# BEŽIČNA SENZORSKA MREŽA ZA MJERENJE KVALITETE ZRAKA

**Botica, Ante**

**Master's thesis / Specijalistički diplomski stručni**

**2019**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Split / Sveučilište u Splitu**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:228:682899

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-07-27**

*Repository / Repozitorij:*

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

**SVEUČILIŠTE U SPLITU**

**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Specijalistički diplomski stručni studij Elektrotehnike

**ANTE BOTICA**

# ZAVRŠNI RAD

## BEŽIČNA SENZORSKA MREŽA ZA MJERENJE KVALITETE ZRAKA

Split, rujan 2019

# SVEUČILIŠTE U SPLITU

## SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Specijalistički diplomski stručni studij Elektrotehnike


**Predmet:** Senzorske mreže

# ZAVRŠNI RAD


**Kandidat**: Ante Botica


**Naslov rada**: Bežična senzorska mreža za mjerenje kvalitete zraka


**Mentor**: dr. sc. Tonko Kovačević



Split, rujan 2019

# SADRŽAJ


Sažetak **5**

1. UVOD **6**

**2. BEŽIČNE KOMUNIKACIJSKE TEHNOLOGIJE** **7**

2.1. Princip bežičnog prijenosa informacija 7

2.2. Prigušenje radio valova 8

2.3. Popularne bežične komunikacijske tehnologije 9

**3. ZIGBEE PROTOKOL** **10**

3.1 Tipovi i režimi rada ZigBee uređaja 10

3.2. ZigBee protokolni stog 12

**4. RAZVOJNA PLATFORMA** **17**

4.1. XBee ZigBee S2c 18

4.1.1. UART 21

4.1.2. XCTU 22

4.1.3. API i AT režimi rada 23

4.2. Arduino platforma 35

4.3. BME280 38

4.4. MH-Z19 40

4.4.1. Princip rada 41

4.4.2. UART komunikacija 43

**5. ARHITEKTURA MREŽE** **45**

5.1. Arhitektura koordinatorskog mrežnog čvora 46

5.2. Arhitektura mjeriteljskog mrežnog čvora 47

**6. IMPLEMENTACIJA** **48**

6.1. Algoritam ciklusa rada 50

6.1. Bitne Arduino biblioteke i funkcije 55

6.2. Raspberry Pi alati i spremanje podataka u datoteku 56

**7. ZIGBEE SIGURNOST I ZAŠTITA PODATAKA** **59**

7.1. Implementacija sigurnosnih mehanizama 59

**8. ZAKLJUČAK** **61**

LITERATURA 62

POPIS SLIKA 63

POPIS TABLICA 65

PRILOZI **66**

**Prilog 1 - Korišteni programski kodovi za Arduino mikrokontrolere** **67**

1. Arduino programski kod koordinatora 67

2. Arduino programski kod mjeriteljskog čvora 72

## Sažetak

**Bežična senzorska mreža za mjerenje kvalitete zraka**

Problematika razmatrana u ovom radu obuhvaća projektiranje te realizaciju bežične senzorske mreže unutar domene *ZigBee* tehničkog standarda čija je svrha nadgledanje kvalitete zraka u zatvorenim prostorima. Navedeni ishodi rada su ostvareni putem *Arduino* platforme i pripadajućih *XBee S2C* radio modula (u nastavku: RF modula) te odgovarajuće senzorske opreme. ZigBee protokol odabran za navedenu svrhu konceptualno zadovoljava sve sigurnosne, prostorno ekonomske, kapacitetske te podatkovno protočne zahtjeve na tipičnu bežičnu senzorsku mrežu.

***Ključne riječi:*** *ZigBee protokol, Arduino, bežična senzorska mreža*

## Summary

**Wireless sensor network for air quality monitoring**

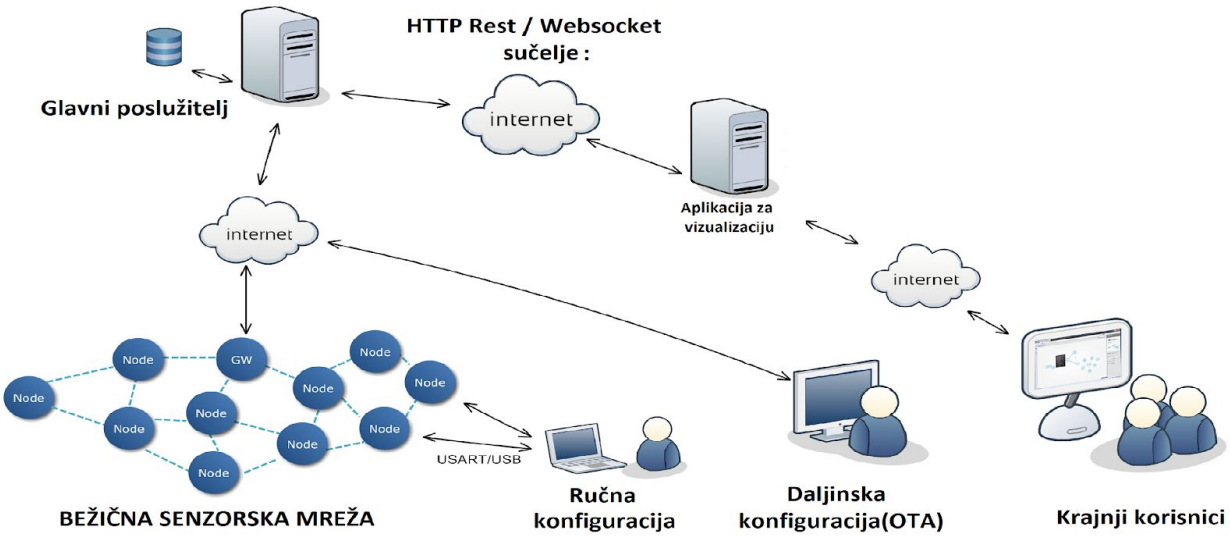
*The issues discussed in this paper include the design and implementation of a wireless sensor network(WSN) accomplished within the ZigBee technical standard domain via the Arduino platform supported by the necessary sensory equipment and the associated XBee S2C radio modules (hereinafter: RF Modules). The purpose of the system is to monitor air quality in enclosed environments. The ZigBee protocol chosen for this purpose conceptually meets all security, space economics, capacity and data flow requirements on a typical wireless sensor network.*

***Keywords:*** *ZigBee protocol, Arduino, wireless sensor network*

# 1. UVOD

Bežične senzorske mreže prostorno su distribuirane računalne mreže koje se sastoje od međusobno udaljenih čvorova čija je namjena mjeriti stanja promatranog procesa te svrsishodno usmjeravati priskrbljene podatke kroz mrežu na željenu lokaciju. Takovi lokalno umreženi sustavi se potom putem *gateway*–a mogu povezati i na internet te na taj način postati dio *IoT*(*Internet of Things*) nadsustava. Uređaje na pojedinim čvorovima u mreži karakteriziraju ograničeni hardverski resursi, mala potrošnja energije te energetska autonomija. Svaki mjeriteljski čvor u bežičnoj senzorskoj mreži se sastoji od osjetnika, RF modula, izvora napajanja te opcionalnog mikrokontrolera.

U sljedećim su poglavljima koračno, kroz primjere razrađeni svi segmenti u postupku konfiguriranja samih RF modula, kreiranja komunikacijskih okvira, uspostavljanja komunikacije između pojedinih mrežnih čvorova te implementiranja navedenog u vidu projektiranja i realizacije jednostavne bežične senzorske mreže za praćenje parametara kvalitete zraka te spremanje istih u *".csv"* datoteku putem *Raspberry Pi gateway*-a.



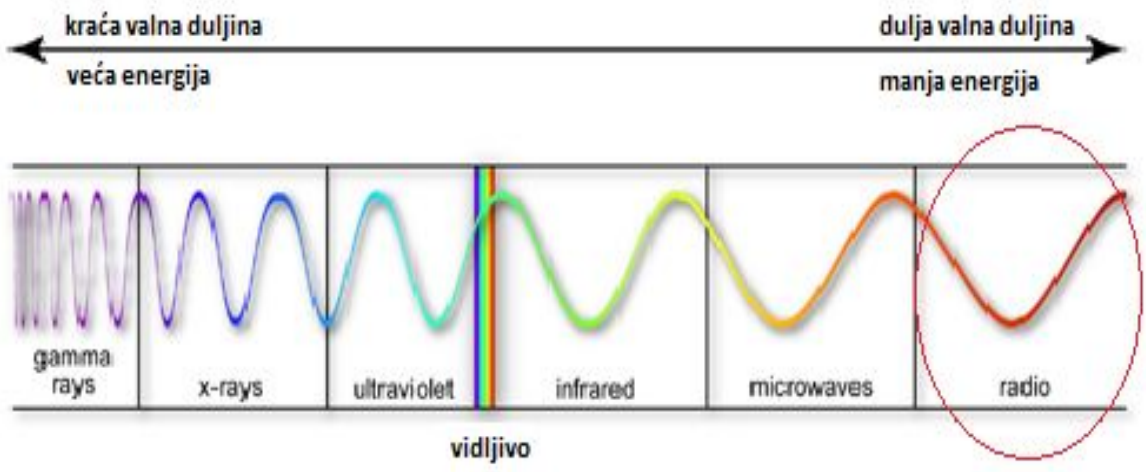

Slika 1. Prikaz bežične senzorske mreže implementirane u Internet Of Things(IOT) koncept

# 2. BEŽIČNE KOMUNIKACIJSKE TEHNOLOGIJE

## 2.1. Princip bežičnog prijenosa informacija

Bežične komunikacijske tehnologije su skupovi pojmova koji podrazumijevaju prijenos informacija kroz prostor na neku udaljenost bez posredne upotrebe kabela, žica ili bilo kakvih drugih vodiča električne energije. Bežična komunikacija se zasniva na prinicipu propagiranja elektromagnetskih signala između dvaju ili više umreženih funkcijski podržanih uređaja(predajnika i prijemnika) putem zračnog ili atmosferskog medija u obliku radio valova, svjetlosti, zvuka te električnih ili magnetskih polja. Elektromagnetski valovi posljedica su gibanja elektrona u prostoru određenom frekvencijom osciliranja koja ujedno uvjetuje i količinu informacija koju elektromagnetski signal može prenijeti. Informacije koje se prenose bežičnim komunikacijskim tehnologijama utiskuju se u prijenosni signal mijenjajući njegove fizičke parametre u skladu sa promjenom prenošenih podataka putem različitih tehnika modulacije(AM, FM, PM,itd...).

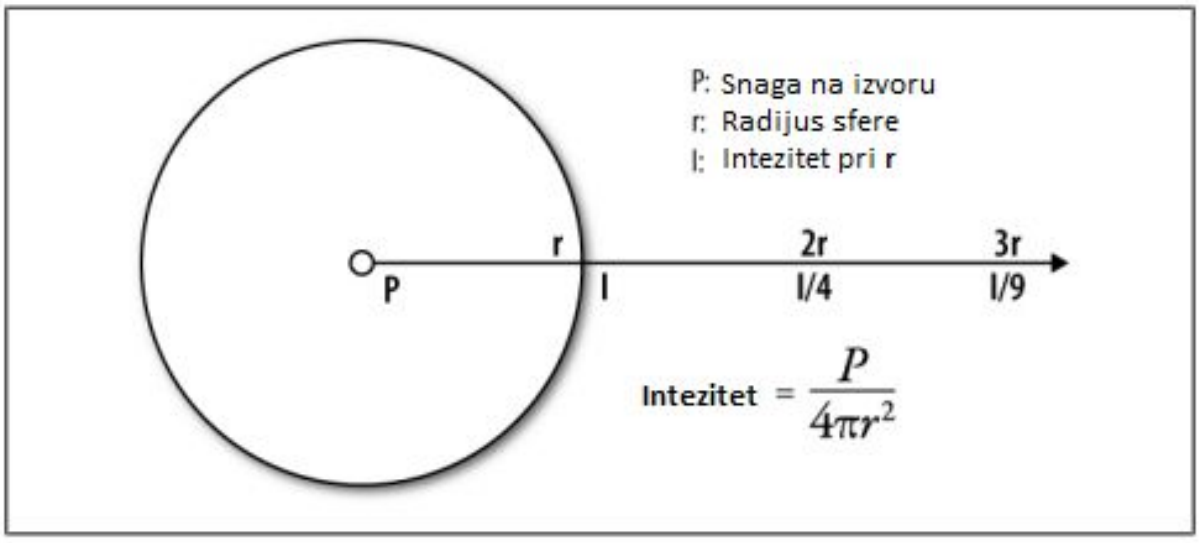
U ovome radu korištena je ZigBee bežična komunikacijska tehnologija koja se zasniva na prijenosu informacija unutar dijela elektromagnetskog spektra koji se odnosi na radio valove.




Slika 2. Spektar elektromagnetskog zračenja

## 2.2. Prigušenje radio valova

Kod bežičnog prijenosa informacija radio valovima povećavanjem udaljenosti između predajnog i primaćeg uređaja dolazi do ubrzane atentuacije signala. Slabljenje signala se događa uslijed njegovog širenja kroz okolni prostor po zakonu obrnutih kvadrata. Pri svakom udvostručenju udaljenosti između prijamnika i predajnika potrebna je četverostruko veća snaga izvorišnog signala za prijenos željenih informacija do odredišnog čvora. *Mesh* topologija bežičnih mreža podržana na ZigBee i sličnim tehnologijama eliminira potrebu za korištenjem jačih odašiljača pri prijenosu informacija na udaljene odredišne čvorove jer se iste proslijeđuju do odredišta posredno preko obližnjih mrežnih čvorova(*routera*) u takozvanim skokovima(eng. *hops*) na osnovi čega se povećanjem broja čvorova u mreži postiže i veća pouzdanost, doseg te energetska ekonomičnost iste.




