SVEUČILIŠTE J.J. STROSSMAYER U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKH TEHNOLOGIJA OSIJEK

Sveučilišni studij

UGRADBENI RAČUNALNI SUSTAVI

Profesor/mentor: doc.dr.sc. Tomislav Keser

TEHNIČKA DOKUMENTACIJA PROJEKTA

SUSTAV ZA MJERENJE KVALITETE ZRAKA

Ilija Jazvić

Akademska godina 2023/2024

Osijek, 15. 6. 2024.

Sadržaj

1. Uvod 2
   1.1. Zadatak i struktura rada 3
2. Sustav za mjerenje kvalitete zraka 5
   2.1. Teorijski osvrt na sustav za mjerenje kvalitete zraka 5
   2.2. Prijedlog sklopovskog riješenja 7
   2.3. Prijedlog programskog riješenja 10
3. Realizacija sustava za mjerenje kvalitete zraka 11
   3.1. Korištene komponente, alati i programska podrška 11
   3.2. Realizacija konstrukcijskog i sklopovskog rješenja 13
   3.3. Realizacija programskog rješenja 16
4. Testiranje i rezultati 22
   4.1. Metodologija testiranja 22
      Provedba ispravnosti mikroupravljača 22
      Testiranje sustava (programskog koda i senzora) 22
      Intepretacija rezultata 23
   4.2. Rezultati testiranja 24
      Testiranje mikroupravljača i Adafruit web servera 24
      Rezultati promjene temperature i vlažnosti zraka 25
      Rezultati promjene koncentracije plinova u prostoriji 27
5. Zaključak 33
6. Literatura 34
Prilog A: Programski kod 35

1. Uvod

Kvaliteta zraka postaje sve važnija tema u kontekstu održivog razvoja i zaštite zdravlja ljudi. Brz industrijski razvoj, urbanizacija i povećanje broja vozila u gradovima značajno doprinose zagađenju zraka. U takvim uvjetima, mjerenje i kontrola kvalitete zraka postaju ključni za osiguranje zdravog okruženja. Postoje brojna rješenja za mjerenje i kontrolu kvalitete zraka, koja koriste različite senzore i metode za detekciju zagađivača i upravljanje sustavima za ventilaciju i gašenje požara.

Jedan od primjera takvih sustava su mreže senzora za praćenje kvalitete zraka koje koriste različite senzore za mjerenje koncentracija čestica, razine CO_2, temperature, vlage i drugih parametara. Ovi sustavi su široko rasprostranjeni u urbanim sredinama, industrijskim postrojenjima i zatvorenim prostorima poput ureda i stambenih zgrada. Njihova korisnost leži u mogućnosti pravovremene reakcije na povećane razine zagađenja, čime se smanjuje izloženost štetnim tvarima i poboljšava kvaliteta zraka.

Sustavi za mjerenje kvalitete zraka često koriste senzore čestica koji detektiraju prisutnost i koncentraciju čestica u zraku. Ovi senzori koriste različite tehnologije, poput laserske difrakcije, za precizno mjerenje koncentracije čestica različitih veličina. Uz to, senzori za mjerenje tlaka, temperature i vlage omogućuju praćenje dodatnih parametara koji utječu na kvalitetu zraka. Senzori dima su također ključni za detekciju dima i potencijalno opasnih situacija koje mogu dovesti do požara. Jedan od glavnih izazova u dizajniranju sustava za mjerenje kvalitete zraka je integracija različitih senzora i obrada velikih količina podataka u stvarnom vremenu. Moderni sustavi često koriste mikroupravljače poput ESP32, koji omogućuju učinkovitu obradu podataka, povezivanje s mrežom i postavljanje web servera za prikaz podataka. Korištenje grafova za vizualizaciju izmjerenih vrijednosti omogućuje korisnicima lakše praćenje promjena u kvaliteti zraka i donošenje informiranih odluka o potrebnim mjerama.

Motivacija za odabir ovakvog sustava leži u želji za stvaranjem rješenja koje ne samo da mjeri i prikazuje podatke o kvaliteti zraka, već i aktivno reagira na promjene u okolini. Automatsko upravljanje ventilatorom i sustavom za gašenje požara na temelju izmjerenih vrijednosti osigurava brzo i učinkovito smanjenje rizika od zagađenja i požara. Uz to, mogućnost manualnog upravljanja sustavom pruža korisnicima dodatnu fleksibilnost i kontrolu nad okolinom. Na ovaj način, sustav ne samo da doprinosi poboljšanju kvalitete zraka, već i povećava sigurnost i udobnost prostora u kojem se koristi.

1.1. Zadatak i struktura rada

Cilj ovog projekta je razviti sustav za mjerenje i upravljanje kvalitetom zraka u zatvorenim prostorima koristeći kombinaciju senzora za praćenje različitih parametara okoliša. Sustav koristi senzor MQ135 za mjerenje kvalitete zraka (posebno koncentracije CO_2), DHT22 senzor za mjerenje temperature i vlažnosti, te BMP180 senzor za mjerenje tlaka i nadmorske visine. Podaci prikupljeni ovim senzorima se zatim koriste za automatsko upravljanje ventilatorom i sustavom za gašenje požara (prskalice za vodu), s ciljem održavanja optimalne kvalitete zraka i sigurnosti prostora. Zbog prevelike cijene nije bilo moguće implementirati sustav za gašenje požara odnosno prskalice za vodu te su one reprezentirane kao LED-ica koja svijetli ovisno o stanju prskalica.

Očekivani rezultati projekta uključuju:

1. **Praćenje kvalitete zraka:** Sustav će kontinuirano pratiti razine CO_2, temperature, vlažnosti i tlaka zraka. Podaci će biti prikazani na web sučelju putem ESP32 mikroupravljača i na Arduino IDE *serial monitor*-u (hrv. serijski monitor, sučelje za prikaz rezultata), omogućujući korisnicima pregled trenutnih vrijednosti i povijesti mjerenja za posljednjih 7 dana.
2. **Automatsko upravljanje ventilatorom i sustavom za gašenje požara:** Na temelju izmjerenih vrijednosti kvalitete zraka, sustav će automatski upravljati ventilatorom. Ako koncentracija CO_2 pređe određeni prag, ventilator će se automatski uključiti kako bi se poboljšala kvaliteta zraka, te ako se dostigne određena temperatura zraka i određena koncentracija CO_2 uključit će se sustav za gašenje požara.
3. **Sigurnosne mjere:** U slučaju da senzor MQ135 detektira visoke razine zagađenja ili dima, sustav će automatski aktivirati relej za pokretanje ventilatora ili prskalica za gašenje požara, ovisno o situaciji. Također, korisnici će imati mogućnost ručnog upravljanja sustavom putem gumba za uključivanje/isključivanje prskalica.
4. **Integracija s Adafruit IO:** Podaci će se slati na Adafruit IO platformu, omogućujući daljinski pristup i praćenje mjerenja u stvarnom vremenu.

Na kraju, projekt će osigurati sveobuhvatan sustav za poboljšanje kvalitete zraka u zatvorenim prostorima, kombinirajući automatsku kontrolu, sigurnosne mjere i jednostavnu vizualizaciju podataka za korisnike. Ovaj sustav će biti koristan za različite aplikacije, uključujući stambene prostore, urede, industrijske objekte i druge zatvorene prostore gdje je kvaliteta zraka od ključne važnosti.

Dokumentacija je organizirana u sljedeća poglavlja:

Uvod - Ovo poglavlje uvodi čitatelja u problematiku kvalitete zraka, važnost praćenja i kontrole kvalitete zraka u zatvorenim prostorima te motivaciju i cilj ovog projekta.

Sustav za mjerenje kvalitete zraka - Poglavlje obuhvaća teorijski osvrt na sustave za mjerenje kvalitete zraka, uključujući pregled postojećih sustava i tehnologija. Također se daje prijedlog sklopovskog i programskog rješenja za razvijeni sustav.

Teorijski osvrt na sustav za mjerenje kvalitete zraka - Diskutira se o fizikalnim principima, kemijskim reakcijama i algoritmima relevantnim za mjerenje kvalitete zraka.

Prijedlog sklopovskog rješenja - Predstavlja se funkcionalni blok dijagram sustava i objašnjavaju se ključne komponente i njihovi međusobni odnosi.

Prijedlog programskog rješenja - Prikazuje se osnovni blok dijagram algoritma i objašnjavaju se koraci u prikupljanju, obradi i analizi podataka te donošenju odluka.

Realizacija sustava za mjerenje kvalitete zraka - Ovo poglavlje detaljno opisuje praktičnu realizaciju sustava, uključujući korištene komponente, alate i programsku podršku, kao i korake u izgradnji hardverskog i softverskog dijela sustava.

Korištene komponente, alati i programska podrška - Opisuje se hardver i softver korišten u projektu.

Realizacija konstrukcijskog i sklopovskog rješenja - Detaljno se opisuje izrada hardverskog dijela sustava.

Realizacija programskog rješenja - Opisuje se implementacija softverskog dijela sustava, uključujući čitanje podataka sa senzora i upravljanje aktuatorima.

Testiranje i rezultati - U ovom poglavlju se opisuje metodologija testiranja sustava te se prikazuju i analiziraju dobiveni rezultati.

Metodologija testiranja - Opisuje se pristup i koraci u testiranju sustava.

Rezultati testiranja - Prikazuju se i analiziraju rezultati dobiveni tijekom testiranja sustava.

Zaključak - Zaključno poglavlje sumira ključne aspekte projekta, uključujući postignute rezultate te se diskutira o mogućim poboljšanjima i budućim radovima na sustavu.

2. Sustav za mjerenje kvalitete zraka

Zatvoreni prostori često imaju lošu kvalitetu zraka zbog nakupljanja CO_2, vlage i drugih zagađivača. Tradicionalni sustavi za ventilaciju često nisu dovoljno učinkoviti u održavanju optimalne kvalitete zraka što može dovesti do nelagode i zdravstvenih problema za ljude koji borave u tim prostorima.

Problem:

- Loša kvaliteta zraka može uzrokovati zdravstvene probleme.
- Tradicionalni sustavi za ventilaciju često ne osiguravaju dovoljno svježeg zraka.
- Nema mogućnosti praćenja kvalitete zraka u stvarnom vremenu.

Na tržištu postoje različiti sustavi za praćenje kvalitete zraka, ali većina njih je komercijalna i skupa. Neki od njih uključuju:

- **Netatmo Healthy Home Coach:** Sustav za praćenje kvalitete zraka koji koristi više senzora za mjerenje CO_2, vlažnosti, temperature i buke. Prikazuje podatke putem mobilne aplikacije.
- **Awair:** Napredni uređaj za praćenje kvalitete zraka koji mjeri prašinu, kemikalije, CO_2, temperaturu i vlažnost. Također nudi mobilnu aplikaciju za praćenje podataka.
- **Foobot:** Uređaj koji mjeri čestice, kemikalije, temperaturu i vlažnost te pruža preporuke za poboljšanje kvalitete zraka putem mobilne aplikacije.

2.1. Teorijski osvrt na sustav za mjerenje kvalitete zraka

Funkcioniranje sustava za praćenje i regulaciju kvalitete zraka temelji se na nekoliko ključnih fizikalnih zakonitosti i principa, uključujući kemijske reakcije, termodinamiku, hidrodinamiku i algoritamske procese za obradu i analizu podataka.

Ključne fizikalne zakonitosti i principi

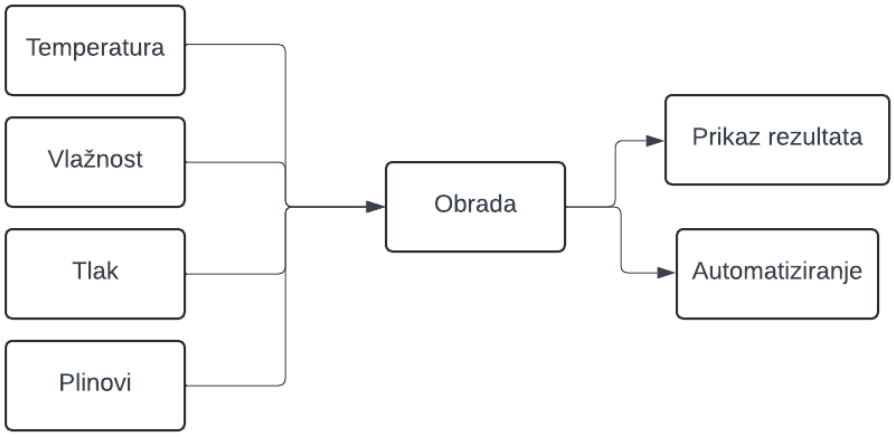
1. **Kemijske reakcije:**
    - Senzori za plinove koriste principe kemijske reaktivnosti za detekciju prisutnosti određenih plinova u zraku. Na primjer, MQ135 senzor koristi metal-oksid poluvodički sloj koji mijenja svoj električni otpor kada je izložen različitim koncentracijama plinova kao što su CO_2, NH_3, benzen, alkohol, dim i druge tvari. [1], [2]

2. **Termodinamika:**
    - Termodinamički principi upravljaju ponašanjem temperature i vlažnosti zraka. Senzori koriste kapacitivni senzor za mjerenje vlažnosti i termistor za mjerenje temperature, pružajući podatke potrebne za regulaciju mikroklime u prostoru.
3. **Hidrodinamika:**
    - Protok zraka i disperzija plinova unutar zatvorenog prostora slijedi zakone hidrodinamike. Ventilacijski sustav koristi ove principe kako bi osigurao ravnomjernu distribuciju svježeg zraka i uklanjanje zagađenog zraka.
4. **Algoritamski principi:**
    - Algoritmi za obradu podataka koriste uzročno-posljedične odnose za donošenje odluka temeljenih na mjerenim vrijednostima. Na primjer, kada koncentracija CO_2 premaši određeni prag, algoritam automatski aktivira ventilator kako bi smanjio koncentraciju plina.

