensor-for-CO2-Monitor-0.jpg

Slika 3.5. preuzeta s: https://ae01.alicdn.com/kf/Hdb3da8527ad14ce3b91ac0f928740d2eo/GY-BME280-3-3-precision-altimeter-atmospheric-pressure-BME280-sensor-module.jpg

Slika 3.7. preuzeta s: https://storage.googleapis.com/production-public-files/public/system/images/photos/000/013/316/original/09088-02-L.jpg

Slika 3.8. preuzeta s: https://www.aliexpress.com/item/32725717757.html

Slika 3.9. preuzeta s: https://opencircuit.shop/Product/NodeMcu-v3-Lua-ESP-12E-WIFI-Development-Board

7. Prilozi

7.1. Cijeli programski kod

7.1.1. Arduino Uno

```
#include "Adafruit_PM25AQI.h"
#include <SoftwareSerial.h>
#include "ArduinoJson.h"
#include <Wire.h>
#include "Adafruit_Sensor.h"
#include "Adafruit_BME680.h"
#include "Adafruit_BME280.h"
#include "RTClib.h"

// PMS5003
SoftwareSerial pm25_Serial(5, 6);  // Rx, Tx  - Rx ne koristimo
Adafruit_PM25AQI aqi = Adafruit_PM25AQI();

// MH-Z19B
SoftwareSerial co2_Serial(A0, A1); // RX, TX
byte cmd[9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79};
unsigned char response[9];
unsigned long ppm;

// BME680 - I2C komunikacija
Adafruit_BME680 bme680;

// BME280 - SPI komunikacija
#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10
Adafruit_BME280 bme280(BME_CS);

// NodeMCU
SoftwareSerial NodeMCU(2,3);  // RX, TX

// Modul stvarnog vremena
RTC_DS3231 rtc;

void setup() {
  // PMS5003
  pm25_Serial.begin(9600);
  while(! aqi.begin_UART(&pm25_Serial)){
    delay(100);
  }

  // MH-Z19B
  co2_Serial.begin(9600);

  // NodeMCU
  NodeMCU.begin(115200);
```

```
  // BME680
  while(!bme680.begin()){
    delay(100);
  }
  // Konfiguracija senzora
  bme680.setTemperatureOversampling(BME680_OS_8X);
  bme680.setHumidityOversampling(BME680_OS_2X);
  bme680.setPressureOversampling(BME680_OS_4X);
  bme680.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme680.setGasHeater(320, 150); // 320*C na 150 ms

  // BME280
  bme280.begin();

  // Svjetlina
  pinMode(A2, INPUT);

  // Sat
  while(! rtc.begin()){
    delay(100);
  }

  // lostPower() funkcija isčitava vrijednost unutarnjeg registra i provjerava da li je izgubio
  // pojam vremena
  If (rtc.lostPower()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
}

void loop(){
  // PMS5003
  unsigned long pm25_millis = millis();
  bool aqi_reading = true;
  pm25_Serial.begin(9600);
  PM25_AQI_Data data;
  do{
    delay(500);
    if(millis() >= pm25_millis+5000){
      aqi_reading = false;
      break;
    }
  }while(!aqi.read(&data));
  pm25_Serial.end();


  // MH-Z19B
  co2_Serial.begin(9600);
  co2_Serial.write(cmd,9);
  co2_Serial.readBytes(response, 9);
  co2_Serial.end();
  unsigned int responseHigh = (unsigned int) response[2];
  unsigned int responseLow = (unsigned int) response[3];
  ppm = (256*responseHigh)+responseLow;


  // BME680
  bme680.performReading();
```

```
  // Fotootpornik
  int svjetlina = analogRead(A2);

  // Sat
  DateTime now = rtc.now();


  // NodeMCU
  NodeMCU.begin(115200);
  StaticJsonDocument<300> doc;
  doc["wsid"] = "hr.10000.0";

  doc["time"] = now.unixtime();

  if(aqi_reading){
    doc["pm10"] = data.pm10_env;
    doc["pm25"] = data.pm25_env;
    doc["pm100"] = data.pm100_env;
    doc["pm_p03"] = data.particles_03um;
    doc["pm_p05"] = data.particles_05um;
    doc["pm_p10"] = data.particles_10um;
    doc["pm_p25"] = data.particles_25um;
    doc["pm_p50"] = data.particles_50um;
    doc["pm_p100"] = data.particles_100um;
  }

  doc["ppm"] = ppm;

  doc["temperature"] = bme680.temperature;
  doc["pressure"] = bme680.pressure / 100.0F;
  doc["humidity"] = bme680.humidity;
  doc["gas"] = bme680.gas_resistance / 1000.0F;

  doc["brightness"] = svjetlina;

  doc["s_temp"] = bme280.readTemperature();
  doc["s_pressure"] = bme280.readPressure() / 100.0F;
  doc["s_hum"] = bme280.readHumidity();
  serializeJson(doc, NodeMCU);
  NodeMCU.end();

  delay(30000);
}
```