Slika 3. Zakon obrnutih kvadrata

## 2.3. Popularne bežične komunikacijske tehnologije

Prethodno spomenute metode bežične komunikacije su definirane i specificirane u okvirima dogovorno uspostavljenih tehničkih standarda i protokola koji objedinjuju sva pravila i aspekte pri razmatranju implementacije određene tehnologije u razvojnoj, industrijskoj, medicinskoj ili opće komercijalnoj domeni primjene iste. Budući da su takvi tehnički standardi uglavnom sporazumno definirani od strana više različitih proizvođača svrsi srodnih ili funkcijski komplementarnih elektroničkih uređaja isti međusobno zadovoljavaju uvjete djelomične ili potpune otvorenosti u vidu kompatibilnosti i interoperabilnosti podržanih uređaja za neki mrežni protokol uz iznimke koje zbog sigurnosnih nadogradnji te namjensko profiliranje mogu ograničiti primjenu nekih uređaja na određene mrežne sustave. U nastavku ovoga poglavlja tablično su navedeni te uspoređeni neki od bežičnih mrežnih standarda koji svojim specifikacijama i aplikativnim paletama konkuriraju ovdje korištenoj ZigBee tehnologiji.

Tablica 1. Usporedba nekih popularnih bežičnih tehnologija

| **Tehnologija** | ZigBee | Z–Wave | Thread | Wi–Fi | Bluetooth Mesh |
|---|---|---|---|---|---|
| **Fizički domet** | 10–20m | 30m | 30m | 50–90m | 100m |
| **Najveći mogući broj umreženih uređaja** | 65000 | 232 | 250–300 | 250(po pristupnoj točki) | 32000 |
| **Brzina protoka podataka** | 40–250 kbps | 9.6–100 kbps | 250 kbps | 54–600Mbps | 1 Mbps |
| **Frekvencija** | 915 MHz/2.4 GHz | 908/916 MHz | 2.4 GHz | 2.4/5 GHz | 2.4 GHz |
| **Topologija mreže** | Mesh | Mesh | Mesh | Mesh | Mesh |
| **Podrška za IP** | Ne | Ne | Ne | Da | Ne |

# 3. ZIGBEE PROTOKOL

**ZigBee** je bežični komunikacijski tehnički standard utemeljen na **802.15.4** IEEE normi koji odgovara na zahtjeve osobnih bežičnih mreža(eng. *WPAN*) male podatkovne propusnosti, male potrošnje energije uz visoku razinu sigurnosti te mogućnost umrežavanja velikog broja uređaja. ZigBee protokol zbog svoje konceptualne jednostavnosti, fleksibilnosti, sigurnosti, ekonomičnosti, robusnosti te mrežno–kapacitetnih gabarita je pogodan za široku aplikativnu paletu u različitim tržišnim domenama. Osobito je pogodan za umrežavanje međusobno udaljenih senzorskih, kontrolnih i aktuatorskih komponenti automatiziranih industrijskih ili stambenih sustava te mjernih instrumentacija u medicini i ostalim područjima primjene sa sličnim zahtjevima na mrežu.

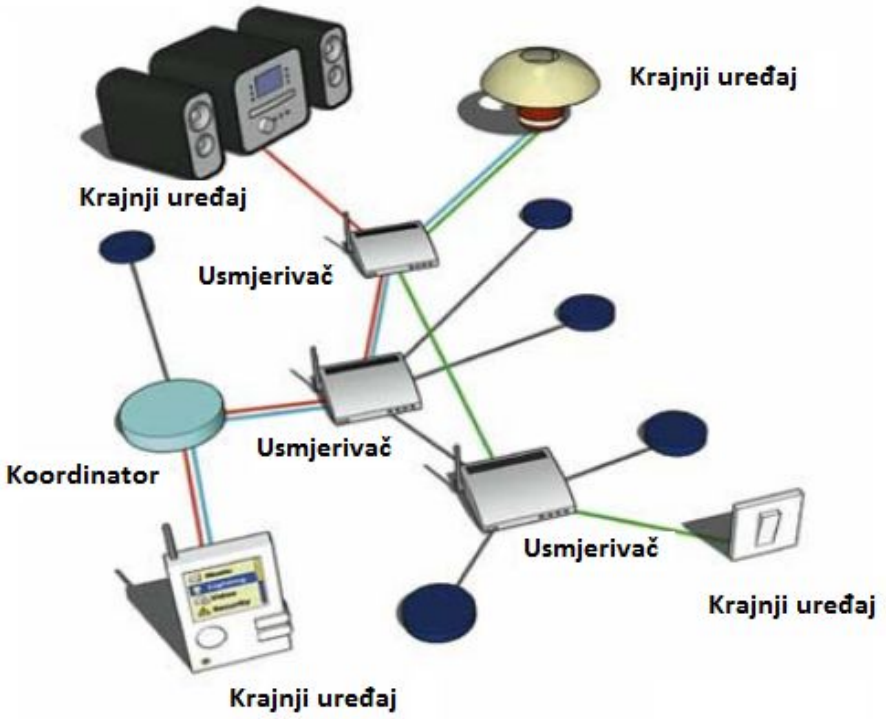
## 3.1 Tipovi i režimi rada ZigBee uređaja

Po IEEE normi definirane su dvije funkcionalne klase ZigBee uređaja: FFD(eng. *Fully Functional Device* – uređaj potpune funkcionalnosti) te RFD(eng. *Reduced Functional Device* – uređaj ograničene funkcionalnosti). FFD uređaji zahtjevaju stabilan i stalan izvor napajanja dok su RFD uređaji uglavnom energetski autonomni te napajani baterijama. ZigBee uređaji su također podijeljeni i po režimu rada, odnosno svojoj logičkoj funkciji u mrežnom sloju ZigBee protokola, razlikujemo koordinatore(FFD), usmjerivače(FFD) te krajnje uređaje(RFD)

. Tablica 2. Tipovi i klase ZigBee uređaja

| ZigBee uređaj | IEEE klasa uređaja | Logička funkcija u mreži |
|---|---|---|
| Koordinator | FFD | Uspostavlja mrežu, dodjeljuje adrese, osigurava ispravnu i sigurnu razmjenu podataka između pojedinih čvorova. |
| Usmjerivač | FFD | Povećava fizički domet mreže, omogućava pristup većeg broja uređaja na mrežu. |
| Krajnji uređaj | RFD | Senzori, kontrolni i izvršni uređaji |

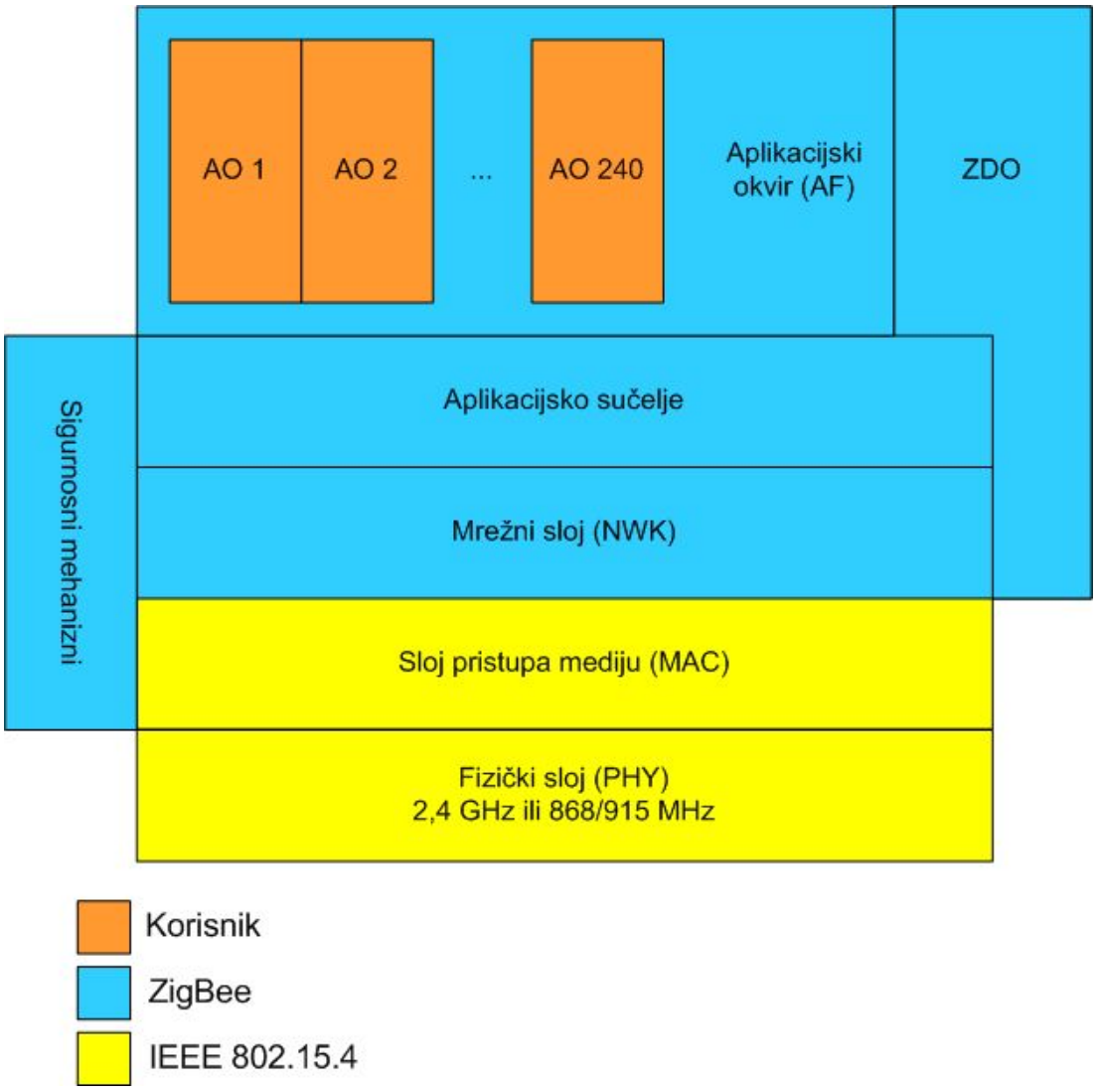
Mrežni koordinator i usmjerivači zahtjevaju stalnu prisutnost na mreži pa samim time i stalno napajanje električnom energijom dok se krajnji uređaji mogu držati u "uspavanom" stanju iz kojeg se periodički mogu "buditi" na zahtjev prema potrebama sustava. Koordinator je zadužen za uspostavljanje mreže, dodjelu mrežnih adresa ostalim uređajima te osiguravanje svih aspekata ispravne i sigurne komunikacije između čvorova te je neophodan element bežične mreže. Usmjerivači povećavaju fizički domet i kapacitetne gabarite mreže te imaju repetitivnu funkciju u odnosu na prenošene podatke koje prosljeđuju do zadane mrežne lokacije. Krajnji uređaji imaju funkciju prikupljanja informacija iz okoline ili procesa, a mogu biti i aktuatorski izvršni te nadzorni elementi umreženog sustava.




Slika 4. Prikaz funkcijski različitih ZigBee uređaja u mreži

### 3.2. ZigBee protokolni stog

ZigBee protokolni stog utemeljen je na OSI referentnom modelu za otvoreno povezivanje sustava definirajući samo one razine koje su od važnosti za osiguravanje optimalne funkcionalnosti na ciljanom području. OSI model na apstraktan način opisuje mehanizme komunikacije između sklopovlja, programa, *software*–a i protokola u mrežnim komunikacijskim tehnologijama kroz podjelu arhitekture mreže na sedam definiranih logičkih funkcijskih slojeva.

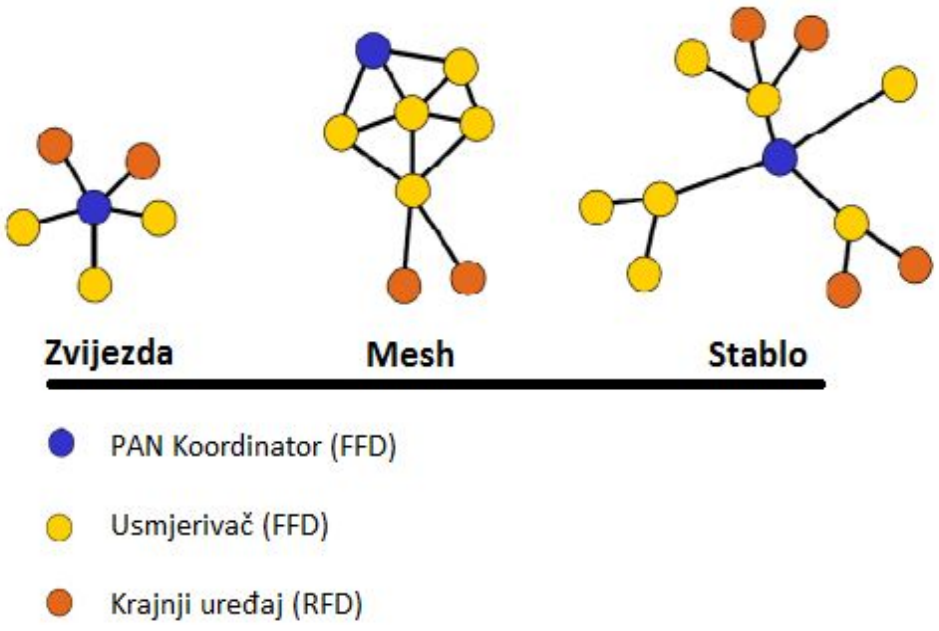



Slika 5. Pojednostavljeni prikaz stoga ZigBee protokola

**Fizički sloj**(eng. *PHY – physical layer*) zadužen je za aktivaciju i deaktivaciju primopredajnika, indikaciju kvalitete veze (eng. LQI – *Link Quality Indicator*), mjerenje energetske razine signala(eng. *ED – Energy Detection*), provjeru dostupnosti kanala (eng. CCA – *Clear Channel Assesment*), odabir kanala te razmjenu podataka. Unutar fizičkog sloja definirana su tri frekventna pojasa: prvi na 868 MHz koji sadrži kanal 0 s podrškom brzine prijenosa podataka do 20 kbps, drugi na 915 MHz sadrži 10 kanala (1 – 10) s podrškom brzine prijenosa podataka do 40 kbps te treći pojas na 2,4 GHz sa 16 kanala (11 – 26) s podržanom brzinom prijenosa podataka do 250 kbps i 5 MHz-nom širinom komunikacijskog kanala. Pri prvom i drugom frekventnom pojasu se koristi binarna fazna modulacija signala(eng. PSK – *Phase Shift Keying*), dok se pri trećem pojasu koristi QPSK (eng. *Quadrature Phase Shift Keying*) tehnika moduliranja signala. Korištenjem modulacije raspršenja spektra direktnim postupkom (eng. *DSSS – Direct Sequence Spread Spectrum*) sigalom nosiocem se modulira signal pseudo šuma čime se postiže veća otpornost na interferencije unutar mreže te bolji odnos signal/vanjski šum.