Metodologije efekata

1. **Praćenje kvalitete zraka:**
    - Senzori kontinuirano prikupljaju podatke o koncentraciji plinova, temperaturi, vlažnosti i tlaku zraka.
    - Prikupljeni podaci se periodično šalju na centralnu jedinicu za obradu (Mikroupravljač) gdje se analiziraju i pohranjuju.
2. **Analiza i obrada podataka:**
    - Algoritmi analiziraju trendove u podacima kako bi identificirali potencijalne probleme s kvalitetom zraka.
    - Podaci se uspoređuju s unaprijed definiranim pragovima za kvalitetu zraka.
3. **Automatska regulacija:**
    - Na temelju analize, sustav automatski upravlja ventilacijom i prskalicama kako bi održao optimalnu kvalitetu zraka.
    - Ventilator se uključuje kada koncentracija CO_2 premaši određeni prag, a prskalice se aktiviraju u slučaju detekcije određene koncentracije čestica i temperature zraka.
4. **Prikaz podataka:**
    - Mikroupravljač šalje podatke na web sučelje gdje korisnici mogu pratiti trenutne vrijednosti i povijesne trendove.

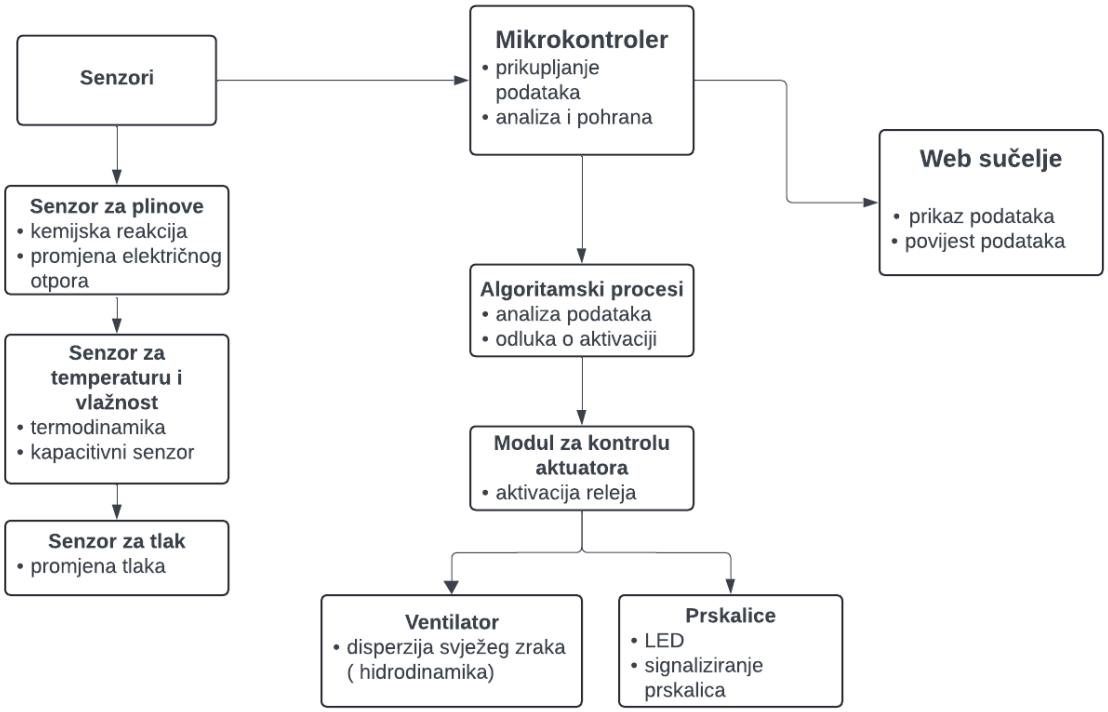
Sljedeća skica prikazuje osnovni koncept sustava za praćenje i regulaciju kvalitete zraka.



***Slika 2.1**. Osnovni koncept sustava za mjerenje kvalitete zraka*

2.2. Prijedlog sklopovskog riješenja

U ovom dijelu prikazan je funkcionalni blok dijagram sustava za praćenje i regulaciju kvalitete zraka. Na dijagramu su prikazane ključne komponente sustava i njihovi međusobni odnosi te je dano detaljno objašnjenje kako te komponente djeluju jedna na drugu i kako utječu na cjelokupni sustav.



***Slika 2.2.** Blok dijagram sustava*

Senzorski modul

Senzorski modul se sastoji od različitih senzora koji mjere kvalitetu zraka, temperaturu, vlažnost, tlak i prisutnost dima. Svaki senzor generira analogne ili digitalne signale koji predstavljaju izmjerene vrijednosti.

Komponente:

- Senzor za plinove i dim
- Senzor za temperaturu i vlažnost
- Senzor za tlak

Svi senzori su povezani s glavnim mikroupravljačem koji prikuplja njihove podatke. Senzori šalju svoje podatke mikroupravljaču. Analogni signali iz senzora plinova, temperature i vlažnosti prolaze kroz ADC pretvarače unutar mikroupravljača koji ih pretvara u digitalne podatke za obradu.

Mikroupravljač

Mikroupravljač je središnja jedinica sustava koja prikuplja podatke sa senzora, obrađuje te podatke i donosi odluke o aktivaciji ostalih modula sustava.

Komponente:

- Analogno-digitalni pretvarači (ADC) za senzore koji šalju analogne signale
- Ulazi za digitalne signale
- Procesorska jedinica za obradu podataka

Mikroupravljač je povezan sa svim senzorima te s modulom za prikaz podataka i modulom za kontrolu aktuatora.

Modul za prikaz podataka

Modul za prikaz podataka omogućava korisniku pregled izmjerenih vrijednosti u stvarnom vremenu putem web sučelja.

Komponente:

- ESP32 za bežičnu komunikaciju i web server

Modul za prikaz podataka prima obrađene podatke od mikroupravljača i prikazuje ih korisniku. ESP32 djeluje kao web server, omogućavajući korisnicima pregled podataka putem web sučelja. Također, mikroupravljač može slati podatke na lokalni zaslon za brzi pregled.

Modul za kontrolu aktuatora

Modul za kontrolu aktuatora upravlja ventilatorom i prskalicama na temelju odluka koje donosi mikroupravljač.

Komponente:

- Relejni moduli za upravljanje strujom aktuatora
- Ventilatori
- Prskalice (LED)

Modul za kontrolu aktuatora prima signale od mikroupravljača i na temelju tih signala aktivira ili deaktivira ventilator i prskalice (LED). Ako koncentracija štetnih plinova premaši zadani prag, mikroupravljač šalje signal relejnom modulu koji aktiviraju ventilator ili prskalice.

Napajanje

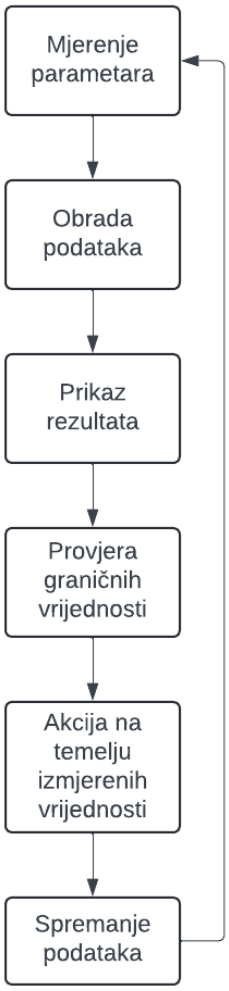
Sustav napajanja osigurava stabilan izvor napona za sve komponente sustava.

Komponente:

- Regulatori napona
- Napajanje iz mreže

Sustav napajanja osigurava potrebne napone za senzorski modul, mikroupravljač i modul za kontrolu aktuatora kako bi se osiguralo njihovo pravilno funkcioniranje

2.3. Prijedlog programskog riješenja



***Slika 2.3**. Osnovni blok dijagram algoritma*

Prvi blok se odnosi na mjerenje parametara sa senzora gdje se parametri odnosno vrijednosti senzora prikupljaju u intervalima. Zatim se podaci prikupljeni od senzora obrađuju i analiziraju kako bi se utvrdila trenutna kvaliteta zraka. Ovo može uključivati filtriranje podataka, uklanjanje šuma i izračunavanje prosječnih vrijednosti te prikaz podataka sa senzora na monitor. Nakon toga slijedi provjera jesu li izmjerene vrijednosti parametara zraka unutar prihvatljivih granica. Ako neki parametar prelazi graničnu vrijednost, prelazi se na akcijski blok. Ako su izmjerene vrijednosti izvan prihvatljivih granica, sustav može poduzeti određene akcije, poput aktiviranja ventilacije, alarma ili obavijesti korisnicima. Te na kraju spremanje samih podataka na server i ponovno čitanje podataka.

3. Realizacija sustava za mjerenje kvalitete zraka

3.1. Korištene komponente, alati i programska podrška

Sustav za mjerenje kvalitete zraka je uspješno izrađen koristeći sljedeće komponente.

- **ESP32-WROOM**
  - **Uloga i funkcija:** Glavni kontroler projekta. ESP32 WROOM je mikroupravljač s integriranom Wi-Fi i Bluetooth funkcionalnošću, što omogućuje prikupljanje podataka sa senzora i njihovu bežičnu komunikaciju.
  - **Najvažnije tehničke karakteristike:**
    - Dual-core procesor
    - Wi-Fi i Bluetooth konekcija
    - Niska potrošnja energije
    - Podrška za različite periferije (GPIO, I2C, SPI, ADC, DAC) [3]
- **DHT22**
  - **Uloga i funkcija:** Senzor za mjerenje temperature i vlažnosti zraka
  - **Najvažnije tehničke karakteristike:**
    - Temperaturni raspon: -40°C do 80°C, točnost ±0.5°C
    - Vlažnost zraka: 0% do 100%, točnost ±2-5%
    - Digitalni izlaz [4], [5]
- **MQ135**
  - **Uloga i funkcija:** Senzor za mjerenje kvalitete zraka, posebno koncentracije različitih plinova (NH_3, NO_x, alkohol, benzen, dim, CO_2).
  - **Najvažnije tehničke karakteristike:**
    - Analogni i digitalni izlaz
    - Brzo vrijeme odziva [6]
- **BMP180**
  - **Uloga i funkcija:** Senzor za mjerenje barometarskog tlaka i temperature.
  - **Najvažnije tehničke karakteristike:**
    - Raspon tlaka: 300-1100 hPa, točnost ±1 hPa
    - Temperaturni raspon: -40°C do 85°C
    - Digitalni izlaz preko I2C ili SPI [7]

- **Relej**
    - **Uloga i funkcija:** Elektronička komponenta koja omogućuje upravljanje visokim naponom (npr. ventilacijskim sustavom) pomoću signala niskog napona iz ESP32.
    - **Najvažnije tehničke karakteristike:**
        - Napon namota: 5V
        - Maksimalna struja: 10A
        - Tipično korištene za prekidanje visokih napona
- **LED**
    - **Uloga i funkcija:** Indikator statusa sustava. LED diode se koriste za vizualno obavještavanje korisnika o stanju sustava, kao što su mjerenja ili upozorenja.
- **Regulator napajanja**
    - **Uloga i funkcija:** Regulator napajanja za stabilizaciju napona, osigurava ispravno napajanje ventilatora.

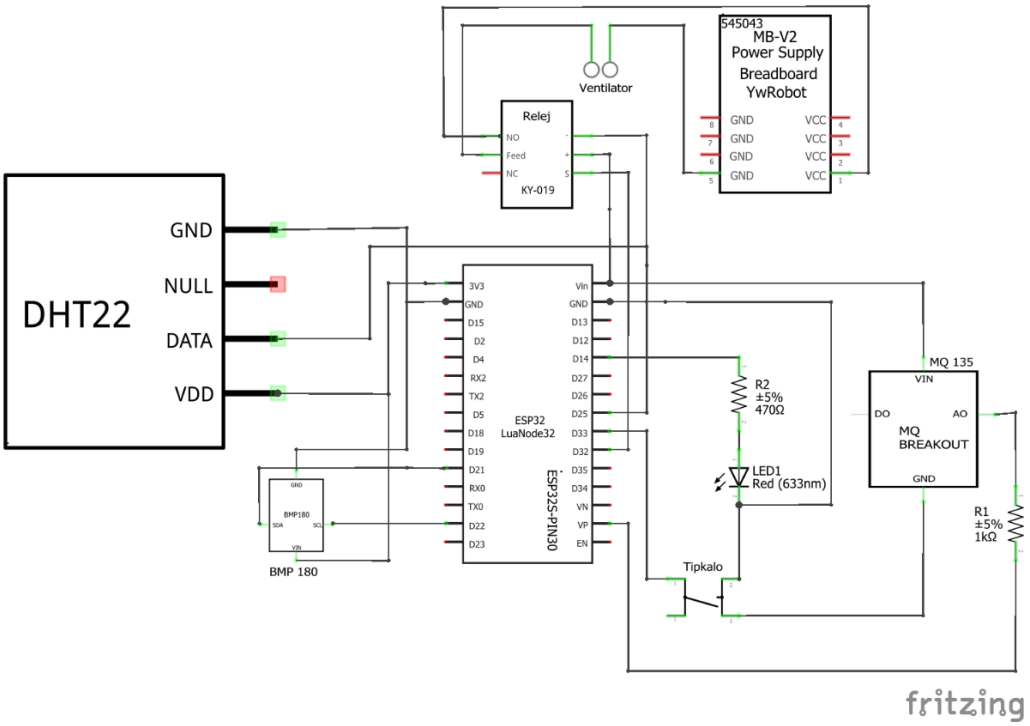
Svaka od ovih komponenti igra ključnu ulogu u funkcionalnosti projekta. Senzori (DHT22, MQ135, BMP180) prikupljaju podatke o okolini, ESP32 WROOM procesira te podatke i upravlja relejom, koji može aktivirati sustave za kontrolu okoline, dok LED pruža vizualne povratne informacije. Regulator napajanja HW-131 osigurava stabilno napajanje ventilatora.