7.1.2. NodeMCU

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <SoftwareSerial.h>
#include "ArduinoJson.h"
#include <MD_CirQueue.h>

// naziv i lozinka WiFi mreže
#define ssid "ImeRutera"
#define password "SifraRutera"

// IP Adresa servera
#define serverName "http://65.21.1.49:5000/"

// Veličina podataka pristiglih s Arduina
#define DATA_SIZE 600

// Broj kopija u akumulatoru
#define QUEUE_SIZE 20

// Definira akumulator
MD_CirQueue Q(QUEUE_SIZE, DATA_SIZE);


SoftwareSerial Uno(4,5);  //Rx, Tx

void setup() {

  // Komunikacija s Arduino Uno
  Uno.begin(115200);

  // Pokretanje WiFi-a
  WiFi.begin(ssid, password);
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
  }

  // Inicijalizacija akumulatora
  Q.begin();
  //  push() funkcija će prebrisati najstariju vrijednost u akumulatoru novom
  Q.setFullOverwrite(true);
}

void loop() {
  // Ako postoji povezanost s internetom i akumulator nije prazan, onda ga isprazni
  if(WiFi.status()== WL_CONNECTED && !Q.isEmpty()){
    HTTPClient http;
    http.begin(serverName);
    http.addHeader("Content-Type", "application/json");
    while(!Q.isEmpty()){
      char buffer_data[DATA_SIZE];
      Q.pop((uint8_t *)&buffer_data);
      http.POST(buffer_data);
    }
    http.end();
  }
```

```
  // Ako je Arduino Uno poslao neke podatke
  if(Uno.available()){

    // Osigurana veličina prostora za podatke
    StaticJsonDocument<DATA_SIZE> doc;

    // Provjeriti da li je deserijalizacija uspješno obavljena
    DeserializationError err = deserializeJson(doc, Uno);

    // Ako je uspješno obavljena:
    if(err == DeserializationError::Ok){

      // Varijabla char tipa u koju se spremaju podatci s Una
      char sensor_data[DATA_SIZE];
      // Serijaliziramo "doc" u char varijablu sensor_data
      serializeJson(doc, sensor_data);


      //Provjerava povezanost s WiFi uređajem
      if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;

        http.begin(serverName);

        // HTTP zahtjev s podatcima tipa json
        http.addHeader("Content-Type", "application/json");
        // Šalje HTTP POST zahtjev
        http.POST(sensor_data);

        // Oslobađaju se resursi
        http.end();
      }
      // Ukoliko je pukla povezanost s internetom
      else{
        // Podatci se spremaju u akumulator
        Q.push((uint8_t *)&sensor_data);
      }
    }
    // Ako deserijalizacija nije uspješno obavljena:
    else{
      // Brišu se svi podatci iz memorije za serijsku komunikaciju, kako bi novi
      // mogli doći
      while (Uno.available() > 0){
        Uno.read();
      }
    }
  }
}
```

7.1.3. Server

```
import json
import os

from flask import Flask, request

app = Flask(__name__)


def send_dict_to_graphite(in_dict):
    for key,value in in_dict.items():
        if key != 'wsid' and key != 'time':
            metric = f"{in_dict['wsid']}.{key}"
            timestamp = int(in_dict['time'])
            cmd = f'echo "{metric} {value} {timestamp} " | nc -q0 65.21.1.49 2003'
            os.system(cmd)


@app.route('/', methods=['GET', 'POST'])
def home():
    data = request.data
    json_data = json.loads(data)
    send_dict_to_graphite(json_data)
    return "OK", 200


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```