**Sloj pristupa mediju**(eng. *MAC – Medium Access Layer*) definira načine na koje će 802.15.4 radio uređaji u mreži dijeliti zračni medij i pristupati istom te funkcionalnost pojedinih uređaja. Zadužen je za podršku pristupa fizičkom sloju, pokretanje koordinatora i generiranje PAN ID-a (eng. *Personal Area Network Identifier*), definiranje i provedbu GTS(eng. *Guaranteed Time Slot*) mehanizma, implementiranje CSMA/CA(eng. *Carrier-Sense Multiple Access with Collision Avoidance*) pristupnog mehanizma, te povezivanje MAC entiteta na različitim čvorovima u mreži.

Unutar **mrežnog sloja** odvija se pridruživanje uređaja mrežama te se iste uspostavljaju, otkrivaju i napuštaju. Također, isti osigurava ispravno funkcioniranje MAC sloja ispod sebe te pruža sučelje na aplikacijski sloj iznad sebe. Koordinator pri pokušaju uspostave ZigBee mreže izvršava mjerenje energetske razine signala (eng. ED – *Energy Detection*) u svrhu nalaženja optimalnog RF kanala. Uslijed uspješnog odabira RF kanala koordinator dodjeljuje logički identifikator osobne mreže(eng. PAN ID – *Personal Area Network Identifier*) nakon čega se ostali čvorovi mogu pridružiti mreži direktno(implementacijom pune 64-bitne adrese pridružiteljskog čvora u memorijski element susjednog čvora pri fazi projektiranja mreže) ili asocijativno(repetitivnim slanjem zahtjeva za pridruženje na različite kanale dok ne nađu pogodnu mežu za pristup). ZigBee mrežni sloj podržava "mesh", "zvijezda" i "stablo" mrežne topologije. Na slici **6.** prikazane su konfiguracije mogućih topologija mreže unutar ZigBee standarda




Slika 6. Prikaz podržanih topologija ZigBee mreže

U nastavku su tablično kategorizirani različiti mrežni uređaji po ovlastima.

Tablica 3.Usporedba ZigBee uređaja različitih po ovlastima unutar mrežnog sloja

| **Funkcija u mrežnom sloju** | **Koordinator** | **Usmjerivač** | **Krajnji uređaj** |
|---|---|---|---|
| Uspostava ZigBee mreže | ✓ | ✗ | ✗ |
| Izdavanje dozvole za pristup ZigBee mreži | ✓ | ✓ | ✗ |
| Dodjela 16-bitnih mrežnih adresa | ✓ | ✓ | ✗ |
| Otkrivanje i pamćenje trajektorija za učinkovitiju razmjenu podataka | ✓ | ✓ | ✗ |
| Otkrivanje i pamćenje popisa svih čvorova u neposrednoj blizini | ✓ | ✓ | ✗ |
| Usmjeravanje podatkovnih paketa kroz mrežu | ✓ | ✓ | ✗ |
| Primanje/slanje podatkovnih paketa | ✓ | ✓ | ✓ |
| Pridruživanje/napuštanje mreže | ✓ | ✓ | ✓ |
| "Uspavani" režim rada | ✗ | ✗ | ✓ |

**Aplikacijski sloj** je najviši specificirani sloj ZigBee stoga, a se sastoji od više podslojeva. Osigurava sučelje ZigBee sustava prema korisniku te sadrži većinu entiteta definiranih ZigBee specifikijom kao što su ZDO(eng. *ZigBee Device Object*) i njegove upravne procedure uz aplikacijske objekte definirane od strane proizvođača ZigBee podržanih uređaja. Ovaj sloj je također odgovoran i za slanje paketiziranih poruka između povezanih uređaja, upravljanje grupnim adresama, pakiranje i raspakiranje prenošenih podatka te pružanje podrške ZigBee aplikacijskim profilima.

# 4. RAZVOJNA PLATFORMA

U tablici **4.** navedene su i prikazane sve *hardware* komponente korištene pri izradi bežične senzorske mreže.

Tablica 4. Korištene hardware komponente

| PRIKAZ | NAZIV KOMPONENTE | OPIS | KOLIČINA |
|---|---|---|---|
| | Digi XBee S2c ZB RF Module | ZigBee podržani radio primopredajnik, RF modul. | 5 |
| | Arduino NANO | Arduino NANO mikrokontroler. | 5 |
| | MH-Z19 infrared gas sensor for detection of CO2 | Senzor za detekciju razina koncentracije CO2 plina u zraku te posljedično i utvrđivanje same kvalitete zraka. | 4 |
| | BME280 Digital Sensor Module for temperature, humidity and air pressure | Senzor temperature, vlage i tlaka zraka. | 4 |
| | SainSmart XBee USB Explorer | Pomoćni modul za povezivanje terminala RF modula sa mikrokontrolerom te povezivanje istog sa računalnim alatima u svrhu konfiguriranja mrežnih parametara pojedinog čvora. | 5 |
| | Raspberry Pi 3 B+ | Minijaturno računalo bazirano na Linux operacijskom sustavu u službi *gateway*-a za realiziranu senzorsku mrežu | 1 |

### 4.1. XBee ZigBee S2c

**XBee Zigbee S2C** je serija radio primopredajnih uređaja(RF modula) proizvođača *Digi International* koji su dizajnirani da funkcioniraju unutar domene ZigBee mrežnog standarda te da podrže sve njegove tehničke, ekonomske i funkcionalne aspekte. Funkcioniraju na *XBee ZB firmware*-u koji je utemeljen na "*EmberZNet 3.x ZigBee PRO Feaure Set*" mesh mrežnom stogu. Inačice *XBee ZB firmware-a* se mogu se instalirati na Xbee uređaje putem pripadajućeg *Digi XCTU* konfiguracijskog alata dostupnog za preuzimanje na službenoj web stranici proizvođača. Na slici **7.** prikazan je izgled Digi XBee S2c ZB RF modula.

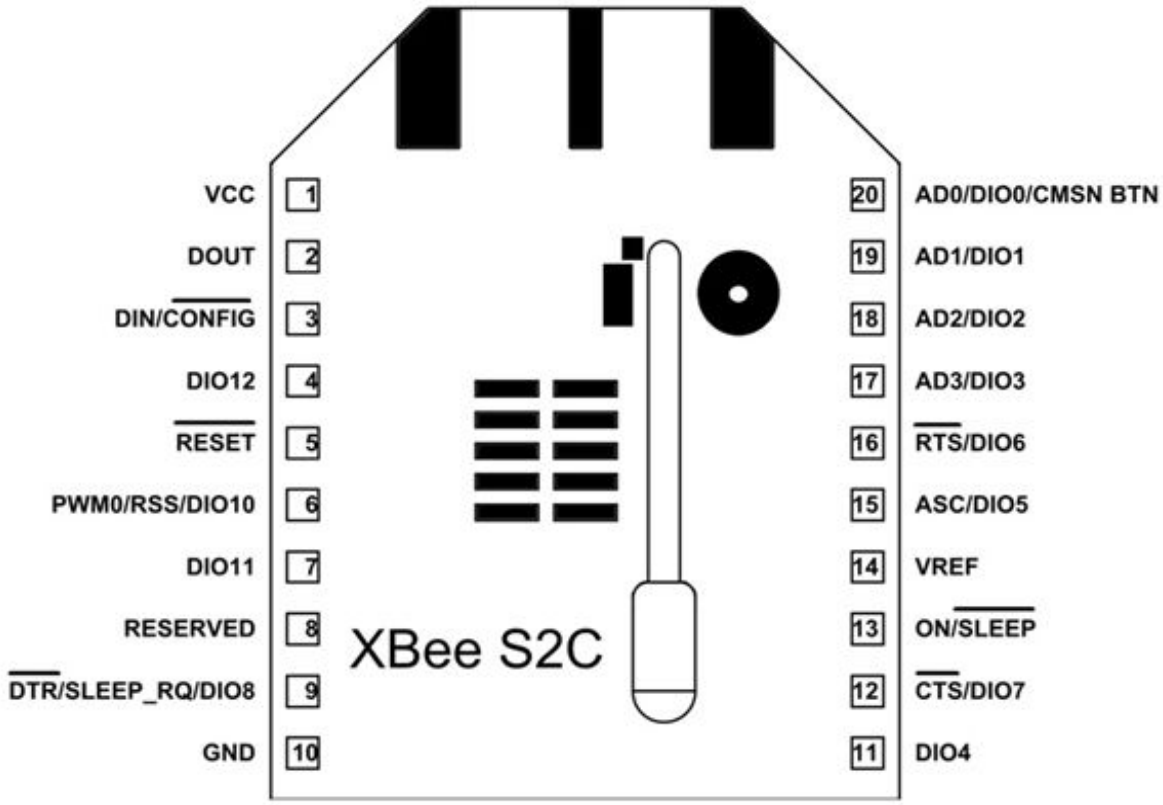
Slika 7. Prikaz izgleda XBee S2c radio modula

XBee RF moduli serije 1 i serije 2 imaju isti raspored terminala(*pin-out*). Međutim, moduli serije 1 ne mogu komunicirati s modulima serije 2 jer ne podržavaju iste mrežne protokole. Tehničke karakteristike XBee S2c RF modula su prikazane u tablici u nastavku.

Tablica 5. Tehničke karakteristike XBee S2c RF modula

| **TEHNIČKE KARAKTERISTIKE** | |
|---|---|
| **Općenito** | |
| Radni frekventni pojas | 2.4GHz |
| Broj kanala | 16 |
| Dimenzije | 2.438cm x 2.761cm |
| Radna temperatura | -40 to 85º C |
| **Performanse** | |
| Doseg u zatvorenom / urbanom okruženju | 40m |
| Doseg u otvorenom okruženju | 120m |
| Snaga predajnika(*software*-ski odabirljiva) | 2mW (+3dBm) |
| RF protok podataka | 250,000 bps |
| Protok podataka serijskog sučelja(*software*-ski odabirljiv) | 1200 - 230400 bps |
| Osjetljivost prijamnika | -95 dBm |
| **Zahtjevi napajanja** | |
| Napon napajanja | 2.8 – 3.4 V |
| Radna struja(pri predajnom režimu primopredajnika) | 40mA (@ 3.3 V) |
| Radna struja(pri primaćem režimu primopredajnika) | 40mA (@ 3.3 V) |
| Struja pri mirovanju | < 1 uA @ 25° C |

Na slici **8.** prikazani su nazivi i raspored *pin*-ova na XBee S2c radio modulu.

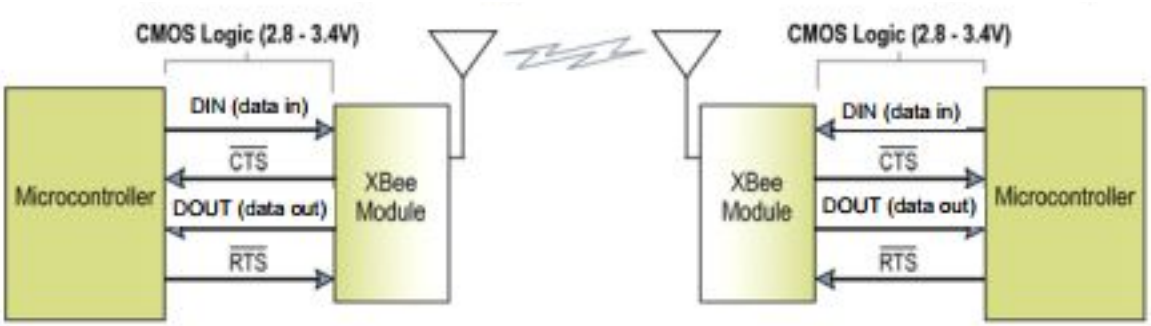



Slika 8. Nazivi i raspored *pin*-ova na XBee S2c radio modulima

Za funkcioniranje RF modula neophodna je upotreba VCC, GND, DOUT i DIN terminala(pinova), dok je za ažuriranje firmware-a serijskim putem potrebno koristiti i RTS te DTR terminale. XBee RF moduli nisu posebno osjetljivi na prisutnost obližnjih procesora, kristala ili ostalih PCB komponenti. Osim mehaničkih razmatranja, nema posebnog preporučenog položaja pri ugradnji RF modula u sklopovne segmente sustava.