Također je korišteno sljedeće programsko okruženje za izradu sustava je

- **Arduino IDE:**
    - Arduino IDE je softverska platforma koja omogućuje korisnicima jednostavno pisanje, kompiliranje i učitavanje koda na Arduino mikroupravljače. Dizajnirana je kako bi bila pristupačna za početnike, a istovremeno dovoljno moćna za iskusne korisnike. Pruža jednostavno korisničko sučelje, podršku na više platformi, velik broj ugrađenih biblioteka za podršku različitim senzorima te serijski monitor za komunikaciju mikroupravljača i računala putem serijskog porta i za ispisivanje poruka, dijagnosticiranje problema i testiranje koda. [8]

3.2. Realizacija konstrukcijskog i sklopovskog rješenja

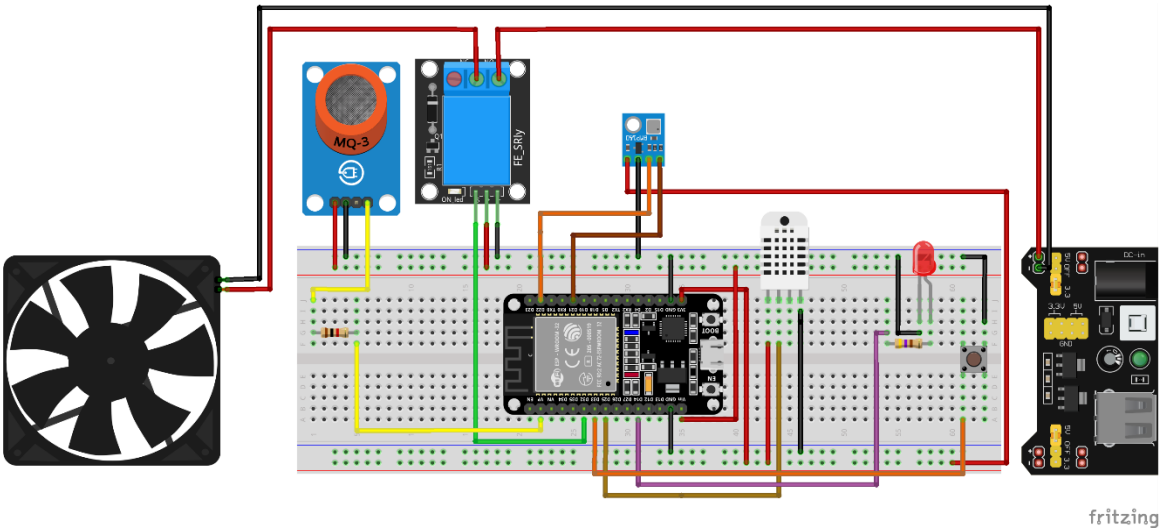
Na sljedećoj slici je prikazana sama električna shema sustava za mjerenje kvalitete zraka.



***Slika 3.1**. Električna shema*

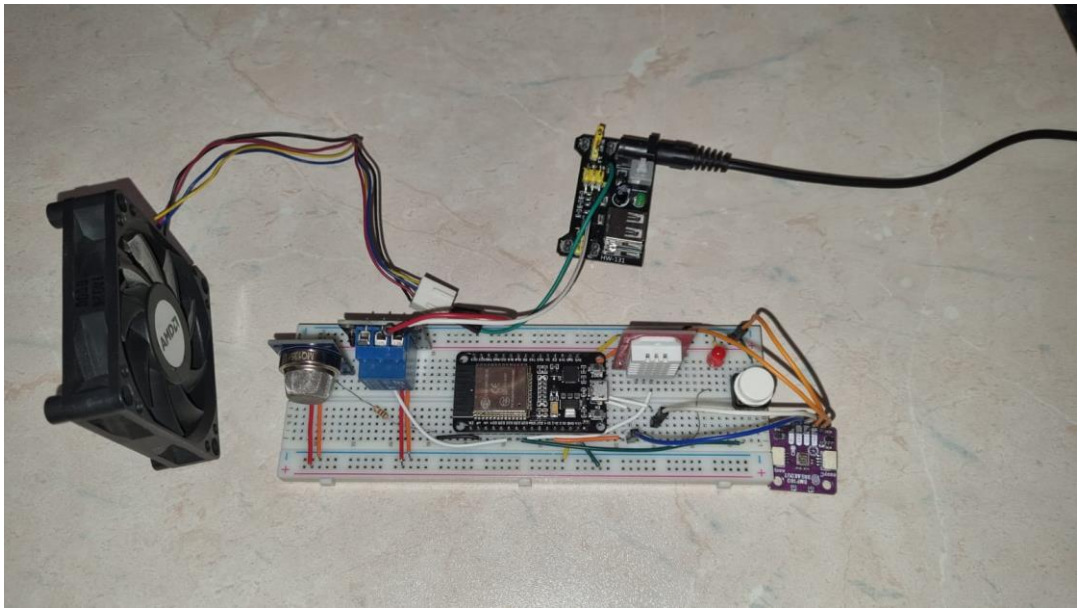
Na slici je vidljivo da je senzor za temperaturu i vlažnost zraka DHT22 spojen preko DATA priključka na ulaz mikroupravljača na priključak D25 te je bilo potrebno spojiti napon i uzemljenje na sam senzor kako bi mogao ispravno raditi. Zatim imamo senzor za mjerenje tlaka BMP180 na kojem je također spojen napon i uzemljenje iz pomenutog razloga te je taj senzor bilo potrebno spojiti pomoću 2 priključka na mikroupravljač. BMP180 radi na osnovi I2C komunikacije te je bilo potrebno spojiti SCL i SDA priključke samog senzora na SCL (D22) i SDA (D21) priključke mikroupravljača kao što je prikazano na gornjoj shemi. Posljednji senzor koji je spojen je senzor za kvalitetu zraka MQ135 na kojeg je doveden napon i uzemljenje te analogni izlaz senzora na ulaz GPIO36 (na shemi označen s EN) mikroupravljača preko otpornika od 1kΩ. Spojena je i LED koja signalizira sustav za gašenje požara preko otpornika od 470Ω, tipkalo koje služi za ručno upravljanje samim sustavom koji je spojen na ulaz D33. Te na kraju je bilo potrebno spojiti sam aktuator sustava odnosno ventilator koji pročišćava zrak, ventilator je spojen preko releja koji je spojen na ulaz D32 mikroupravljača, na relej je također doveden napon i uzemljenje, ventilator je spojen preko releja tako da se relej ponaša kao sklopka te kada dođe signal s priključka D32 strujni krug bude zatvoren, odnosno strujni krug s modula za dovođenje 12V koji su potrebni ventilatoru. Sam modul za dovođenje napona je spojen preko adaptera na izmjeničnu mrežu.

Na sljedećoj slici je prikazana *breadboard* (hrv. ploča za povezivanje) shema u *Fritzing-u*.



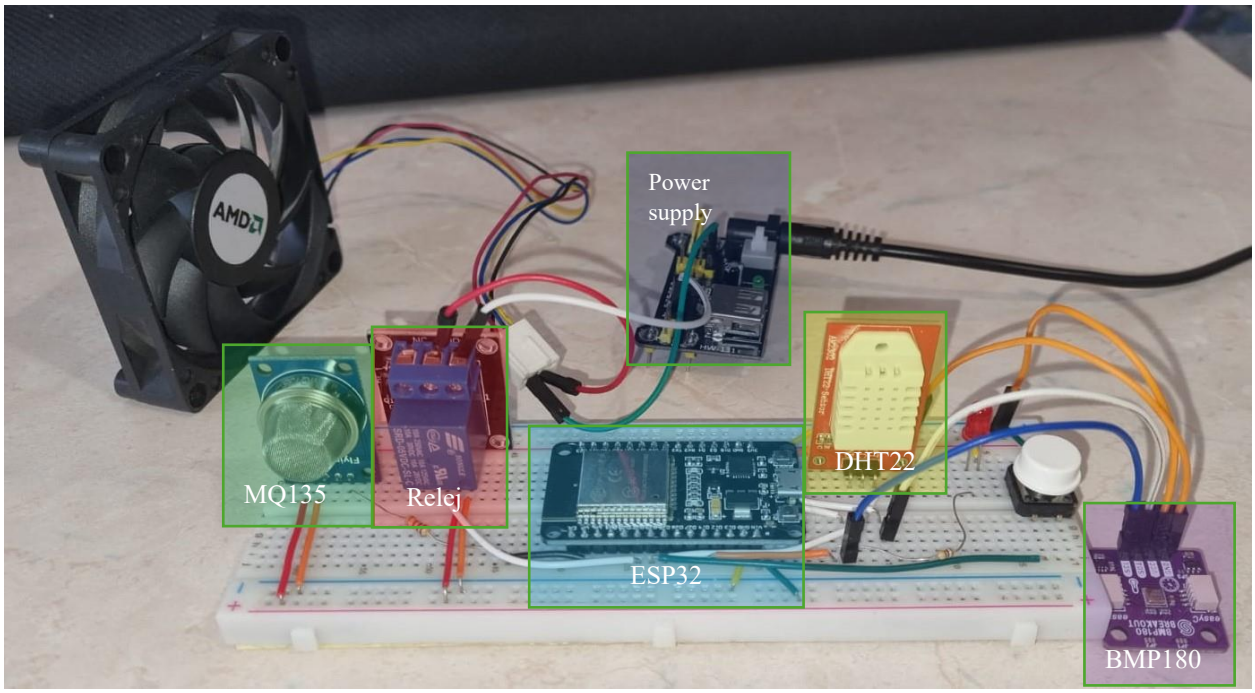
Slika 3.2. *Breadboard shema*

Ovim prikazom su korištene komponente jednostavnije i jasnije vidljive te način na koji su one u stvarnosti povezane na isti *breadboard* što je vidljivo i na slici ispod.



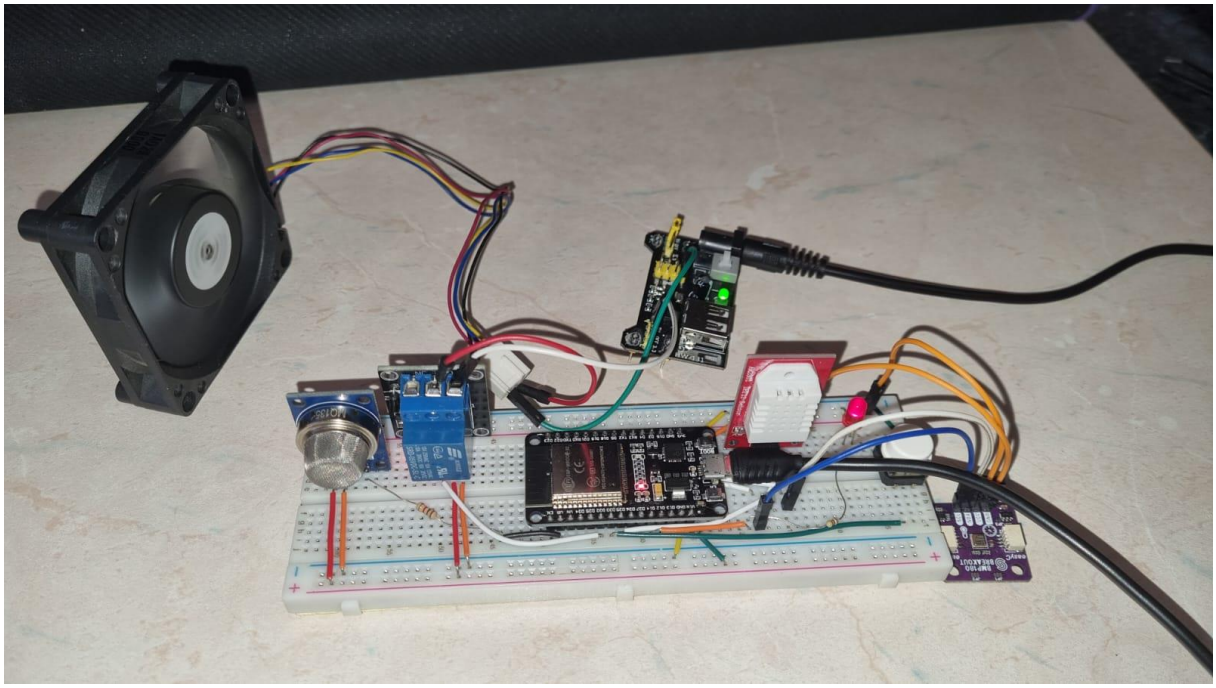
Slika 3.3*. Gotov hardverski projekt*

Na sljedećoj slici vidimo označene komponente odnosno senzore, mikroupravljač i modul za napajanje radi lakšeg raspoznavanja istih.