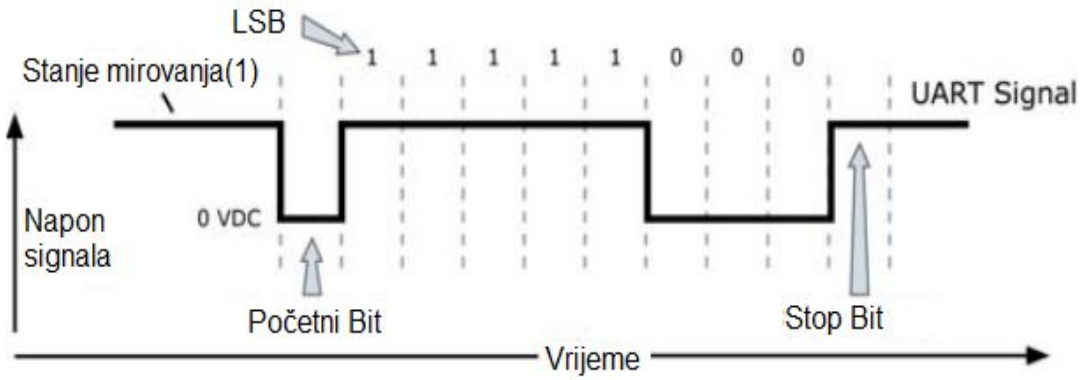
### 4.1.1. UART

Univerzalno asinkrono prijemnik/odašiljač sučelje(eng. UART-*Universal Asynchronous Receiver-Transmitter*) prima bajtove podataka te ih prosljeđuje u sekvencijalnom obliku toka bitova te na taj način omogućava pretvorbu između paralelnog i serijskog toka podataka u sustavu razmjene istih. Uređaji koji imaju UART sučelje mogu se izravno spojiti na terminale RF modula kao što je prikazano na slici ispod.




Slika 9. Dijagram protoka podataka kroz sustav u okruženju UART sučelja

Podaci ulaze u UART sučelje modula kroz DIN(pin 3) kao asinkroni serijski signal.Kada nema prijenosa podataka signal bi trebao biti u stanju mirovanja(1). Svaki podatkovni bajt se sastoji od početnog bita(0), 8 bitova podataka (LSB prvi) i stop bita(1). Slijedeća slika prikazuje serijski uzorak protoka bitova kroz modul.

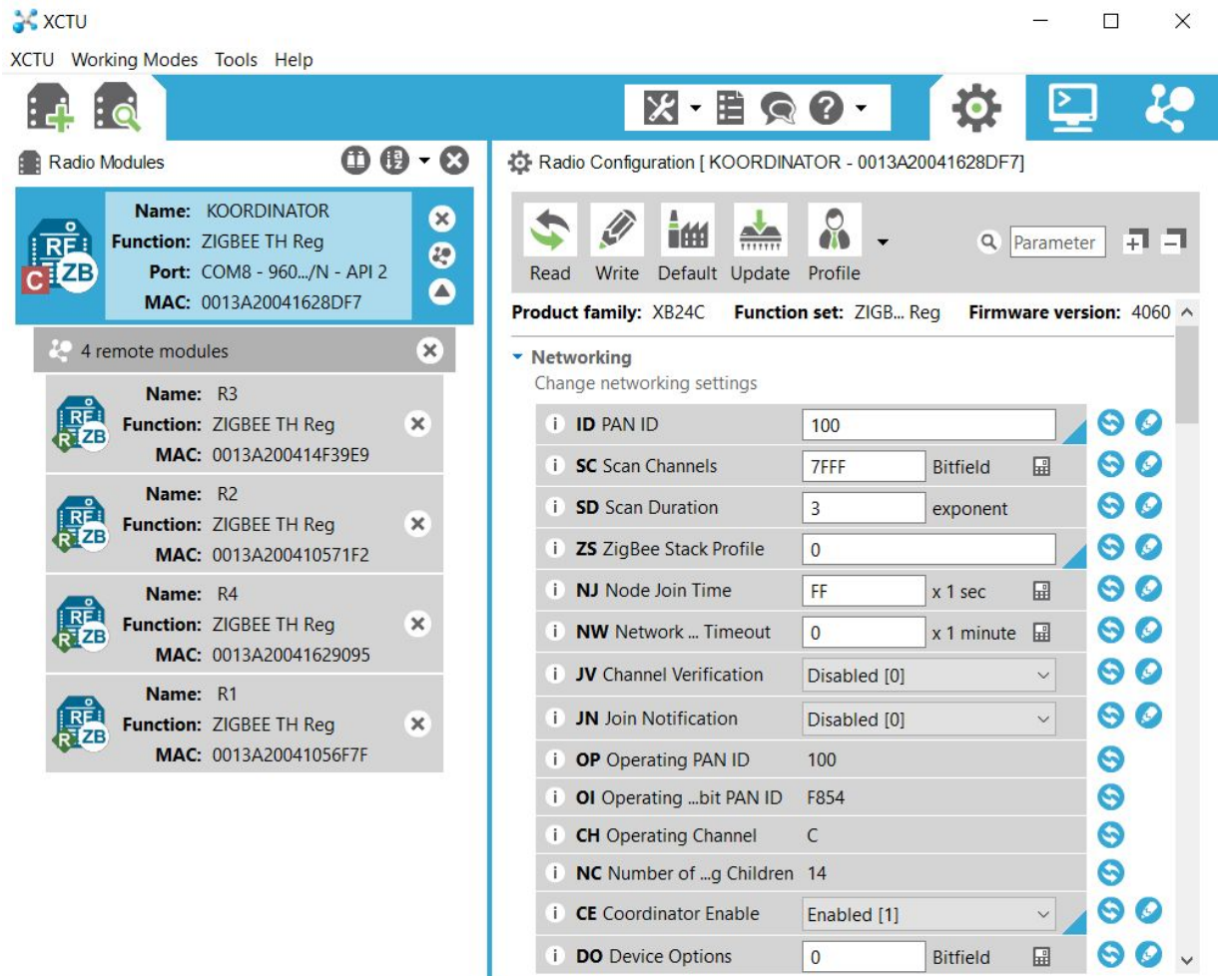



Slika 10. Primjer UART podatkovnog paketa: 0x1F (decimalni broj "31")

UART sučelje modula je zaduženo za provjeru vremena te provjeru pariteta, koji su potrebni za razmjenu podataka. Serijska komunikacija ovisi o dva UART-a koje je potrebno konfigurirati kompatibilnim postavkama (brzina prijenosa podataka, paritet, početni bitovi, zaustavni bitovi, podatkovni bitovi).

### 4.1.2. XCTU

Da bi više RF modula funkcioniralo unutar iste mreže, potrebno ih je konfigurirati interoperabilnim skupovima komunikacijskih postavki. Moduli su konfigurirani putem USB/UART sučelja izdavanjem *AT* naredbi. Najjednostavnija metoda konfiguracije RF modula izuzev korištenja terminala je upotreba *Digi XCTU* konfiguracijskog okruženja. Unutar navedenog okruženja moguće je ažurirati ili promijeniti tip *firmware*-a po kojemu RF modul funkcionira, definirati konfiguracijske postavke(PAN ID, opcije filtriranja kanala, razine izlazne snage, podešavanje perioda mirovanja modula, sigurnosne postavke mreže, brzine serijskog sučelja, itd,..), testirati komunikaciju u *AT* režimu rada, vizualizirati bežičnu mrežu te vratiti RF module na tvorničke postavke. Na slici **11.** prikazan je izgled *Digi XCTU* korisničkog sučelja.

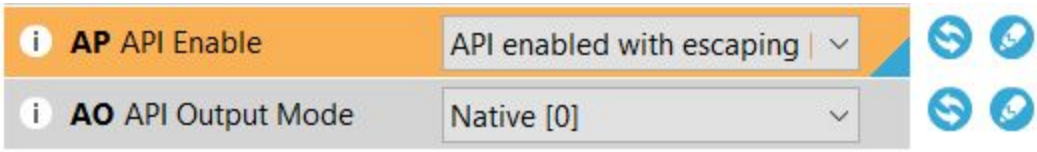



Slika 11. Prikaz izgleda XCTU korisničkog sučelja

### 4.1.3. API i AT režimi rada

RF moduli konfigurirani za funkcioniranje u *API* režimu rada koriste format koji paketizira podatke unutar unaprijed definiranih okvira što je izrazito pogodno za komunikaciju između više računala, ali samo po sebi nije dovoljno intuitivno za direktnu interakciju sa ljudima.

Pri konfiguraciji za funkcioniranje u *AT* režimu rada, isti funkcioniraju u tzv. "Transparentnom" režimu rada na način da se podaci koji dolaze na ulazni podatkovni terminal(DIN) izravno prosljeđuju preko mreže na željeni radio prijemnik bez mogućnosti ikakvih izmjena ili manipulacije u prijenosu istih. Takav jednostavan način prijenosa samo zamjenjuje upotrebu kabela pri serijskom komuniciranju te sadrži brojna ograničenja. Dolazni paketi mogu se direktno adresirati na jedno odredište(od točke do točke) ili emitirati na više odredišta(zvijezda). Ovaj se način prvenstveno koristi u slučajevima kada postojeći protokol ne može tolerirati promjene oblika podataka. Pri istoj *AT* konfiguraciji RF moduli mogu funkcionirati i u takozvanom *"Command"* režimu rada u kojem se na svaki RF modul lokalno izdaju određene konfiguracijske *AT* naredbe. *AT* naredbe se koriste za kontrolu postavki uređaja te se tretiraju kao lokalni konfiguracijski entiteti pojedinog modula a ne kao predmet razmjene podataka između više čvorova u mreži. Na slici u nastavku prikazan je odabir korištenog režima rada unutar *Digi XCTU* konfiguracijskog okruženja.




Slika 12. Prikaz odabira API režima rada unutar XCTU korisničkog sučelja

U tablici **6.** navedene su i opisane neke od najvažnijih **AT naredbe** te priroda i opseg njihovih parametara.

Tablica 6. Neke od najvažnijih AT naredbi

| **AT NAREDBA** | **NAZIV I OPIS** | **TIPOVI UREĐAJA** | **OPSEG** | **TVORNIČKE POSTAVKE** |
|---|---|---|---|---|
| ID | **PAN ID**, Postavlja zadani identifikator osobne mreže. | CRE | 0x0 - 0xFFFFFFFFFFFFFFFF | 0x0 |
| SC | **Scan Channels**, Postavlja listu kanala određenih za skeniranje pri formiranju PAN-a kao polja bitova. | CRE | 0x1 - 0xFFFF | 0xFFFF |
| ZS | **ZigBee Stack Profile,** Postavlja profil ZigBee stoga. | CRE | 0x0 - 0x2 | 0x0 |
| NI | **Node Identifier**, Postavlja identifikator pojedinog mrežnog čvora. | CRE | 0 - 20 ASCII znakova | ‘’ |
| DH | **Destination Address High**, Postavlja početna 32 bita 64 bitne adrese odredišta. | CRE | 0x0-0xFFFFFFFF | 0x0 |
| DL | **Destination Address Low**, Postavlja preostala 32 bita 64 bitne adrese odredišta. | CRE | 0x0-0xFFFFFFFF | 0x0 |

| AT NAREDBA | NAZIV I OPIS | TIPOVI UREĐAJA | OPSEG PARAMETRA | TVORNIČKE POSTAVKE |
|---|---|---|---|---|
| ND | **Node Discover**, Otkriva sve mrežne uređaje te pruža izvještaj sa osnovnim podacima svakog otkrivenog RF modula. | CRE | 20 bajtova "NI" ili" MY" vrijednosnih parametara | - |
| MY | **16-bit Network Address**, Pruža izvještaj o vrijednosti 16 bitne mrežne adrese RF modula. | CRE | 0x0 - 0xFFFE | 0xFFFE |
| SH | **Serial Number High,** Pruža izvještaj o vrijednosti početna 32 bita jedinstvene 64 bitne adrese modema. | CRE | 0x0 - 0xFFFFFFFF | [*Read Only*] |
| SL | **Serial Number High,** Pruža izvještaj o vrijednosti preostala 32 bita 64 bitne adrese modema. | CRE | 0x0- 0xFFFFFFFF | [*Read Only*] |
| AP | **API Enable**, Omogućava jedan od 2 moguća API režima rada. | CRE | 1-2 | 1 |

**Sučelje za programiranje aplikacija** (eng. API - *Application Programming Interface*) je skup standardiziranih sučelja koja omogućavaju međusobnu komunikaciju između više različitih aplikacija. Pri API režimu rada RF moduli razmjenjuju podatke u paketiziranom obliku obavijene unaprijed definiranim standardnim okvirima. Važno je napomenuti da su projektirani s namjerom da omoguće učinkovitu međusobnu komunikaciju između računala te da kao takvi nisu općenito dizajnirani za ***"front end"*** izravnu ljudsku interakciju. XBee API protokol definira okvire unutar kojih se podaci pakiraju u obliku sekvencijalnog toka bajtova. U tablici **7.** prikazana je osnovna struktura XBee API okvira.

Tablica 7. Osnovna struktura XBee API okvira

| **XBee API okvir** | | | | | | |
|---|---|---|---|---|---|---|
| OZNAKA POČETKA OKVIRA | DULJINA OKVIRA | | PODATKOVNI OKVIR | | | CHECKSUM |
| 1. Bajt | 2. Bajt | 3. Bajt | 4. Bajt | ... | *n*-ti Bajt | *n+1*-ti Bajt |
| 0x7E | MSB | LSB | Definirano tipom okvira | | | Suma svih bajtova okvira |

Početak svakog XBee API okvira označen je "startnim" bajtom koji osigurava razlučivanje te kategoričku interpretaciju primljenog toka podataka. Sljedeća dva bajta označavaju duljinu okvira te kao takvi pružaju neophodno saznanje o opsegu interpretacije pojedinog okvira. Unutar podatkovnog okvira prenose se bajtovi koji sadrže efektivno korisno opterećenje API okvira, a njihova značenja su specifična za pojedini tip API okvira. "*Checksum*" bajt je posljednji bajt API okvira a koristi se u svrhu otkrivanja i otklanjanja porgešaka pri prijenosu podataka. U tablici **7.1.** navedeni su različiti tipovi struktura XBee API okvira uz vrijednosti njihovih specifičnih identifikacijskih bajtova.