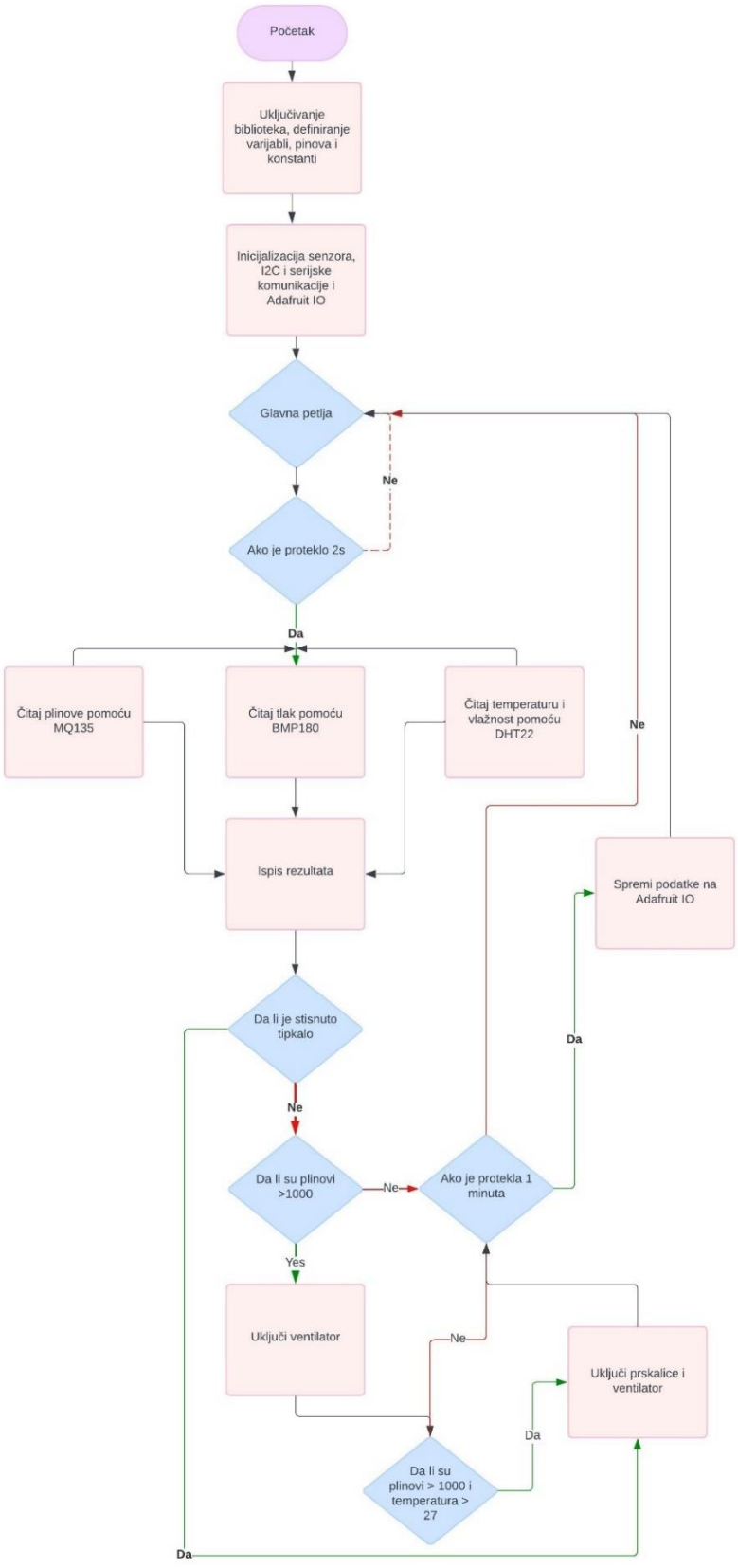
***Slika 3.4**. Označene komponente sustava*

Na slici u nastavku vidljiv je i sam sustav kada je priključen napon odnosno dok je sustav uključen i u funkciji, vidljivo je kako se ventilator vrti zbog određenog iznosa kvalitete zraka i temperature, pored ventilatora također je uključena i LED-ica koja signalizira vodene prskalice za gašenje požara.



***Slika 3.5**. Sustav u pogonu*

3.3. Realizacija programskog rješenja



Slika 3.6. *Detaljan blok dijagram*

Na samom početku programskog riješenja odrađeno je uključivanje biblioteka koje su bile neophodne za dobivanje podataka sa senzora i za Adafruit IO web server. Definirani su i podaci za web server kao što su podaci o mreži na kojoj je povezan sam ESP32. Definirani su pinovi na koje će biti povezani senzori s ESP32 te pomoćne varijable i konstante koje su potrebne za rad sa senzorima.

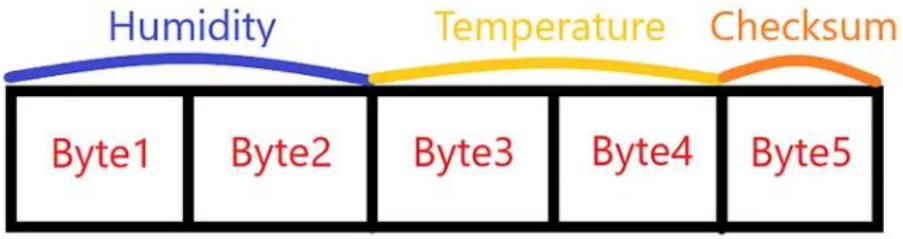
Nakon toga slijedi sama inicijalizacija gdje se postavljaju pinovi za senzore, pokretanje serijske i I2C komunikacije za rad sa senzorom te za ispis rezultata na monitor, spajanje na Adafruit IO koji služi za pohranu podataka te samo inicijaliziranje korištenih senzora. Uz inicijaliziranje bilo je potrebno i napraviti provjere kako bi osigurali da je uspješno realizirana serijska komunikacija, da je realizirana veza s Adafruit IO web serverom, te provjera inicijalizacije senzora. U slučaju neuspješne inicijalizacije, korisnik će biti obaviješten prikladnom porukom.

Zatim slijedi glavna petlja koja čita podatke sa senzora i provjerava granične vrijednosti. Na početku glavne petlje se pokreće Adafruit IO komunikacija te nakon toga slijedi provjera vremenskih intervala odnosno, postavljena su 2 timer-a koji prate koliko je vremena prošlo od ispisivanja rezultata i slanja rezultata na web server. To je bilo potrebno napraviti iz razloga što bi se rezultat trebao prikazati odmah na monitoru, a rezultati koji se šalju na server se moraju ograničiti zbog prevelike količine podataka koji se šalju. Ako je zadovoljen prvi uvjet odnosno ako su prošle 2 sekunde od ponovnog pokretanja petlje ili mikroupravljača onda se čitaju podaci sa senzora, izračunava se temperatura, vlažnost zraka, ppm (*parts per million*), tlak i nadmorska visina te se ispisuju isti podaci na serijski monitor. Ako su podaci neuspješno pročitani, odnosno ako je došla nekakva smetnja kao npr. veza između senzora i mikroupravljača se prekinula onda će se ispisati odgovarajuća poruka. Zatim se provjeravaju granične vrijednosti tj. ako je pritisnuto tipkalo kojim manuelno upravljamo sustavom uključuje se ventilator i prskalice (LED), ako ne onda se provjerava granična vrijednost za plinove te ako je veća od 1000 ppm uključuje se ventilator, zatim se provjerava temperatura te ako je i ona veća od 27 °C upalit će se i prskalice za vodu (LED). Odnosno ako je ispunjen zadnji uvjet uključit će se i prskalice i ventilator.

Na kraju slijedi slanje podataka na web server, tj. ako je prošla jedna minuta onda se podaci šalju na Adafruit web server, a ako nije ide se na početak petlje koja se izvodi beskonačno ili do isključenja samog mikroupravljača. U nastavku slijedi objašnjenje funkcija koje su realizirane kako bi dobili podatke sa senzora. Nisu korištene nikakve biblioteke namijenje isključivo za rad s pojedinim senzorom. Umjesto toga funkcije su realiziranje uz pomoć shvaćanja kako koji senzor radi te uz pomoć jednostavnih biblioteka kao npr. „Wire.h“.

3.3.1. Čitanje podataka sa senzora DHT22

Za realizaciju funkcije pomoću koje se dobivaju podatci s DHT22 prvo je potrebno razumjeti kako senzor radi. DHT22 je digitalni senzor za mjerenje temperature i vlage koji koristi kapacitivni senzor za mjerenje vlažnosti zraka i termistor za mjerenje temperature. Unutar senzora nalazi se mikroupravljač koji uvijek sluša dani priključak senzora. Kada je taj priključaj pokrenut (promjena stanja na priključku) od strane drugog mikroupravljača, onda mikroupravljač senzora krene slati podatke o vlažnosti i temperaturi u digitalnom obliku. Prvo je potrebno na početku „probuditi" senzor s logičkom 0 u trajanju od 18ms kako bi senzor znao da se započinje komunikacija, zatim se postavlja pin (hrv. priključak) na *HIGH* (hrv. logička 1) u trajanju od 20-40 µs. Zatim se čita odgovor senzora tako da se pin postavi na ulaz i pričeka se da senzor pošalje *LOW* (hrv. logička 0) signal (80 µs), zatim *HIGH* signal (80 µs). Senzor šalje 40-bitni podatak (5 bajtova) u sljedećem formatu: 8 bitova vlažnosti, 8 bitova vlažnosti decimalno, 8 bitova temperature, 8 bitova temperature decimalno, i 8 bitova *checksum*. Potrebno je čitati svaki bit tako da se mjeri trajanje visokog signala: 26-28 µs predstavlja logičku 0, dok 70 µs predstavlja logičku 1. Na kraju slijedi provjera ispravnosti podataka tako da se izračuna *checksum* zbrajanjem prva četiri bajta i taj rezultat mora biti jednak petom bajtu. To je postignuto funkcijom *readDHT22(data)* koja prima prazno polje od 5 elemenata po 8 bajtova. Te se zatim izvodi kod koji je opisan i funkcija šalje *true* ili *false* vrijednosti. Poslane vrijednosti nam govore da li je podatak dobro pročitan ili ne (*checksum*). [5]



Slika 3.7. *[5] Polje s podacima o temperaturi i vlažnosti zraka*

3.3.2. Čitanje podataka sa senzora MQ135

MQ135 je senzor za detekciju raznih plinova (amonijak, alkohol, benzen, dim, CO_2). Senzor koristi metal-oksidni poluvodič za detekciju prisutnosti plinova. Otpor senzora mijenja se u prisutnosti različitih koncentracija plinova. Kako bi dobili podatke sa senzora potrebno je mjeriti otpor tako što se mjeri napon na analognom ulazu na koji je spojen sam senzor s mikroupravljačem. Zatim uz pomoć formule računa napon i koncentraciju plina. Detaljnije je objašnjeno u nastavku.

Prvo je potrebno izračunati korekcijski faktor na temelju temperature i vlažnosti. Korekcijski faktor se razlikuje za temperature ispod i iznad 20°C. Korekcijski faktor računa se pomoću funkcije *getCorrectionFactor(t,h)* koja prima temperaturu i vlažnost zraka, a izlaz funkcije je vrijednost koja kasnije služi za dobivanje ispravljene vrijednosti otpora senzora. Zatim se samo ispituje da li je temperatura niža od 20, u ovisnosti o tome korištena je odgovarajuća formula. Formule su vidljive kasnije u dokumentaciji u kodu, te za izračun korekcijskog faktora korištene su globalne varijable (COREA, COREB, COREC, CORED). Parametri CORA, CORB, CORC, CORD, CORE, CORF i CORG koji se nalaze u kodu služe za modeliranje ovisnosti MQ135 senzora o temperaturi i relativnoj vlažnosti. Ovi parametri nisu jedinstveni za MQ135 senzor, već su empirijski dobiveni i odnose se na specifičnu funkciju koja pokušava kvantificirati kako se otpornost senzora mijenja pri različitim temperaturama i vlažnostima. Formula za *getCorrectionFactor(t, h)* je izvedena na osnovu eksperimentalnih podataka i analize. Ovi parametri se koriste kako bi se prilagodila otpornost senzora na različite uslove okoline, što omogućava preciznije mjerenje koncentracije CO_2.

Nakon toga u funkciji *getResistance(pin)* računa se otpor senzora na temelju analognog očitanja s odgovarajućeg pina. Vrijednost s analognom pina se čita, zatim se skalira da se dobije vrijednosti u rasponu od 0 do 1024, skalira se s vrijednosti 4095 jer je ADC kod ESP32 12-bitni. Na kraju se dobivena vrijednost množi s unutarnjim otporom čija je pretpostavljena vrijednost 10kΩ.

Funkcija *getCorrectedResistance(resistance,t,h)* prima parametre otpor (već izračunati otpor senzora), temperaturu i vlažnost koji služe za dobivanje korekcijskog faktora. Funckija vraća ispravljenu (korigiranu) vrijednost otpora senzora koji ovisi o temperaturi i vlažnosti zraka (rezultati mjerenja na taj način su precizniji nego kada bi se čista vrijednost otpornika pretvarala s analognog pina senzora u koncentraciju plinova).