Tablica 7.1. Neke od najvažnijih AT naredbi

| VRIJEDNOST IDENTIFIKACIJSKOG BAJTA(HEX) | NAZIV OKVIRA |
|---|---|
| **0x08** | AT command (immediate) |
| **0x09** | AT command (queued) |
| **0x17** | Remote Command Request |
| **0x88** | AT command response |
| **0x8A** | Modem Status |
| **0x10** | TX request |
| **0x8B** | TX response |
| **0x90** | RX received |
| **0x92** | RX I/O data received |
| **0x95** | Node Identification Indicator |
| **0x97** | Remote Command Response |

Pri izradi ovoga rada u ostvarenu bežičnu senzorsku mrežu implementirani su: **0x08**, **0x88**, **0x10 0x8B** te **0x90** tipovi struktura XBee API okvira. Korišteni tipovi okvira tablično su opisani u nastavku.

Tablica 8. *AT command (immediate)* tip API okvira

| **AT command (immediate)** | | | | |
|---|---|---|---|---|
| **POLJE BAJTOVA API OKVIRA** | | **OFFSET** | **PRIMJER** | **OPIS** |
| Oznaka početka okvira | | 0 | 0x7E | Indikator početka okvira. |
| Duljina okvira | | MSB1 | 0x00 | Broj bajtova između duljine okvira i checksum-a. |
| | | LSB2 | 0x04 | |
| Podatkovni okvir | Tip okvira | 3 | 0x08 | Tip strukture API okvira. |
| | ID okvira | 4 | 0x01 | Identifikator tipa i strukture podatkovnog okvira. |
| | AT naredba | 5 | 0x4E(N) | Naziv AT naredbe. |
| | | 6 | 0x44(D) | |
| | Vrijednost parametra | Neobavezno | | Ako je polje zadano tretira se kao zahtjev za izmjenu navedenog parametra, u suprotnom tretira se kao zahtjev za izvještaj o vrijednosti istoga. |
| Checksum | | 7 | 0x64 | 0xFF-Suma bajtova od ”*offset*=3” bajta. |

Tablica 9. *AT command response* tip API okvira

| **AT command response** | | | | |
|---|---|---|---|---|
| **POLJE BAJTOVA API OKVIRA** | | **OFFSET** | **PRIMJER** | **OPIS** |
| Oznaka početka okvira | | 0 | 0x7E | Indikator početka okvira. |
| Duljina okvira | | MSB1 | 0x00 | Broj bajtova između duljine okvira i checksum-a. |
| | | LSB2 | 0x1A | |
| Podatkovni okvir | Tip okvira | 3 | 0x88 | Tip strukture API okvira. |
| | ID okvira | 4 | 0x01 | Identifikator tipa i strukture podatkovnog okvira. |
| | AT naredba | 5 | 0x4E(N) | Naziv AT naredbe. |
| | | 6 | 0x44(D) | |
| | Status AT naredbe | 7 | 0x00 | 0 = OK<br>1 = ERROR<br>2 = Invalid Command<br>3 = Invalid Parameter<br>4 = Tx Failure |
| | Izvještaj o zahtjevanoj naredbi u binarnom formatu | | | Ako je polje prethodno zadano tretira se kao odgovor na zahtjev za izvještaj o vrijednosti istoga. |
| Checksum | | 8 | 0xAD | 0xFF-Suma bajtova od ”*offset*=3” bajta. |

Tablica 10. *TX request* tip API okvira

| **TX request** | | | | |
|---|---|---|---|---|
| **POLJE BAJTOVA API OKVIRA** | | **OFFSET** | **PRIMJER** | **OPIS** |
| Oznaka početka okvira | | 0 | 0x7E | Indikator početka okvira. |
| Duljina okvira | | MSB1 | 0x00 | Broj bajtova između duljine okvira i checksum-a. |
| | | LSB2 | 0x16 | |
| Podatkovni okvir | Tip okvira | 3 | 0x10 | Tip strukture API okvira. |
| | ID okvira | 4 | 0x01 | Identifikator tipa i strukture podatkovnog okvira. |
| | 64 bit-na adresa odredišnog čvora | MSB5 | 0x00 | -64 bit-na adresa odredišnog čvora.<br><br>-Adresa rezervirana za mrežnog koordinatora: 0x0000000000000000<br><br>-Broadcast adresa: 0x000000000000FFFF |
| | | 6 | 0x13 | |
| | | 7 | 0xA2 | |
| | | 8 | 0x00 | |
| | | 9 | 0x40 | |
| | | 10 | 0x0A | |
| | | 11 | 0x01 | |
| | | LSB12 | 0x27 | |
| | 16 bit-na adresa odredišnog čvora | MSB13 | 0xFF | Ako je vrijednost iste nepoznata ili broadcast polje se treba postaviti u 0xFFFE |
| | | LSB14 | 0xFE | |

| POLJE BAJTOVA API OKVIRA | | OFFSET | PRIMJER | OPIS |
|---|---|---|---|---|
| Podatkovni okvir | Broadcast radijus | 15 | 0x00 | Postavlja najveći mogući broj mrežnih skokova pri prijenosu paketa, vrijednost 0x00 postavlja najveći mogući broj . |
| | Opcije | 16 | 0x00 | Dodatne opcije<br>0x20 – omogući APS enkripciju (ako je EE=1) |
| | RF data | 17 | 0x54 | Korisno opterećenja okvira |
| | | 18 | 0x78 | |
| | | 19 | 0x44 | |
| | | 20 | 0x61 | |
| | | 21 | 0x74 | |
| | | 22 | 0x61 | |
| | | 23 | 0x30 | |
| | | 24 | 0x41 | |
| Checksum | | 25 | 0x13 | 0xFF-Suma bajtova od ”*offset*=3” bajta. |

Tablica 11. *TX response* tip API okvira

| **TX response** | | | | |
|---|---|---|---|---|
| **POLJE BAJTOVA API OKVIRA** | | **OFFSET** | **PRIMJER** | **OPIS** |
| Oznaka početka okvira | | 0 | 0x7E | Indikator početka okvira. |
| Duljina okvira | | MSB1 | 0x00 | Broj bajtova između duljine okvira i checksum-a. |
| | | LSB2 | 0x07 | |
| Podatkovni okvir | Tip okvira | 3 | 0x8B | Tip strukture API okvira. |
| | ID okvira | 4 | 0x01 | Identifikator tipa i strukture podatkovnog okvira. |
| | 16 bit-na adresa odredišnog čvora | 5 | 0x7D | 16 bit-na adresa odredišnog čvora. |
| | | 6 | 0x84 | |
| | Brojač pokušaja prijenosa podataka | 7 | 0x00 | Broj pokušaja prijenosa. |
| | Status uspješnosti dostave paketa | 8 | 0x00 | 0x00 = Success |
| | Status otkrivanja | 9 | 0x01 | 0x01 = Address discovery |
| Checksum | | 10 | 0x71 | Suma bajtova od ”*offset*=3” bajta. |

Tablica 12. RX received tip API okvira

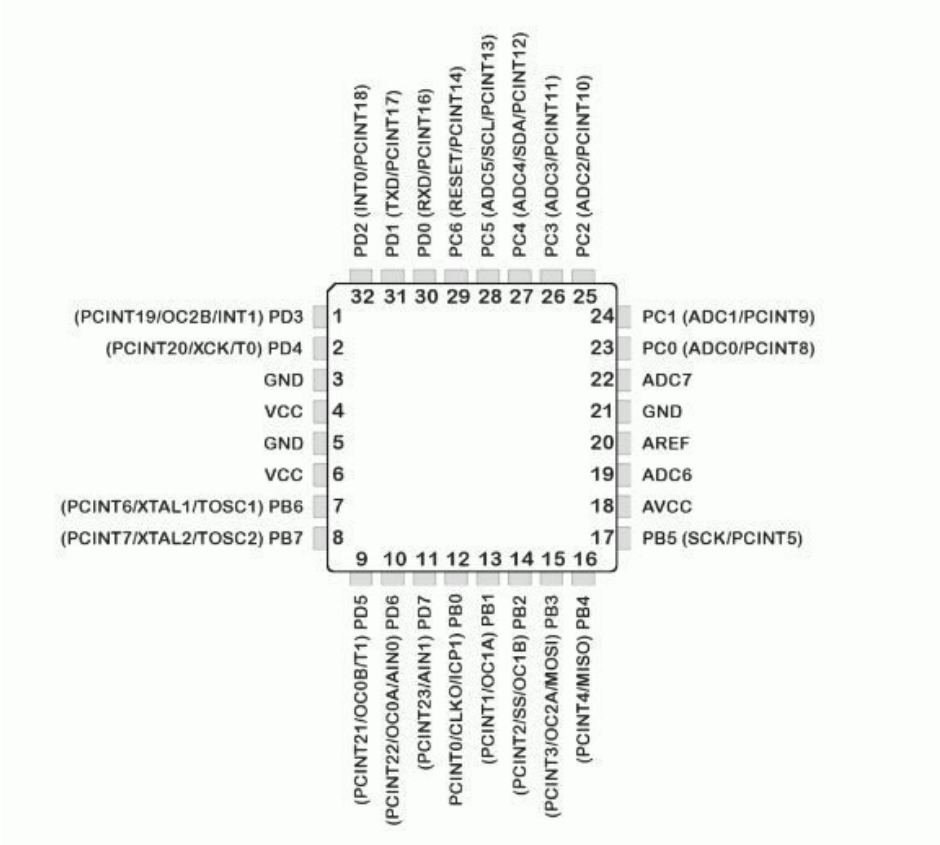
| RX received | | | | |
|---|---|---|---|---|
| **POLJE BAJTOVA API OKVIRA** | | **OFFSET** | **PRIMJER** | **OPIS** |
| Oznaka početka okvira | | 0 | 0x7E | Indikator početka okvira. |
| Duljina okvira | | MSB1 | 0x00 | Broj bajtova između duljine okvira i checksum-a. |
| | | LSB2 | 0x11 | |
| Podatkovni okvir | Tip okvira | 3 | 0x90 | Tip strukture API okvira. |
| | 64 bit-na adresa izvorišnog mrežnog čvora | MSB4 | 0x00 | -64 bit-na adresa čvora pošiljatelja. |
| | | 5 | 0x13 | |
| | | 6 | 0xA2 | |
| | | 7 | 0x00 | |
| | | 8 | 0x40 | |
| | | 9 | 0x52 | |
| | | 10 | 0x2B | |
| | | LSB11 | 0xAA | |
| | 16 bit-na adresa pošiljatelja | MSB12 | 0x7D | -64 bit-na adresa čvora pošiljatelja. |
| | | LSB13 | 0x84 | |
| | Opcije | 14 | 0x01 | 0x01 – Packet acknowledged. |
| | Zaprimljeni podatci | 15 | 0x78 | Korisno opterećenja okvira. |
| | | 16 | 0x52 | |
| Checksum | | 17 | 0x0B | Suma bajtova od ”*offset*=3” bajta. |

### 4.2. Arduino platforma

Arduino je *open–source* elektronička platforma temeljena na Atmelovim 8–, 16– ili 32–bitnim AVR mikrokontrolerima (*ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560*). Projekt Arduino platforme započet je u Italiji 2005. godine kao derivacija projekta "*Wiring"* s instituta "*Interaction Design Institute Ivrea*". Tim koji je iznjedrio projekt sastojao se od sljedećih suradnika: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino i David Mellis.

Programiranje Arduino mikrokontrolera vrši se pomoću besplatnog razvojnog okruženja dostupnog za preuzimanje na službenoj Arduino web stranici. Programiranje mikrokontrolera pomoću spomenutog razvojnog okruženja putem serijske veze je moguće ako je na istome predinstaliran *boot loader*. Ako korisnik u slučaju nedostatka programske memorije odluči obrisati *boot loader* kako bi oslobodio dio iste za potrebe svog programa, programiranje mikrokontrolera morati će izvršiti putem nekog od sljedećih unutar sustavnih programera: AVR ISP, Arduino as ISP, USBtinyISP, AVRISP mkII ili putem USBasp modula.

Za potrebe ovog projekta kao centralna upravljačka jedinica svakog pojedinog mrežnog čvora odabrana je Arudino NANO platforma, koja je bazirana na 8–bitnom *AVR ATmega328p* mikrokontroleru. Arduino mikrokontroler raspolaže sa: 22 digitalnih izlazno/ulaznih pinova, 6 analognih ulaza, 16MHz–nim kvarcnim kristalom, USB mini portom, ICSP zaglavljem te dugmetom za reset. Pri pisanju pripadajućeg upravljačkog koda pojedinog mikrokontrolera u mreži korištene su biblioteke koje sadrže potrebne setove funkcija i metode za software-sku implementaciju Xbee S2C radio modula kao i korištenih senzorskih modula u domeni Arduino razvojnog okruženja. Na slici **13.** prikazani su nazivi i raspored *pin*-ova na *ATmega328*p mikrontroleru dok je na slici **14.** prikazan izgled Arduino NANO platforme.




Slika 13. Nazivi i raspored pinova na “ATmega328p”

Slika 14. Prikaz izgleda Arduino NANO platforme

U tablici **13.** prikazane su tehničke karakteristike Arduino NANO platforme.

Tablica 13. Tehničke karakteristike “Arduino NANO”

| | |
|---|---|
| **Mikrokontroler** | ATmega328p |
| **Arhitektura** | AVR |
| **Radni napon** | 5V |
| **Ulazni napon (preporučeno)** | 7–12V |
| **Digitalni I/O pinovi** | 22 ( 6 podržavaju PWM ) |
| **Analogni ulazni pinovi** | 8 |
| **DC struja po I/O pinu** | 40 mA |
| **Struja potrošnje** | 19 mA |
| **Flash memorija** | 32 KB(od čega 2 KB zauzima bootloader) |
| **SRAM** | 2 KB |
| **EEPROM** | 1 KB |
| **Radni takt** | 16 MHz |
| **Dimenzije PCB-a** | 18 x 45 mm |

### 4.3. BME280

BME280 je digitalni elektronički sklop za mjerenje temperature, tlaka i relativne vlažnosti zraka. Ovaj precizni osjetnik može mjeriti relativnu vlažnost od 0 do 100% s točnošću od ± 3%, barometarski tlak od 300 Pa do 1100 hPa s apsolutnom točnosti ± 1 hPa, i temperaturu od -40 ° C do 85 ° C s točnošću ± 1,0 ° C. Mjerenja tlaka su dovoljna precizna da se može koristiti i kao visinomjer s točnošću od ± 1 metar. Na modulu je ugrađen LM6206 3.3V regulator i I2C prevoditelj naponske razine, tako da se može koristiti s 3.3V ili 5V logičkim mikrokontrolerima poput ovdje implementiranog Arduina.

Tehničke karakteristike osjetnika prikazane su u tablici **13. 1.**

Tablica 13.1. Tehničke karakteristike "BME280"

| | |
|---|---|
| **Model** | BME280 |
| **Napon napajanja** | 3.3/5 V |
| **Struja potrošnje** | <2.5 mA u aktivnom stanju te oko 5μA u stanju mirovanja |
| **Mjerenje vlažnosti** | Od 0 do100% relativne vlažnosti |
| **Odstupanja mjerenja vlažnosti** | ±3% |
| **Temperaturni domet** | -40 - 85°C |
| **Odstupanja mjerenja temperature** | ±1°C |
| **Mjerni opseg za mjerenje tlaka** | 300hPa - 1100hPa |

Izgled BME280 senzorskog modula prikazan je na slici **15.**

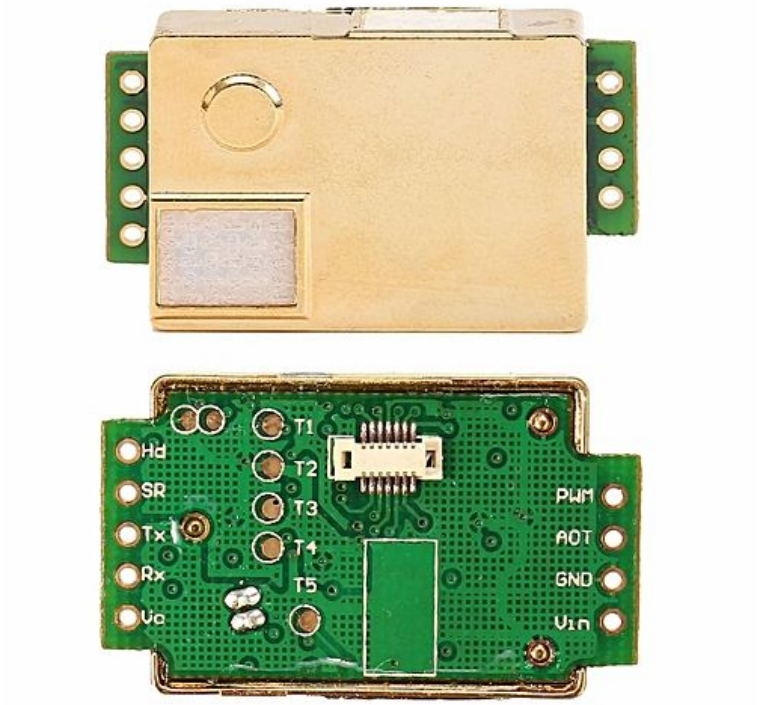
Slika 16. Prikaz izgleda “BME280”

BME280 senzorski modul ima jednostavno dvožično *I2C* sučelje koje se može lako povezati s većinom dostupnih mikrokontrolera. Zadana *I2C* adresa BME280 modula je 0x76 i može se lako promijeniti u 0x77 na način da se oštrim predmetom zagrebe površina između bakrene površine koja se nalazi u sredini u skupini od 3 iste i one sa kojom je spojena te da se nakon toga poveže sa preostalom pomoću kratkospojnika. Moguće konfiguracije *I2C* adresa prikazane su na slici **17.**

Slika 17. Prikaz mogućih I2C adresnih konfiguracija putem kratkospojnika

### 4.4. MH-Z19

MH-Z19 je spektroskopski osjetnik čija je svrha mjerenje koncentracije CO2 plina u zraku, a funkcioniranje mu se temelji se na principu nedisperzivnog infracrvenog signala (NDIR). Stožerne komponente NDIR osjetnika su: izvor infracrvene svjetlosti, komora za uzorke ili svjetlosna cijev, svjetlosni filter i infracrveni detektor. Na slici **18.** prikazan je izgled ”MH-Z19” osjetnika.




Slika 18. Prikaz izgleda ”MH-Z19” osjetnika

Tablica **14.** prikazuje tehničke karakteristike “MH-Z19” osjetnika.

Tablica 14. Tehničke karakteristike “MH-Z19”

| | |
|---|---|
| **Model** | MH-Z19 |
| **Napon napajanja** | 3.6 ~ 5.5 V DC |
| **Struja potrošnje** | < 18 mA |
| **Mjerni opseg** | 0〜5000 ppm |
| **Odstupanja mjerenja koncentracije CO2** | ±50 ppm |
| **Vrste izlaznih signala** | UART |
| | PWM |
| **Vrijeme zagrijavanja** | 3 min |

### 4.4.1. Princip rada

Princip rada osjetnika zasniva se na tome da je infracrvena svjetlost usmjerena od njenog izvora prema detektoru kroz komoru u kojoj se nalazi uzorak ambijentalnog zraka nad kojim vršimo mjerenje konceratracije C02. Paralelno postoji još jedna komora s detektorom u kojoj se nalazi referentni plin, obično dušik. Zrak u komori za uzorke uzrokuje apsorpciju specifičnih valnih duljina prema *Beer-Lambertovom* zakonu, a prigušivanje istih u odnosu na detektirani snop svjetlosti koji je prošao kroz komoru sa referentnim plinom mjeri se detektorom za određivanje koncentracije plina. Optički filter ispred detektora uklanja svu svjetlost osim valne duljine koju odabrane molekule plina mogu apsorbirati. *Beer-Lambertov* zakon definiran je sljedećom formulom:

$$A = \log(Po/P) = \varepsilon bc \qquad (1)$$

gdje je:

A - apsorbancija na danoj valnoj duljini svjetlosti

ε - molarni apsorpcijski (ekstinkcijski) koeficijent (L mol-1cm-1)

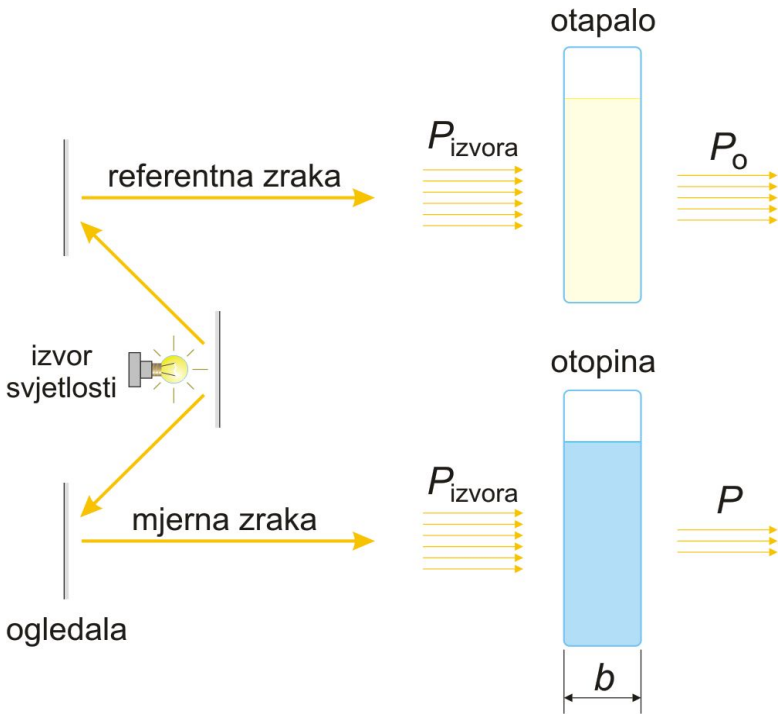
b - duljina puta svjetlosti kroz uzorak (cm)

c - koncentracija tvari u otopini (mol L-1).

Po - intezitet detektiranog snopa svjetlosti kroz ambijentalni zrak

P - intezitet detektiranog snopa svjetlosti kroz referentni plin

Na slici **19.** prikazan je primjer *Beer-Lambertova* zakona pri mjerenju inteziteta detektirane svjetlosti kroz otopinu u odnosu na referentno otapalo.




Slika 19. Prikaz primjera *Beer-Lambertova* zakona

### 4.4.2. UART komunikacija

U tablici **15.** prikazane su postavke za UART komunikaciju sa osjetnikom.

Tablica 15. UART postavke

| | |
|---|---|
| **Baud rate** | 9600 bps |
| **Broj bajtova podataka** | 8 |
| **Broj "STOP" bajtova** | 1 |
| **Paritet** | Nije podržano |

Komunikacija sa senzorom putem UART sučelja zasniva se na slanju i primanju okvira od devet bajtova, a početni bajt uvijek sadržava vrijednost "0xFF". Tablica **16.** prikazuje format okvira naredbe *"Gas concentration"* koja šalje upit senzoru za očitanom vrijednosti mjerene veličine.

Tablica 16. Prikaz okvira naredbe

| ***"Gas concentration"* - naredba** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bajt0 | bajt1 | bajt2 | bajt3 | bajt4 | bajt5 | bajt6 | bajt7 | bajt8 |
| Oznaka početka okvira | ID senzora | Naredba | - | - | - | - | - | checksum |
| 0xFF | 0x01 | 0x86 | 0x | 0x | 0x | 0x | 0x | 0x79 |

Tablica **17.** prikazuje format okvira odgovora na naredbu *"Gas concentration"*.

Tablica 17. Prikaz okvira odgovora

| ***"Gas concentration"* - odgovor** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| bajt0 | bajt1 | bajt2 | bajt3 | bajt4 | bajt5 | bajt6 | bajt7 | bajt8 |
| Oznaka početka okvira | Naredba | Razina koncentracije (visoka) | Razina koncentracije (niska) | - | - | - | - | checksum |

Koncentracija C02 se iz vrijednosti bajtova 2 i 3 izračuna na način opisan sljedećom formulom:

$$\text{Gas concentration} = \text{bajt2} * 256 + \text{bajt3} \quad (2)$$

gdje je:

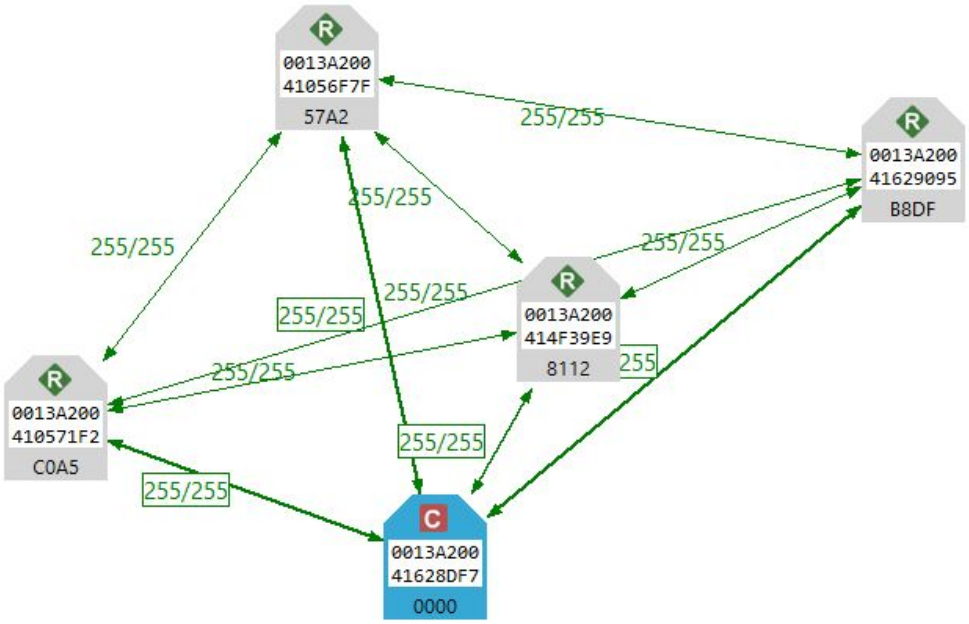
Gas concentration - koncentracija C02 (ppm)

bajt2 - razina koncentracije(visoka)

bajt3 - razina koncentracije(niska)