Na kraju je izračunata koncentracija plinova, funkcija *getPPM (resistance, t, h)* prima otpor senzora, temperaturu i vlažnost zraka, te se pomoću tih parametara računa popravljena vrijednost otpora senzora, te se on uvrštava u formulu s dodatnim parametrima kako bi dobili konačnu koncentraciju plinova. Korišteni parametri u formuli su RZERO koji predstavlja kalibracijski otpor pri atmosferskom CO_2 nivou, PARA i PARB su kao već pomenuti parametri empirijski dobiveni, odnosno eksperimentalno. Često se ovi parametri određuju kroz eksperimente gdje se senzor izlaže različitim poznatim koncentracijama CO_2 i mjere se njegove otpornosti. Zatim se na osnovu tih

mjerenja može napraviti model koji povezuje otpornost senzora sa koncentracijom CO_2, koristeći parametre poput PARA i PARB. [2], [6], [9]

3.3.3. Čitanje podataka sa senzora BMP180

BMP180 je senzor za mjerenje atmosferskog tlaka i temperature (temperatura utječe na vrijednost tlaka te se zbog toga mjeri kako bi se dobili točni rezultati). Senzor koristi piezoelektrični efekt za mjerenje tlaka i termistor za mjerenje temperature. Podaci se prenose preko I2C komunikacije. Dobiveni analogni signal pretvara se u digitalni pomoću ADC-a. BMP180 ima integriranu EEPROM memoriju s podacima za kalibraciju. Senzor prvo mjeri sirove vrijednosti temperature i tlaka, koje se zatim kalibriraju koristeći pohranjene kalibracijske podatke. Svaki senzor ima posebni, specifični 176 bitni niz podataka koji se koristi za otklanjanje šumova, kompenzaciju pomaka i određivanje ovisnosti o temperaturi. To su zapravo zapisi o omjerima stvarnih i očitanih mjerenja, a upravo oni omogućavaju precizne izračune konačnih vrijednosti tlaka i temperature.

Prvo se zadaje način rada senzora te se zatim čitaju kalibracijski podaci iz EEPROM memorije koje očitavamo preko I2C-a, to radi funkcija *begin(mode)* koja prima način rada senzora (govori o tome koliki će *delay* (hrv. odgoda) biti između čitanja podataka te tako rezultati ovise o brzini čitanja podataka) te vraća *true* ako su podaci uspješno pročitani.

Zatim se računa atmosferski tlak pomoću *readPressure(void)* funkcije. Nakon ADC konverzije dobivaju se nekompenzirane vrijednosti: UP – *uncompensated preassure* (hrv. nekompenzirani tlak) i UT – *uncompensated temperature* (hrv. nekompenzirana temperatura). Te vrijednosti su isčitane pomoću funkcije *readRawTemperature(void)* koja čita sirovu vrijednost temperature sa senzora. Funkcija upisuje naredbu za čitanje temperature (BMP085_READTEMPCMD) u kontrolni registar te vraća 16 – bitnu vrijednost iz temperaturnog registra BMP085_TEMPDATA. Funkcija *readRawPressure(void)* čita sirovu vrijednost tlaka. Piše naredbu za čitanje tlaka (BMP085_READPRESSURECMD) u kontrolni registar te čita 16-bitnu vrijednost tlaka i dodatnih 8 bitova, kombinira ih i vraća rezultat. Te funkcije rade pomoću pomoćnih funkcija koje koriste I2C komunikaciju za čitanje i pisanje podataka:

- **read8**: Čita 8-bitni registar.
- **read16**: Čita 16-bitni registar.
- **write8**: Piše 8-bitnu vrijednost u registar.

Nakon očitavanja sirove temperature i atmosferskog tlaka potrebno je izračunati B5 vrijednost koja je pomoćna varijabla koja se koristi za daljne izračune temperature i tlaka. Koriste se kalibracijski podaci i složene formule za izračun konačnog tlaka.

Funkcija *readAltitude(sealevelPressure)* je funkcija koja računa nadmorsku visinu pomoću atmosferskog tlaka i koristi poznatu formulu za izračun nadmorske visine iz tlaka i tlaka na nivou mora. [7], [10]

4. Testiranje i rezultati

4.1. Metodologija testiranja

U ovom poglavlju opisana je metodologija koja je korištena prilikom testiranja sustava za mjerenje kvalitete zraka. Testiranje obuhvaća provjeru ispravnosti svih dijelova sustava, uključujući senzore, mikroupravljač i programski kod. Testiranje je provedeno u različitim uvjetima kako bi se osigurala vjerodostojnost rezultata. Prvo je testirana ispravnost mikroupravljača, zatim ispravnost programskog koda i senzora, odnosno provjereno je hoće li sustav ispravno (onako kako je zadano) raditi promjenom vanjskih parametara.

Provedba ispravnosti mikroupravljača

Prvo je potrebno testirati ispravnost glavnog dijela sustava i komunikaciju s Adafruit IO web serverom. Testiranje mikroupravljača provodeno je tako da je testirana ispravnost prikupljanja podataka sa senzora, odnosno ispitano je prikuplja li mikroupravljač ispravno podatke sa samih senzora i pohranjuje li ih u odgovarajući format. To se može ostvariti tako da se uključi čitav sustav i provjeravaju vrijednosti korištenih senzora da se osigura da su vrijednosti svakog u uvjetima normale, tako što se čita *serial monitor* i prati vrijednosti senzora te manualno upravlja sustavom za ventilaciju i gašenje požara. Zatim se ispita da li se podaci uspješno šalju na web server tako što se uđe u *dashboard* (hrv. nadzorna ploča) web servera te se uspoređuju vrijednosti koje su ispisane na *serial monitor* s vrijednostima koje su prikazane na web serveru.

Testiranje sustava (programskog koda i senzora)

Testiranje istog podijeljeno je na tri slučaja: promjena temperature, vlažnosti i koncentracije plinova u prostoriji što bi trebalo rezultirati aktiviranjem kreiranog sustava za provjetravanje (automatsko uključivanje ventilatora) i gašenje požara (automatsko uključivanje LED). Također manualno upravljanje automatskim sustavom.

Promjena temperature i vlažnosti: Senzori su testirani u kontroliranom okruženju (prostorija od 6m^2) gdje su se temperatura i vlaga mijenjali uz pomoć klima uređaja. Očitavanja senzora se uspoređivalo s referentnim uređajem za mjerenje temperature (termometar) kako bi se provjerila točnost senzora, dobivene vrijednosti vlage nije bilo moguće usporediti zbog nedostatka referentnog uređaja za mjerenje iste. Mjerenje je provedeno na način da su prvo izmjerene temperatura i vlažnost zraka s uključenom klimom, jednom osobom u prostoriji i zatvorenim prozorom te zatim bez uključene klime i otvorenim prozorom u prostoriji. Također istovremeno se mjerila i koncentracija plinova u prostoriji za oba slučaja, u drugom slučaju koncentracija plinova

se ne bi trebala mjenjati zato što je prozor otvoren postoji protok zraka. Bitno je napomenuti da se klima uređaj ne nalazi direktno u prostoriji nego u susjednoj prostoriji. Pod pojmom klimatizirane prostorije smatran je slučaj kada su vrata od prostorije u kojoj se nalazi sustav za mjerenje otvorena.

Promjena koncentracije plinova (CO_2, butan, alkohol): Senzori su izloženi različitim koncentracijama različitih plinova u sigurnim uvjetima. Koncentracije plinova nije bilo moguće mjeriti pomoću referentnog uređaja zbog nedostatka mjernog uređaja. Mjerenje je provedeno na način da je senzor izlagan različitim plinovima. Najprije je potrebno uključiti senzor na 15 minuta kako bi se zagrijao odnosno kalibrirao kako bi dao ispravne rezultate. Kalibracija senzora obavljena je u klimatiziranim sobnim uvjetima s jednom osobom u prostoriji i bez izlaganja plinovima. Zatim je senzor izložen alkoholu kako bi se dobio očekivani porast vrijednosti senzora, te butanu uz pomoć upaljača. Samo kalibriranje senzora je realizirano prije bilo kakvih provedenih mjerenja. Na kraju je testiran automatski sustav tako što je izložen butanu koji aktivira ventilator i uspoređeno je koliko vremena je potrebno da se kvaliteta zraka stabilizira (zrak pročisti) u odnosu na stabiliziranje zraka bez ventilatora.

Mjerenje amtosferskog tlaka: U ovom slučaju nije bilo moguće mjenjati vanjske parametre. Atfmosferski tlak je većinom isti te se ne mjenja tako brzo u kratkom vremenskom periodu, isto to vrijedi i za nadmorsku visinu koja bi trebala biti konstantna s obzirom da je testiranje provedeno na istoj lokaciji. Bitno je napomenuti da se atmosferski tlak mjerio za sve slučajeve promjena parametara. Usporedba dobivenih rezultata sa stvarnim obavljena je tako što su izmjereni rezultati uspoređeni s rezultatima koje dostupnim na metorološkoj postaji Osijek – Čepin.

Intepretacija rezultata

Rezultati testiranja interpretirani su kroz analizu prikupljenih podataka. Ako su odstupanja unutar prihvatljivih granica, sustav se smatra vjerodostojnim. U slučaju značajnih odstupanja, analizirat će se uzroci i predložiti poboljšanja. Rezultati će biti ispisani na *serial monitor* (na Arduino razvojnom okruženju) svake 2s gdje se može odmah vidjeti rezultate mjerenja u stvarnom vremenu, također će rezultati biti dostupni na web serveru koji će se osvježavati svake minute sa stvarnim podacima iščitanih sa senzora.

4.2. Rezultati testiranja

Testiranje mikroupravljača i Adafruit web servera

Kao što je bilo spomenuto prvo je provedeno testiranje mikroupravljača koje je prošlo uspješno. Mikroupravljač se uspješno povezao s Adafruit IO web serverom te su se svi senzori uspješno inicijalizirali, dali smislene rezultate te su uspješno rezultati (podaci) poslani na web server.

```
Temperatura: 25.40 *C
Vlaznost: 58.70 %
PPM: 183.03 ppm
Pritisak = 100339 Pa
Visina = 96.19 metara

Prskalice su uključene.
```

Slika 4.1. *Serial monitor s rezultatima*

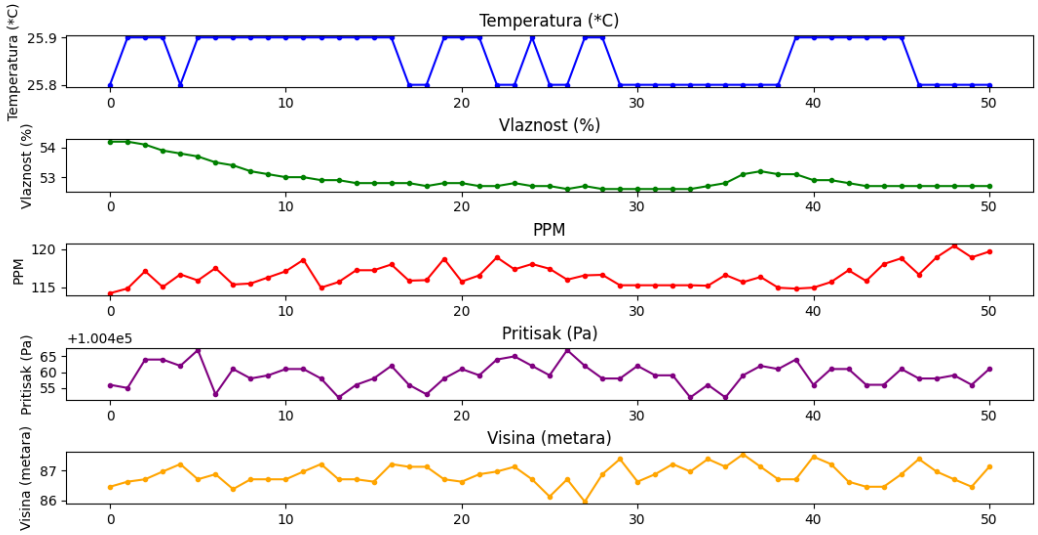
Vidljivi su jasne vrijednosti rezultata sa senzora i vidljivo je kako se na *serial monitor-u* s Arduino IDE-a prikazuje stanje prskalica odnosno LED. Navedeno je prikazano zato što je provedeno manualno upravljanje prskalicama i ventilatorom pomoću tipkala nakon čega je dobivena odgovarajuća poruka.

Slika 4.2. *Dashboard werb servera - prikaz rezultata s web servera*

Vidljivo je da postoje nagli porasti podataka na web serveru, razlog toga je što su rezultati bili prikazivani u većem vremenskom periodu te zbog uključivanja i isključivana mikroupravljača, odnosno rezultati senzora koji se dobiju odmah nakon uključivanja sustava nisu nužno točni pa je potrebno sačekati 10 minuta kako bi se rezultati stabilizirali i bili smisleni. Zbog toga su vidljivi nagli porasti u vrijednostima.

Rezultati promjene temperature i vlažnosti zraka

Testiranje je obavljeno kako je ranije navedeno, prvo je testiran sustav u klimatiziranim uvjetima, svake 2 sekunde su se spremali podaci sa senzora 51 puta (51 uzorkovanja), izračunate su srednje vrijednosti senzora, te grafički prikazani podataci sa senzora u ovisnosti o vremenu. Grafički prikaz podataka sa senzora je vidljiv na slici ispod.