# 5. ARHITEKTURA MREŽE

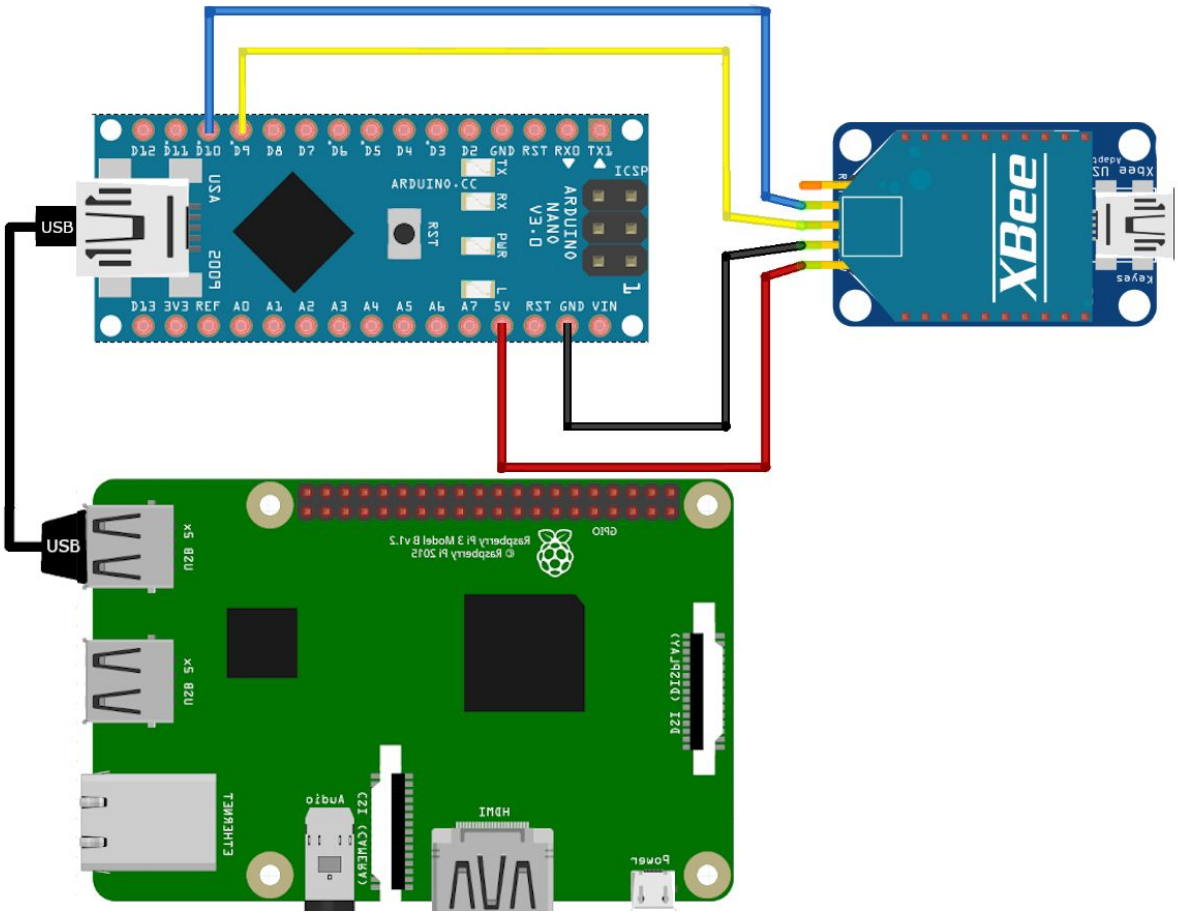
Bežična senzorska mreža realizirana u ovom radu sastoji se od pet mrežnih čvorova. Mreža s obzirom na funkcijsku podjelu čvorova sadrži četiri usmjerivača u ulogama mjernih čvorova te jednog pripadajućeg koordinatora. Svaki usmjerivač je lokalno fizički spojen na po jedan Arduino mikrokontroler te putem njega i na MH-Z19 i BME280 osjetnike. Mrežni koordinator je lokalno fizički spojen na jedan Arduino mikrokontroler koji je povezan te periodički, na zahtjev, prima očitanja sa senzora udaljenih mjeriteljskih mrežnih čvorova. Arduino mikrokontroler na koordinatorskom čvoru povezan je USB kabelom na Raspberry Pi putem kojeg se očitani podatci pohranjuju u *.CSV* tekstualnu datoteku. Mreža je strukturirana unutar *mesh* topologijskog koncepta te nema ustaljenu prostornu infrastrukturu. Također, postignuta je određena kapacitetna autonomija sustava u vidu omogućavanja skalabilnosti mreže putem algoritma ostvarenog unutar Arduino mikrokontrolera spojenog na koordinatorski mrežni čvor. Neosjetljivost mreže na određenu razinu dinamike broja prisutnih uređaja na istoj postignuta je periodičkim sakupljanjem adresa uređaja na mreži od koordinatora. Na slici **20.** prikazana je topologija realizirane mreže.




Slika 20. Prikaz topologije ostvarene bežične senzorske mreže

### 5.1. Arhitektura koordinatorskog mrežnog čvora

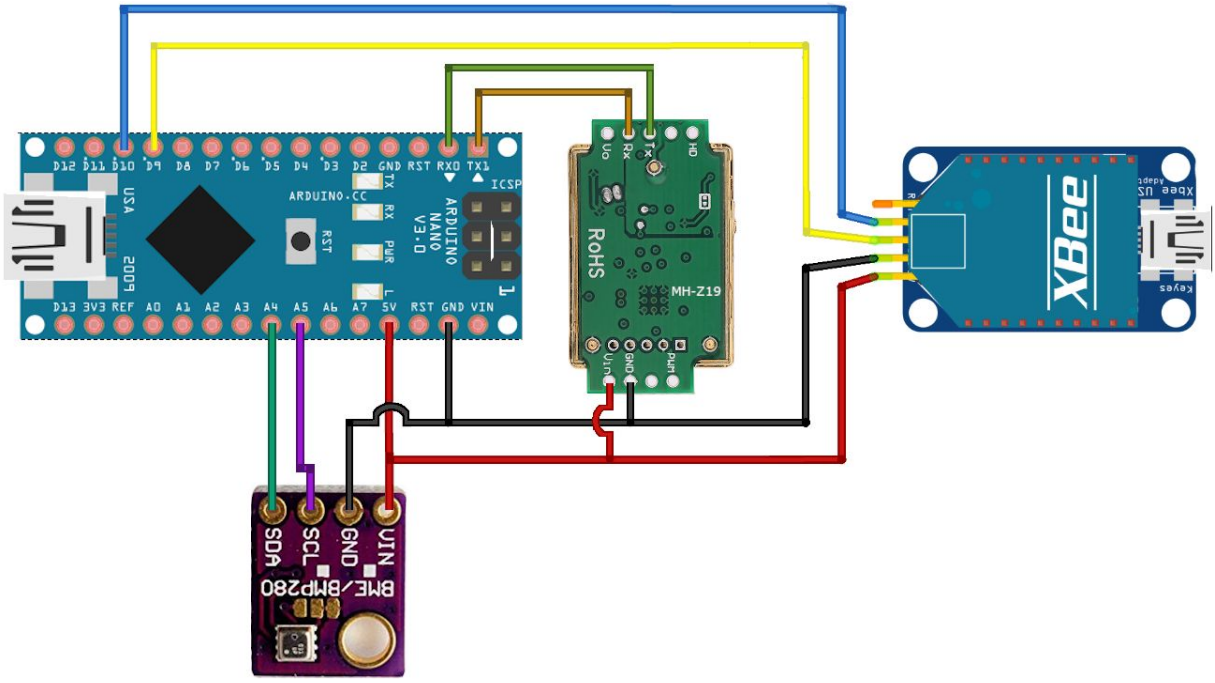
Na slici **21.** prikazana je arhitektura koordinatorskog mrežnog čvora koji se sastoji od Arduino NANO mikrokontrolera, XBee S2C radio modula, pomoćnog modula za povezivanje navedenih te Raspberry Pi 3B+ minijaturnog računala. Funkcijski se čvor može podijeliti u 3 sučelja. Pomoćni modul predstavlja sučelje između mikrokontrolera i radio modula dok radio modul predstavlja sučelje između samog čvora i prijenosnog komunikacijskog medija, a USB sučelje između Arduino mikrokontrolera i Raspberry Pi računala.




Slika 21. Prikaz izgleda arhitekture i pokazne sheme spajanja koordinatorskog mrežnog čvora

### 5.2. Arhitektura mjeriteljskog mrežnog čvora

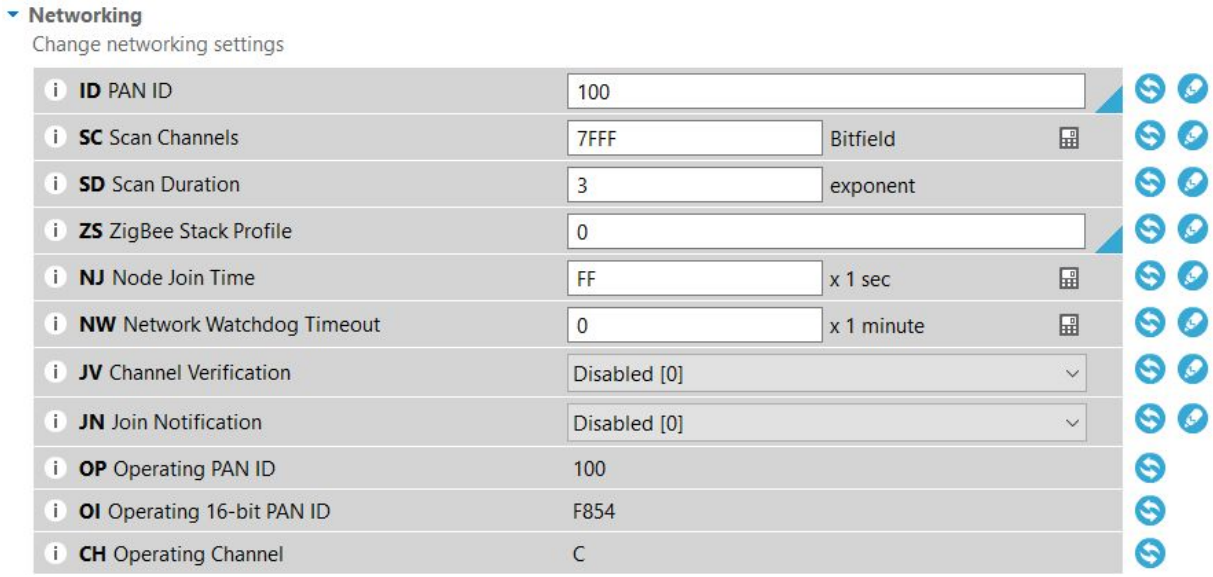
Na slici **22.** prikazana je arhitektura mjeriteljskog mrežnog čvora koji se sastoji od Arduino UNO R3 mikrokontrolera, XBee S2C radio modula, pomoćnog modula za povezivanje navedenih te MH-Z19 i BME280 senzorskih modula. Funkcijski se čvor može podijeliti u 3 sučelja. Pomoćni modul predstavlja sučelje između mikrokontrolera i radio modula dok radio modul predstavlja sučelje između samog čvora i prijenosnog komunikacijskog medija ,a analogni ulazi na mikrokontroleru sučelje između senzora i samog mikrokontrolera.




Slika 22. Prikaz izgleda arhitekture i pokazne sheme spajanja mjeriteljskog mrežnog čvora

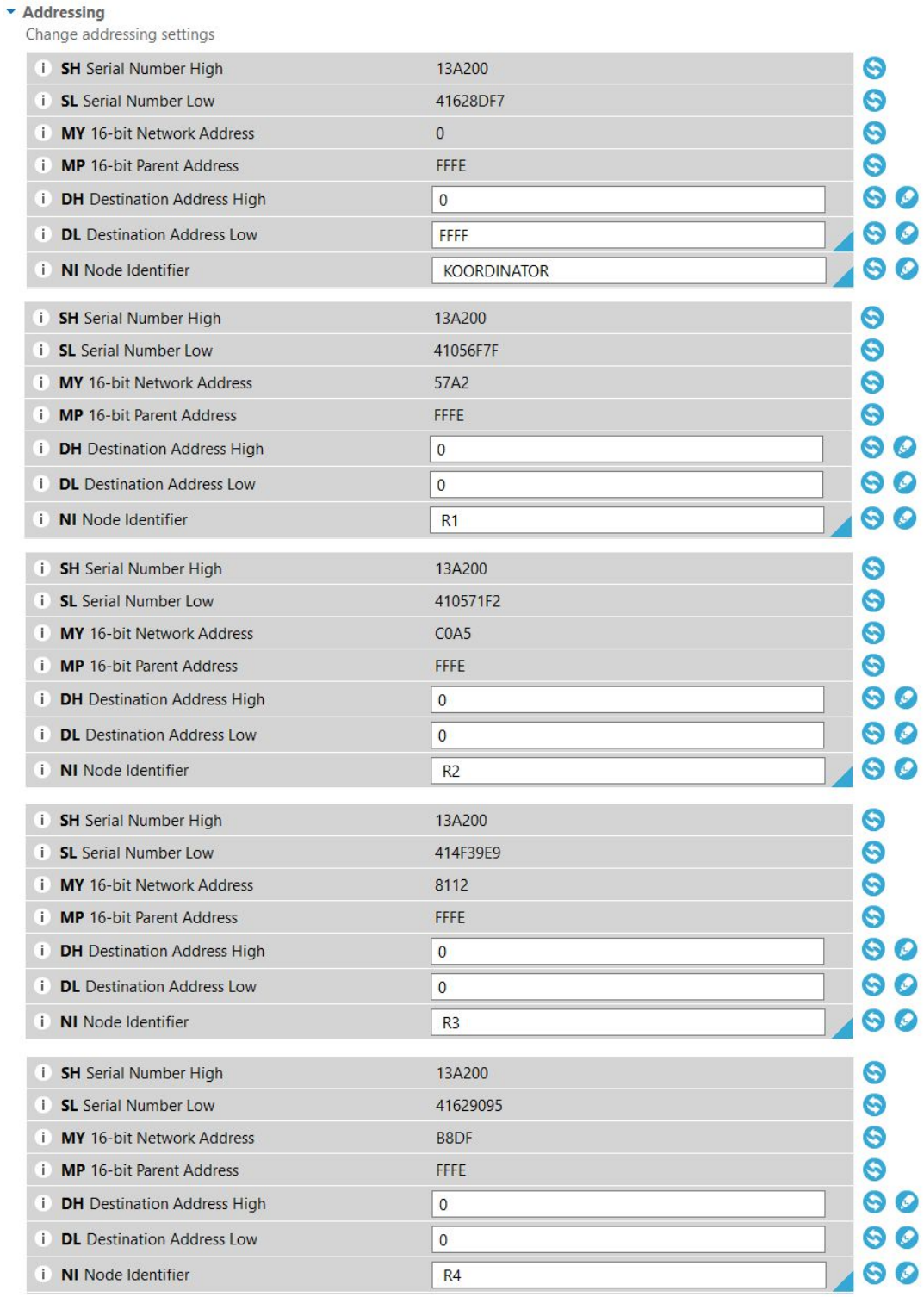
# 6. IMPLEMENTACIJA

Kako bi RF moduli u mreži međusobno komunicirali isti su konfigurirani kompatibilnim, odnosno interoperabilnim, postavkama unutar XCTU konfiguracijskog okruženja. Na slici u nastavku prikazane su zajedničke mrežne postavke za sve RF module u mreži unutar *Digi XCTU* konfiguracijskog okruženja.