Slika 4.3. *Grafički prikaz podataka sa senzora u klimatiziranoj prostoriji*

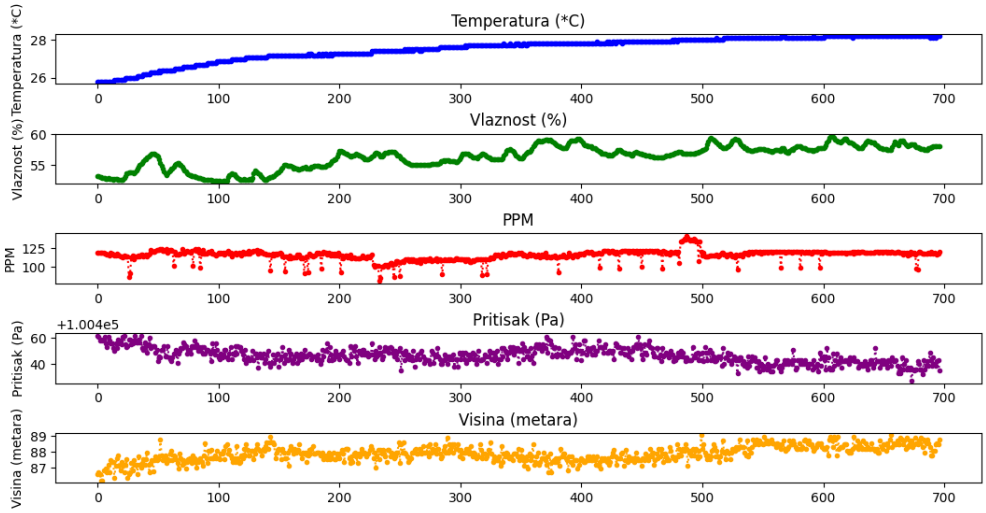
Vidljivo je kako se temperatura vremenom jako malo mijenja odnosno za 0.1 ºC što je u skladu s normalom. Termometar za dobivanje referentne vrijednosti je pokazivao istu vrijednost kao i senzor dok je termostat bio postavljen odmah pored samog senzora kako bi dobili što bolju usporedbu rezultata. Vlažnost, kako je prethodno rečeno, nije bilo moguće usporediti s nekom refetentnom vrijednosti, ali prema specifikacijama senzora zna se da je odstupanje rezultata senzora od stvarnih vrijednost vlage ali i temperature zraka jako malo. Vidljivo je kako se vlažnost jako malo mjenjala (1%), to je očekivano s obzirom da je senzor bio u blizini osobe pa je pojava manjih oscilacija normalna. Vlažnost je opala s 54% na 53% nakon čega se i stabilizirala, opadala je zato što je klima uređaj uključen nedugo prije početka mjerenja pa su temperatura i vlažnost zraka bili veći nego dok je uređaj bio uključen. Koncentracija plinova u prostoriji je mirovala,

vidljive su samo male oscilacije koje mogu nastati zbog puhanja zraka iz usta osobe u senzor, zbog elektronskog šuma, te zbog same promjene koncentracije čestica u prostoriji. Atmosferski tlak je također jako malo varirao što je moguće zbog elektronskog šuma ili malih promjena u atmosferi Rezultati atmosferskog tlaka se razlikuju od stvarnog atmosferskog tlaka u Osijeku za ±1hPa, takvo odstupanje je moguće iz više razloga, a neki od njih su: inteferencija signala, kvaliteta zraka (temperatura i vlažnost zraka), kalibracija, nelinearnost. Kako je nadmorska visina usko povezana s atmosferskim tlakom tako je i ona jako malo varirala. Izmjerena vrijednost nadmorske visine je točna. Izračunate su i srednje vrijednosti mjerenja kako bi lakše usporedili rezultate sva tri mjerenja.

- Srednja vrijednost temperature: 25.85 °C
- Srednja vrijednost vlažnosti: 52.96 %
- Srednja vrijednost PPM: 116.59
- Srednja vrijednost tlaka: 100459.18 Pa
- Srednja vrijednost visine: 86.86 metara

Nakon što je isključen klima uređaj, čekalo se 15 minuta kako bi zagrijali sobu, te se testirao sustav u zatvorenoj prostoriji, s jednom osobom dok je prozor bio otvoren kako bi se dobila veća temperatura i vlažnost zraka u prostoriji.

Vrijednosti senzora testirana 697 puta kako bi se dobili bolji rezultate na kojima je moguće uočiti promjene rezultata mjerenja. Grafički prikaz rezultata je prikazan slikom ispod.



Slika 4.4. *Grafički prikaz podataka sa senzora u neklimatiziranoj prostoriji*

Ponovno su izračunate srednje vrijednosti radi usporedbe podataka.

- Srednja vrijednost temperature: 27.54 °C
- Srednja vrijednost vlažnosti: 56.34 %
- Srednja vrijednost PPM: 116.02
- Srednja vrijednost pritiska: 100446.57 Pa
- Srednja vrijednost visine: 87.91 metara

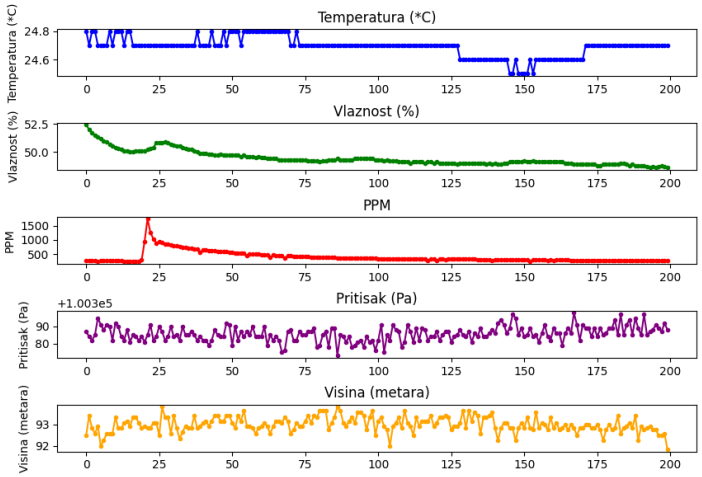
Iz rezultata je vidljivo da je temperatura porasla za 2.15 °C. Temperatura je nastavila rasti i nakon prestanka uzimanja mjerenja sustava. Rezultati temperature u ovom mjerenju nemaju odstupanje s rezultatima mjerenja s termometrom kao i u prvom slučaju. Na osnovu toga je zaključeno da senzor DHT22 daje točne rezultate. Vlažnost zraka je također porasla za 3% što se očekivalo jer je klima uređaj isključen i prestao se hladiti zrak. Koncentracije plinova su ponovno oscilirale iz pomenutih razloga. Kako je i predviđano, koncentracija plinova se smanjila zbog provjetravanja prostorije, a atmosferski tlak se blago smanjio zbog promjene temperature, sukladno s tim i sama vrijednost nadmorske visine se promjenila.

Kada se promatra grafički prikaz podataka u prostoriji koja nije klimatizirana vidljivo je da je temperatura konstantno rasla što je očekivano, vlažnost zraka se također povećavala postupno s manjim oscilacijama, te su atmosferski tlak, nadmorska visina i koncentracije plinova također neznačajno oscilirale. Oscilacije su moguće zbog već pomenutog elektronskog šuma, nedovoljno dobrog kalibriranja samih senzora, promjene stvarnih parametara zraka te najvjerovatniji razlog je period uzorkovanja podataka sa senzora, odnosno kada bi mjerenje provodili svake minute imali bi manje oscilacije tj. rezultati sa senzora bi bili konzistentniji, s tim da su oscilacije koje se javljaju male i neznačajne. Iz tog razloga nije bilo potrebe za promjenom rada sustava.

Rezultati promjene koncentracije plinova u prostoriji

Testiranje sustava na izlaganje alkoholu

Najprije je potrebno kalibrirati senzor za mjerenje koncentracije plinova u čistom zraku. Nakon toga je senzor izložen alkoholu (40% alkohol) oko 3 sekunde i odmah je uočen porast koncentracije plinova u prostoriji. Dobiveni rezultat prikazan je slikom u nastavku.



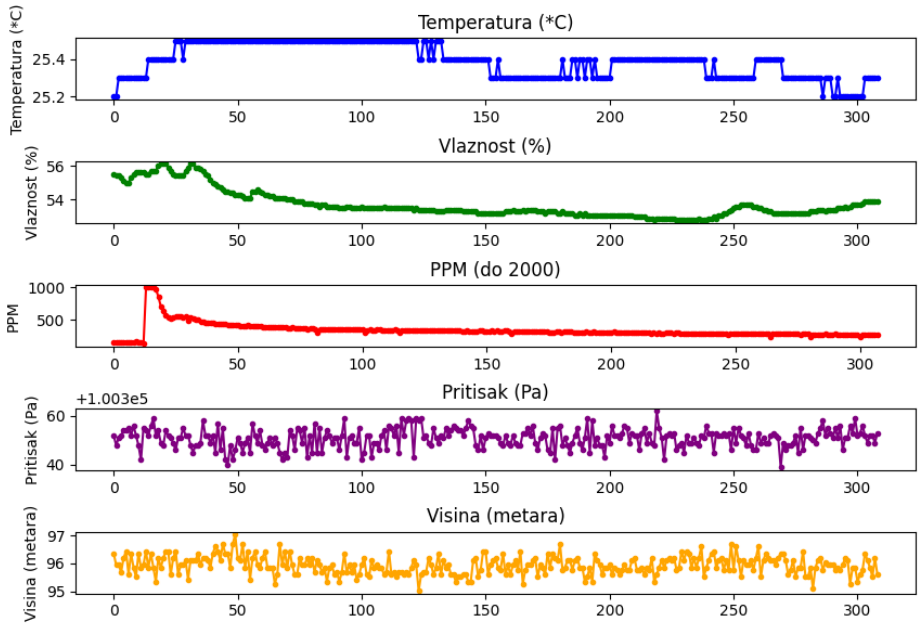
Slika 4.5. *Grafički prikaz vrijednosti senzora na utjecaj alkohola*

Koncentracija plinova je s 250 ppm naglo skočila preko 1500 ppm zbog utjecaja alkohola u zraku. Vidljivo je kako je taj nagli porast trajao otprilike onoliko vremena kao i samo izlaganje senzora alkoholu. Nakon prestanka izlaganja alkoholu, vrijednost dobivena sa senzora se smanjivala jer je zrak postajao sve čistiji. Zaključeno je da senzor radi ispravno. Vidljivo je i smanjivanje temperature i vlažnosti u prostoriji jer je mjerenje provedeno u klimatiziranim uvjetima kako bi zrak bio čistiji i kako bi se dobio bolji prikaz utjecaja alkohola na senzor. Također je bitno napomenuti kako je ispitivanje provedeno u 200 uzoraka jer se čekalo ponovno stabiliziranje koncentracije plinova na početno stanje. Sustav za provjetravanje je uklonjen iz čitavog sustava kako bi se vidio porast koncentracije plinova i prirodno stabiliziranje (smanjenje) koncentracije plinova u zraku. Važno je napomenuti kako se atmosferski tlak i nadmorska visina razlikuju od onih kada je sustav testiran na promjenu temperature i vlažnosti. Razlikuju se zato što samo testiranje nije bilo obavljeno isti dan te se i sam atmosferski tlak mogao promijeniti, također i temperatura se razlikovala te s tim i rezultati mjerenja sa senzora za mjerenje atmosferskog tlaka i nadmorske visine. Srednje vrijednosti mjerenja sustava prilikom izlaganja alkoholu:

- Srednja vrijednost temperature: 24.69 °C
- Srednja vrijednost vlažnosti: 49.44 %
- Srednja vrijednost PPM: 397.23
- Srednja vrijednost pritiska: 100385.69 Pa
- Srednja vrijednost visine: 93.01 metara

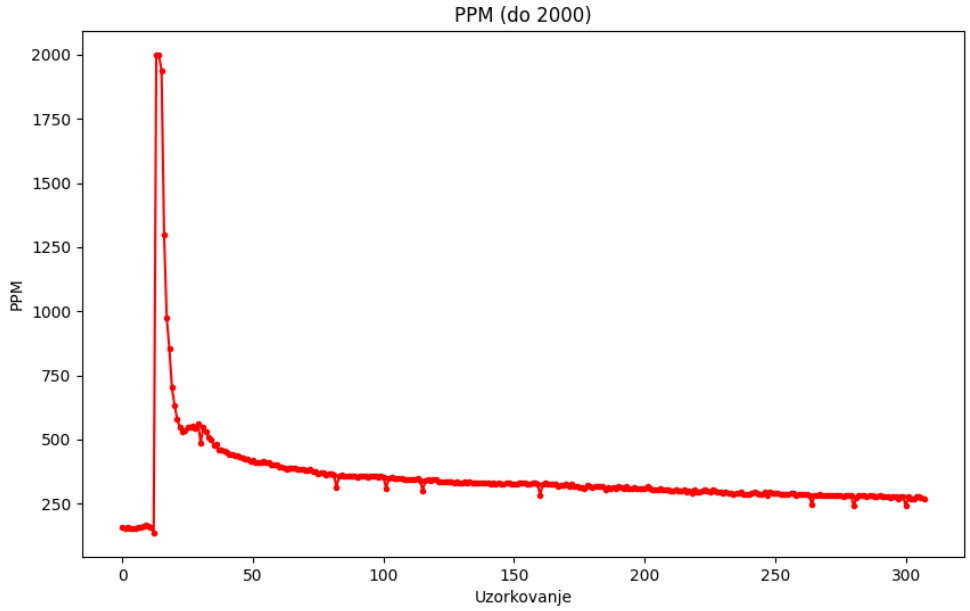
Testiranje sustava na izlaganje butanu bez ventilatora

Nakon toga je provedena ponovna kalibracija sustava s čistim (klimatiziranim) zrakom kako bi se sustav (senzor) doveo u referentno (normalno) stanje. Zatim je provedeno testiranje izlaganja sustava butanu pomoću upaljača koji je bio u blizini senzora za mjerenje koncentracije plinova. To je provedeno tako što se upaljač držao upaljenim otprilike 3 sekunde kako bi plin koji se nalazi u upaljaču izašao i mogao utjecati na senzor odnosno sustav. Grafički prikaz vrijednosti koncentracije plinova u zraku prikazan je sljedećom slikom.



Slika 4.6. *Grafički prikaz vrijednosti senzora na utjecaj butana*

Rezultate je bilo potrebno normalizirati kako bi se dobila ispravnija promjenu koncentracije plinova tokom vremena. Koncentracija plinova je prešla preko 20 000 ppm, ali je normalizirana na 1000 ppm. Na sljedećoj slici prikazan je graf koncentracije plinova kako bi promjene bile lakše uočljive.



Slika 4.7. *Grafički prikaz koncentracije plinova u zraku pod uticajem butana*

U ovom slučaju vrijednost koncentracije plinova normalizirana je na 2000ppm. Trenutak kada koncentracija plinova naglo poraste poklapa se s trenutku u kojem je upaljač bio upaljen. U ovom slučaju koncentracija plinova je veća u odnosu na slučaj kada je sustav bio izložen alkoholu. Nakon naglog porasta slijedi postepeni pad vrijednosti odnosno stabiliziranje koncentracije plinova u zraku, bilo je potrebno preko 300 uzoraka, koji su ekvilantni vremenu od otprilike 10 minuta, za stabilizaciju. Izračunate su i srednje vrijednosti mjerenja koje prikazuju veću količinu plinova u odnosnu na prošlo testiranje:

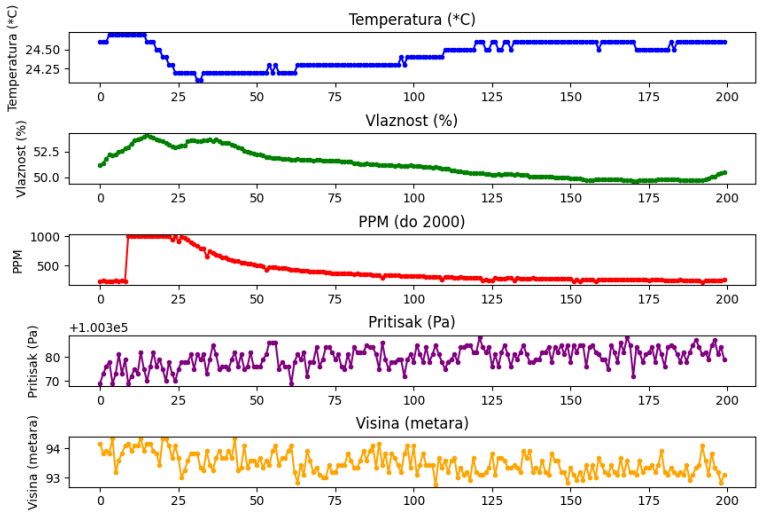
- Srednja vrijednost temperature: 25.39 °C
- Srednja vrijednost vlažnosti: 53.73 %
- Srednja vrijednost PPM: 460.70
- Srednja vrijednost pritiska: 100350.87 Pa
- Srednja vrijednost visine: 95.94 metara

Rezultati temperature i vlažnosti su slični kao u prošlom testiranju, tj. prisutan je pad temperature i vlažnosti zbog klima uređaja. Atmosferski tlak i nadmorska visine imaju slične vrijednosti kao u prethodnom mjerenju.

Testiranje sustava na izlaganje butana s ventilatorom

U ovom slučaju aktiviran je automatski sustav koji će pokrenuti ventilator koji služi za pročišćavanje zraka kada koncentracija plinova pređe određenu vrijednost. Mjerenje je provedeno

u istim uvjetima kao i testiranje bez ventilatora, najrpije je provedena kalibriracija nakon čega je sustav izložen butanu u trajanju od otprilike 3 sekunde i dobiveni su sljedeće rezultate:

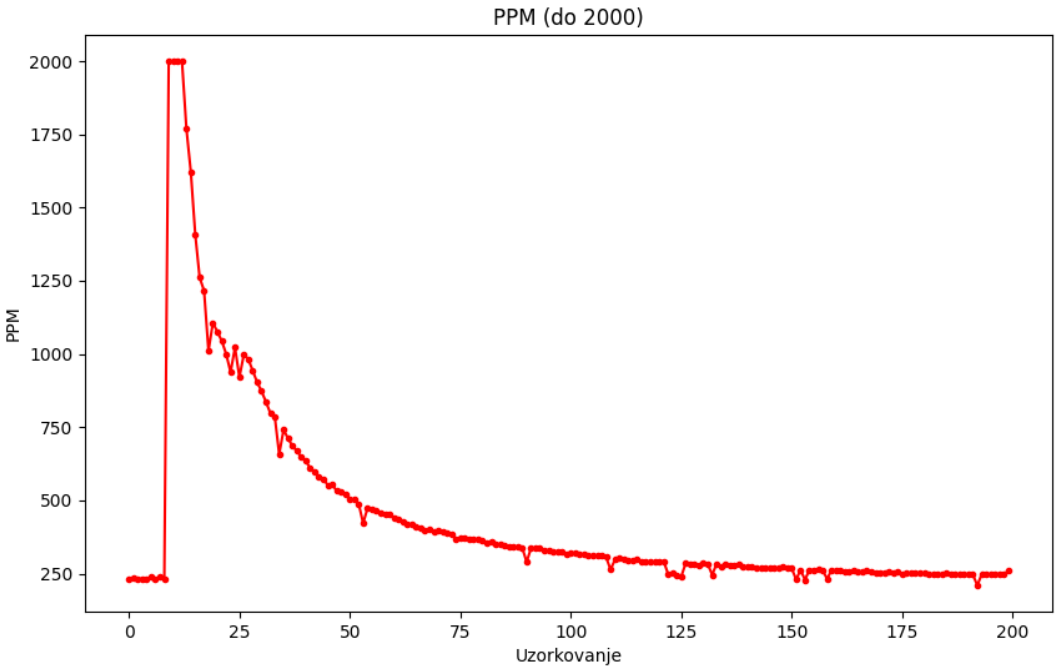


Slika 4.8. *Grafički prikaz vrijednosti senzora na utjecaj butana s ventilatorom*

Ponovno je bilo potrebno normalizirati podatke i uočljivi su slični rezultati kao kod mjerenja bez korištenja ventilatora s tim da sada ima manje uzoraka odnosno bilo je potrebno manje vremena za stabiliziranje zraka zbog uticaja ventilatora. Izračunate su i srednje vrijednosti provedenih mjerenja:

- Srednja vrijednost temperature: 24.44 °C
- Srednja vrijednost vlažnosti: 51.23 %
- Srednja vrijednost PPM: 548.10
- Srednja vrijednost pritiska: 100379.74 Pa
- Srednja vrijednost visine: 93.50 metara

Uočljiva je veća srednju vrijednost za PPM zbog manjeg broja uzoraka. Sljedećom slikom prikazana je ovisnost koncentracije plinova i vremena.



Slika 4.9. *Grafički prikaz vrijednosti senzora na utjecaj butana s ventilatorom*

Rezultati su normalizirani na 2000 ppm kao u mjerenju bez korištenja ventilatora, vidljivo je postepenije smanjivanje koncentracije plinova u zraku i brže stabiliziranje zraka, s tim da je i koncentracija plinova u zraku manja (oko 270 ppm) nego u prošlom mjerenju gdje je bila preko 300 ppm.

5. Zaključak

Sustav za mjerenje kvalitete zraka predstavljen u ovom dokumentu demonstrira uspješnu implementaciju hardverskog i programskog rješenja za praćenje ključnih parametara kvalitete zraka kao što su temperatura, vlažnost, koncentracija plinova, atmosferski tlak i nadmorska visina. Ovaj sustav je značajan alat u kontekstu održivog razvoja i zaštite zdravlja ljudi, s obzirom na sve veću zabrinutost za kvalitetu zraka uslijed urbanizacije i industrijskog razvoja.

Za realizaciju projekta korištene su različite komponente i alati, uključujući mikroupravljač ESP32, senzore DHT22, MQ135 i BMP180 te Adafruit IO web server za pohranu i prikaz podataka. Hardverski dio projekta uključuje povezivanje senzora i mikroupravljača, dok je programsko rješenje fokusirano na prikupljanje podataka, njihovu obradu i slanje na web server.

U testiranju sustava za mjerenje kvalitete zraka analizirani su podaci sa i bez korištenja ventilatora. Primijećeno je da se kvaliteta zraka značajno brže stabilizirala kada je korišten ventilator. Bez ventilatora, koncentracija plinova u zraku pokazala je nagle poraste i padove, dok su s ventilatorom ti prijelazi bili postupniji i glađi. Korištenje ventilatora kao dio sustava za mjerenje i kontrolu kvalitete zraka pokazalo se vrlo učinkovitom metodom za brzo i učinkovito stabiliziranje koncentracije plinova u zraku, čime se osigurava zdravije i sigurnije okruženje. To jasno naglašava važnost automatiziranih sustava za upravljanje kvalitetom zraka u urbanim i industrijskim područjima gdje je zagađenje zraka često problem.

Na temelju prikupljenih podataka, sustav se pokazao vjerodostojnim no identificirane su i mogućnosti za daljnja poboljšanja. To uključuje bolju kalibraciju senzora, smanjenje elektronskog šuma te integraciju dodatnih senzora za mjerenje drugih zagađivača. Također, dugoročno testiranje u različitim uvjetima okoliša može pomoći u daljnjem poboljšanju pouzdanosti i točnosti sustava.

Zaključeno je da projekt predstavlja značajan korak prema razvoju pristupačnog i učinkovitog sustava za mjerenje kvalitete zraka. Kroz daljnja poboljšanja i testiranja, ovakav sustav može postati ključan alat za praćenje i poboljšanje kvalitete zraka, što je od izuzetne važnosti za javno zdravlje i okoliš.

Ovaj zaključak temelji se na detaljnoj analizi prikupljenih podataka i usporedbi s referentnim vrijednostima čime se potvrđuje pouzdanost i funkcionalnost razvijenog sustava.

6. Literatura

[1] Državni hidrometeorološki zavod, Kvaliteta zraka, https://www.airq.hr/kvaliteta-zraka/, pristup: 15.06.2024.

[2] MQ135 Arduino library, https://hackaday.io/project/3475-sniffing-trinket/log/12363-mq135-arduino-library, pristup: 15.06.2024.

[3] Espressif Systems ESP32-WROOM-32 MCU Modules, https://eu.mouser.com/new/espressif/espressif-esp32-wroom-32-modules/, pristup: 16.06.2024.

[4] DHT22 temperature-humidity sensor, https://www.adafruit.com/product/385, pristup: 16.06.2024.

[5] DHT-22 INTERFACING IN RASPBERRY PI WITHOUT USING LIBRARY, https://medium.com/@zordakal171/dht-22-interfacing-in-raspberry-pi-without-using-library-7ca61e8af4b8, pristup: 16.06.2024.

[6] MQ135 Air Quality Sensor : Pin Configuration, Working & Its Applications, https://www.elprocus.com/mq135-air-quality-sensor/, pristup: 16.06.2024.

[7] KKM: BMP180, https://soldered.com/hr/learn/kkm-bmp180/, pristup: 16.06.2024.

[8] Using the Arduino Software (IDE), https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide/, pristup: 16.06.2024.

[9] MQ135 GAS SENSOR, https://github.com/NuclearPhoenixx/MQ135/blob/master/README.md, pristup: 18.06.2024.

[10] BMP180 Digital pressure sensor, https://hr.mouser.com/datasheet/2/783/BST-BMP180-DS000-1509579.pdf, pristup: 20.06.2024.

Prilog A: Programski kod

```
//Uključivanje biblioteka
#include "config.h"
#include <Arduino.h>
#include <WiFi.h>
#include "AdafruitIO_WiFi.h"
#include <Wire.h>
#include <math.h>


// Adafruit IO podaci za pristup
#define IO_USERNAME "ilijajazz"
#define IO_KEY "aio_buZC22bN4rkNSpP1hyYHBVXRPzUI"
#define WIFI_SSID "MillenniumCaffe"
#define WIFI_PASS "pitajBrunE1304?"


// Definiranje pinova za wifi
#if defined(USE_AIRLIFT) || defined(ADAFRUIT_METRO_M4_AIRLIFT_LITE) || \
    defined(ADAFRUIT_PYPORTAL)
// Konfiguracija pinova za ESP32 konekciju
#if !defined(SPIWIFI_SS) // ako WiFi definicija nije u varijanti ploče
#define SPIWIFI SPI
#define SPIWIFI_SS 10 // Chip select pin
#define NINA_ACK 9    // BUSY ili READY pin
#define NINA_RESETN 6 // Reset pin
#define NINA_GPIO0 -1 // Nije spojeno
#endif
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS, SPIWIFI_SS,
NINA_ACK, NINA_RESETN, NINA_GPIO0, &SPIWIFI);
#else
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
#endif


// GPIO pinovi za senzore
#define DHT22_PIN 25 // Pin za DHT22 senzor
#define RELAY_PIN 32 // Pin za relej
#define PIN_MQ135 36 // Pin za MQ135 senzor
#define LED_PIN 14   // Pin za LED diodu
#define BTN_PIN 33   // Pin za gumb


// Konstante za MQ135 senzor
#define RLOAD 10.0      // Otpor na ploči (u kOhm)
#define RZERO 76.63     // Kalibracijski otpor pri atmosferskom CO2 nivou
#define ATMOCO2 415.58  // Atmosferski CO2 nivo za kalibraciju
#define CORA 0.00035
#define CORB 0.02718
#define CORC 1.39538
#define CORD 0.0018
#define CORE -0.003333333
#define CORF -0.001923077
#define CORG 1.130128205
#define PARA 116.6020682
#define PARB 2.769034857
```

```
// Adresa I2C za BMP085/BMP180
#define BMP085_I2CADDR 0x77

// BMP085 registri
#define BMP085_CAL_AC1 0xAA
#define BMP085_CAL_AC2 0xAC
#define BMP085_CAL_AC3 0xAE
#define BMP085_CAL_AC4 0xB0
#define BMP085_CAL_AC5 0xB2
#define BMP085_CAL_AC6 0xB4
#define BMP085_CAL_B1  0xB6
#define BMP085_CAL_B2  0xB8
#define BMP085_CAL_MB  0xBA
#define BMP085_CAL_MC  0xBC
#define BMP085_CAL_MD  0xBE
#define BMP085_CONTROL 0xF4
#define BMP085_TEMPDATA 0xF6
#define BMP085_PRESSUREDATA 0xF6
#define BMP085_READTEMPCMD 0x2E
#define BMP085_READPRESSURECMD 0x34

// Načini rada BMP085
#define BMP085_ULTRALOWPOWER 0
#define BMP085_STANDARD 1
#define BMP085_HIGHRES 2
#define BMP085_ULTRAHIGHRES 3


// Globalne varijable za kalibracijske podatke za BMP180
int16_t ac1, ac2, ac3, b1, b2, mb, mc, md;
uint16_t ac4, ac5, ac6;
uint8_t oversampling;

//Globalne varijable za spremanje vrijednosti senzora
float resistance;
float PPM;
float temperature;
float humidity;
int pressure;
float altitude;

//Timeri za slanje i ispis podataka
unsigned long previousMillisPrint = 0;
unsigned long previousMillisSend = 0;
const long intervalPrint = 2000; // Interval za ispis u ms (2 sekunde)
const long intervalSend = 60000; // Interval za slanje na Adafruit IO u ms
(60 sekundi)

// Inicijalizacija Adafruit IO feed-ova za temperature, vlažnost, ppm, tlak i
visinu
AdafruitIO_Feed *temperature_feed = io.feed("temperature");
AdafruitIO_Feed *humidity_feed = io.feed("humidity");
AdafruitIO_Feed *ppm_feed = io.feed("ppm");
AdafruitIO_Feed *pressure_feed = io.feed("pressure");
AdafruitIO_Feed *altitude_feed = io.feed("altitude");


//////// Rad s BMP 180 senzorom ////////
// Inicijalizacija BMP180 senzora
bool begin(uint8_t mode) {
    if (mode > BMP085_ULTRAHIGHRES)
```

```
        mode = BMP085_ULTRAHIGHRES;
    oversampling = mode;

    if (read8(0xD0) != 0x55)
        return false;

    ac1 = read16(BMP085_CAL_AC1);
    ac2 = read16(BMP085_CAL_AC2);
    ac3 = read16(BMP085_CAL_AC3);
    ac4 = read16(BMP085_CAL_AC4);
    ac5 = read16(BMP085_CAL_AC5);
    ac6 = read16(BMP085_CAL_AC6);
    b1 = read16(BMP085_CAL_B1);
    b2 = read16(BMP085_CAL_B2);
    mb = read16(BMP085_CAL_MB);
    mc = read16(BMP085_CAL_MC);
    md = read16(BMP085_CAL_MD);

    return true;
}

// Izračun B5 vrijednosti
int32_t computeB5(int32_t UT) {
    int32_t X1 = (UT - (int32_t)ac6) * ((int32_t)ac5) >> 15;
    int32_t X2 = ((int32_t)mc << 11) / (X1 + (int32_t)md);
    return X1 + X2;
}

// Čitanje sirove temperature za BMP 180 radi dobivanja podataka za izračun
atmosferskog pritiska
uint16_t readRawTemperature(void) {
    write8(BMP085_CONTROL, BMP085_READTEMPCMD);
    delay(5); // 5ms kašnjenje
    return read16(BMP085_TEMPDATA);
}

// Čitanje sirovog tlaka
uint32_t readRawPressure(void) {
    uint32_t raw;

    write8(BMP085_CONTROL, BMP085_READPRESSURECMD + (oversampling << 6));

    if (oversampling == BMP085_ULTRALOWPOWER)
        delay(5); // 5ms
    else if (oversampling == BMP085_STANDARD)
        delay(8); // 8ms
    else if (oversampling == BMP085_HIGHRES)
        delay(14); // 14ms
    else
        delay(26); // 26ms

    raw = read16(BMP085_PRESSUREDATA);
    raw <<= 8;
    raw |= read8(BMP085_PRESSUREDATA + 2);
    raw >>= (8 - oversampling);

    return raw;
}

// Čitanje amotsferskog pritiska
int32_t readPressure(void) {
```

```
    int32_t UT, UP, B3, B5, B6, X1, X2, X3, p;
    uint32_t B4, B7;

    UT = readRawTemperature();
    UP = readRawPressure();

    B5 = computeB5(UT);

    B6 = B5 - 4000;
    X1 = ((int32_t)b2 * ((B6 * B6) >> 12)) >> 11;
    X2 = ((int32_t)ac2 * B6) >> 11;
    X3 = X1 + X2;
    B3 = ((((int32_t)ac1 * 4 + X3) << oversampling) + 2) / 4;

    X1 = ((int32_t)ac3 * B6) >> 13;
    X2 = ((int32_t)b1 * ((B6 * B6) >> 12)) >> 16;
    X3 = ((X1 + X2) + 2) >> 2;
    B4 = ((uint32_t)ac4 * (uint32_t)(X3 + 32768)) >> 15;
    B7 = ((uint32_t)UP - B3) * (uint32_t)(50000UL >> oversampling);

    if (B7 < 0x80000000) {
        p = (B7 * 2) / B4;
    } else {
        p = (B7 / B4) * 2;
    }
    X1 = (p >> 8) * (p >> 8);
    X1 = (X1 * 3038) >> 16;
    X2 = (-7357 * p) >> 16;

    p = p + ((X1 + X2 + (int32_t)3791) >> 4);
        return p;
}

// Čitanje nadmorske visine
float readAltitude(float sealevelPressure) {
    float altitude;
    float pressure = readPressure();
    altitude = 44330 * (1.0 - pow(pressure / sealevelPressure, 0.1903));
    return altitude;
}

// Čitanje 8-bitnog registra preko I2C
uint8_t read8(uint8_t reg) {
    Wire.beginTransmission(BMP085_I2CADDR);
    Wire.write(reg);
    Wire.endTransmission();
    Wire.requestFrom(BMP085_I2CADDR, 1);
    return Wire.read();
}

// Čitanje 16-bitnog registra preko I2C
uint16_t read16(uint8_t reg) {
    uint16_t value;
    Wire.beginTransmission(BMP085_I2CADDR);
    Wire.write(reg);
    Wire.endTransmission();
    Wire.requestFrom(BMP085_I2CADDR, 2);
    value = (Wire.read() << 8) | Wire.read();
    return value;
}
```

```
// Pisanje 8-bitnog registra preko I2C
void write8(uint8_t reg, uint8_t value) {
    Wire.beginTransmission(BMP085_I2CADDR);
    Wire.write(reg);
    Wire.write(value);
    Wire.endTransmission();
}
//////// Kraj koda za rad s BMP 180 senzorom ////////


//////// Rad s DHT22 senzorom ////////
// Čitanje podataka s DHT22 senzora
bool readDHT22(int *data) {
    uint8_t lastState = HIGH;
    uint8_t counter = 0;
    uint8_t j = 0, i;

    // Inicijacija komunikacije s DHT22 senzorom
    pinMode(DHT22_PIN, OUTPUT);
    digitalWrite(DHT22_PIN, LOW);
    delay(20);
    digitalWrite(DHT22_PIN, HIGH);
    delayMicroseconds(30);
    pinMode(DHT22_PIN, INPUT);

    // Čitanje odgovora s DHT22 senzora
    for (i = 0; i < 85; i++) {
        counter = 0;
        while (digitalRead(DHT22_PIN) == lastState) {
            counter++;
            delayMicroseconds(1);
            if (counter == 255) {
                break;
            }
        }
        lastState = digitalRead(DHT22_PIN);

        if (counter == 255) {
            break;
        }

        if ((i >= 4) && (i % 2 == 0)) {
            data[j / 8] <<= 1;
            if (counter > 16) {
                data[j / 8] |= 1;
            }
            j++;
        }
    }

    // Provjera checksuma
    if ((j >= 40) && (data[4] == ((data[0] + data[1] + data[2] + data[3]) &
0xFF))) {
        return true;
    } else {
        return false;
    }
}
//////// Kraj koda za rad s DHT22 senzorom ////////
```

```
//////// Rad s MQ135 senzorom ////////
// Dobivanje korekcijskog faktora za MQ135 senzor
float getCorrectionFactor(float t, float h) {
    if (t < 20) {
        return CORA * t * t - CORB * t + CORC - (h - 33.0) * CORD;
    } else {
        return CORE * t + CORF * h + CORG;
    }
}

// Dobivanje otpora MQ135 senzora
float getResistance(int pin) {
    int val = analogRead(pin);
    return ((4095.0 / (float)val) - 1.0) * RLOAD;
}

// Dobivanje korigiranog otpora MQ135 senzora
float getCorrectedResistance(float resistance, float t, float h) {
    return resistance / getCorrectionFactor(t, h);
}

// Dobivanje PPM vrijednosti CO2
float getPPM(float resistance, float t, float h) {
    return PARA * pow((getCorrectedResistance(resistance, t, h) / RZERO), -
PARB);
}
//////// Kraj koda za rad s MQ135 senzorom ////////


// Inicijalizacija
void setup() {
    //Postavljanje pinova za rad s senzorima
    pinMode(RELAY_PIN, OUTPUT);
    pinMode(LED_PIN, OUTPUT);
    pinMode(BTN_PIN, INPUT_PULLUP);

    //Postavljanje serijske komunikacije
    Serial.begin(115200);

    // Inicijalizacija I2C komunikacije
    Wire.begin();
    while (!Serial); // Čeka se otvaranje serijskog monitora

    Serial.print("Spajanje na Adafruit IO ");
    io.connect(); // Spajanje na Adafruit IO

    // Čeka se uspostava veze
    while (io.status() < AIO_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println();
    Serial.println(io.statusText()); // Ispis statusa veze

    // Provjera inicijalizacije BMP180
    if (!begin(BMP085_STANDARD)) {
        Serial.println("Inicijalizacija BMP180 nije uspjela!");
        while (1);
    }else{
        Serial.println("Inicijalizacija BMP085 uspješna!");
    }
```

```
    // Provjera inicijalizacije MQ135 senzora
    if (getResistance(PIN_MQ135) > 0) {
        Serial.println("Inicijalizacija MQ135 senzora uspješna!");
    } else {
        Serial.println("Nije uspjela inicijalizacija MQ135 senzora!");
    }

    // Provjera inicijalizacije DHT22 senzora
    int checkData[5] = {0, 0, 0, 0, 0}; // Polje za podatke o temperaturi i
vlažnosti
    while (!readDHT22(checkData)) {
        Serial.println("Nije uspjela inicijalizacija DHT22 senzora!");
    }
    Serial.println("Inicijalizacija DHT22 senzora uspješna!");

    Serial.println();
    delay(1000);  // Čekanje 1 sekundu za stabilizaciju serijske komunikacije
}

// Glavna petlja
void loop() {
    io.run(); // Pokretanje Adafruit IO komunikacije

    unsigned long currentMillis = millis();

    // Ispis podataka svakih 2 sekunde
    if (currentMillis - previousMillisPrint >= intervalPrint) {
        previousMillisPrint = currentMillis;

        int data[5] = {0, 0, 0, 0, 0}; // Polje za podatke o temperaturi i
vlažnosti

        // Čitanje podataka s DHT22 senzora
        resistance = getResistance(PIN_MQ135);
        pressure = readPressure();
        if (readDHT22(data) && resistance < 200 && (pressure > 0 && pressure
< 150000)) {
            temperature = data[2] & 0x7F;
            temperature *= 256;
            temperature += data[3];
            temperature *= 0.1;

            if (data[2] & 0x80) {
                temperature *= -1;
            }

            humidity = data[0];
            humidity *= 256;
            humidity += data[1];
            humidity *= 0.1;

            PPM = getPPM(resistance, temperature, humidity);
            altitude = readAltitude(101500); // Tlak na razini mora u Pa

            Serial.print("Temperatura: ");
            Serial.print(temperature);
            Serial.println(" *C");
            Serial.print("Vlaznost: ");
            Serial.print(humidity);
            Serial.println(" %\t");
```

```
            Serial.print("PPM: ");
            Serial.print(PPM);
            Serial.println(" ppm");
            Serial.print("Pritisak = ");
            Serial.print(pressure);
            Serial.println(" Pa");
            Serial.print("Visina = ");
            Serial.print(altitude);
            Serial.println(" metara");
            Serial.println();

        } else {
            Serial.println("Neuspjelo čitanje podataka s jednim od
senzora.");
        }

        // Upravljanje LED diodom i relejem prema uvjetima
        if (digitalRead(BTN_PIN) == LOW) {
            digitalWrite(LED_PIN, HIGH);
            digitalWrite(RELAY_PIN, HIGH);
            Serial.println("Prskalice su uključene.");
        } else if (PPM > 1000.0) {
            digitalWrite(RELAY_PIN, HIGH);
            if (temperature > 27) {
                digitalWrite(LED_PIN, HIGH);
                Serial.println("Prskalice su uključene.");
            } else {
                digitalWrite(LED_PIN, LOW);
            }
        } else {
            digitalWrite(RELAY_PIN, LOW);
            digitalWrite(LED_PIN, LOW);
        }
    }

    // Slanje podataka na Adafruit IO svake minute
    if (currentMillis - previousMillisSend >= intervalSend) {
        previousMillisSend = currentMillis;

        temperature_feed->save(temperature);
        humidity_feed->save(humidity);
        ppm_feed->save(PPM);
        pressure_feed->save(pressure);
        altitude_feed->save(altitude);
    }
}
```

Prilog A. programski kod