Slika 23. Zajedničke konfiguracijske mrežne postavke svih RF modula u mreži

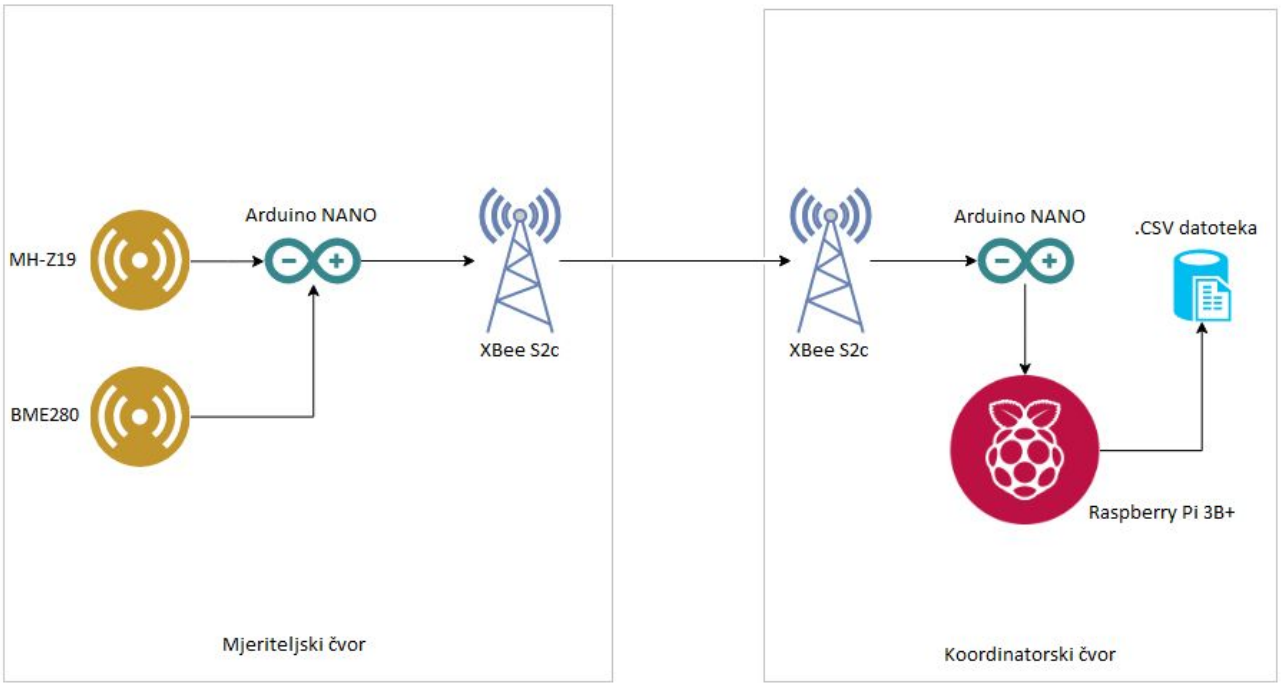
Na slici u nastavku prikazane su pojedinačne adresne postavke svih radio modula u mreži unutar *Digi XCTU* konfiguracijskog okruženja.




Slika 24. Pojedinačne adresne  konfiguracijske postavke svih RF modula u mreži

### 6.1. Algoritam ciklusa rada

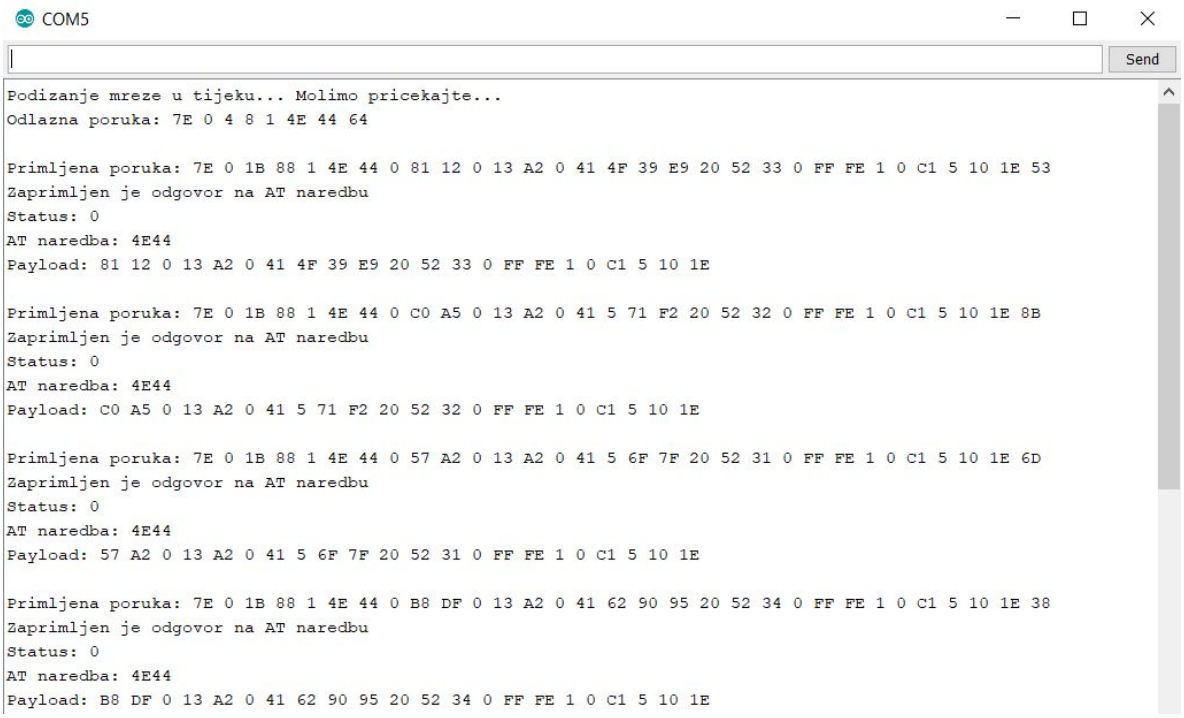
Na sljedećoj slici prikazan je tok podataka koji čine vrijednosti očitane na senzorima te sučelja kroz koja prolaze.




Slika 25. Protok podataka kroz različita sučelja u sustavu

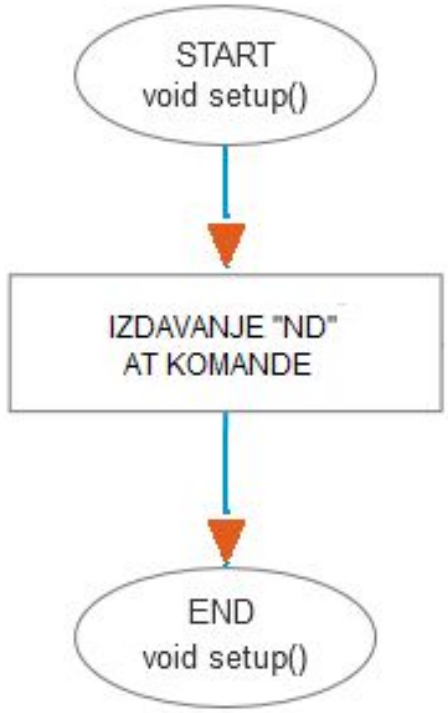
Postupak očitanja podataka senzora sa udaljenih mjeriteljskih mrežnih čvorova na koordinatorskom mrežnom čvoru odvija se u tri sekvencijalne faze. Pri inicijalnoj fazi mrežni koordinator na podražaj *AT* naredbe prikuplja adrese svih prisutnih uređaja na mreži u datom trenutku.

Prikupljene adrese privremeno se spremaju u adresno dvodimenzionalno polje te se prebrisuju novima uslijed slijedećeg periodičnog zahtjeva za prikupljanje mrežnih adresa u vidu izdavanja "*ND*" *AT* naredbe na koordinatorski RF modul. Između pojedinih prekidnih mehanizama koji iniciraju osvježavanje adresne matrice protekne vrijeme u trajanju od tri minute. Unutar tog vremenskog perioda koordinator šalje pojedinačne zahtjeve za primanje podataka očitanja senzora na svaki mjeriteljski čvor te ponavlja navedeni postupak u vremenskim razmacima od jedne minute što rezultira uzorkovanjem mjerenih veličina frekvencijom od 0.02 Hz. Ciklus rada mjeriteljskog čvora uvjetovan je i odgovara zahtjevima koordinatorskog čvora. Na slici ispod prikazan je ispis iniciranja mreže te primitka adresnih podataka mjeriteljskih mrežnih čvorova pri popunjavanju adresne matrice.



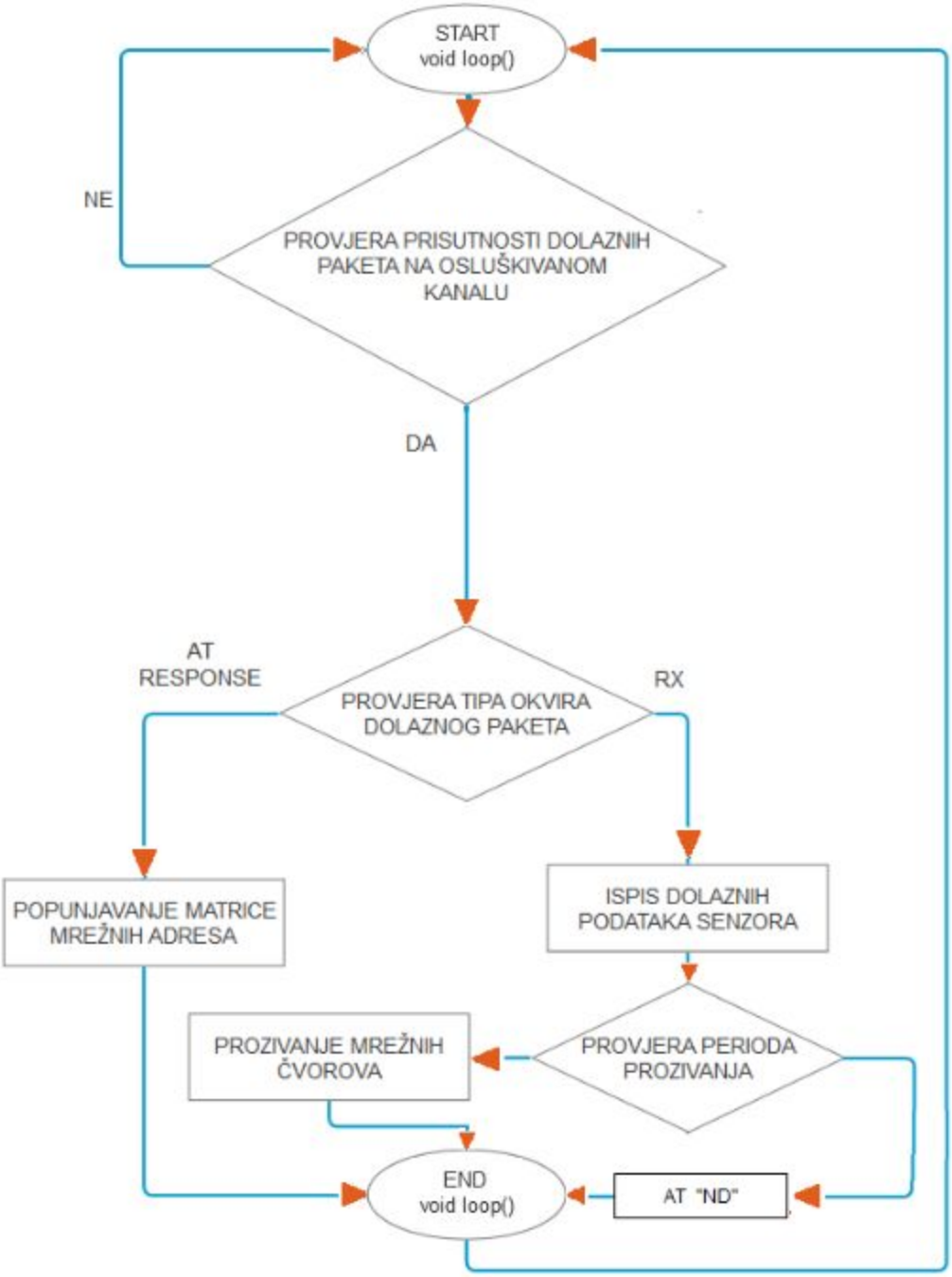

Slika 26. Ispis inicijalizacije mreže sa koordinatorskog čvora

Na slici ispod prikazan je funkcijski blok dijagram *void setup()* funkcije koju Arduino mikrokontroler koordinatorskog mrežnog čvora izvršava jedan put pri početku rada uređaja.



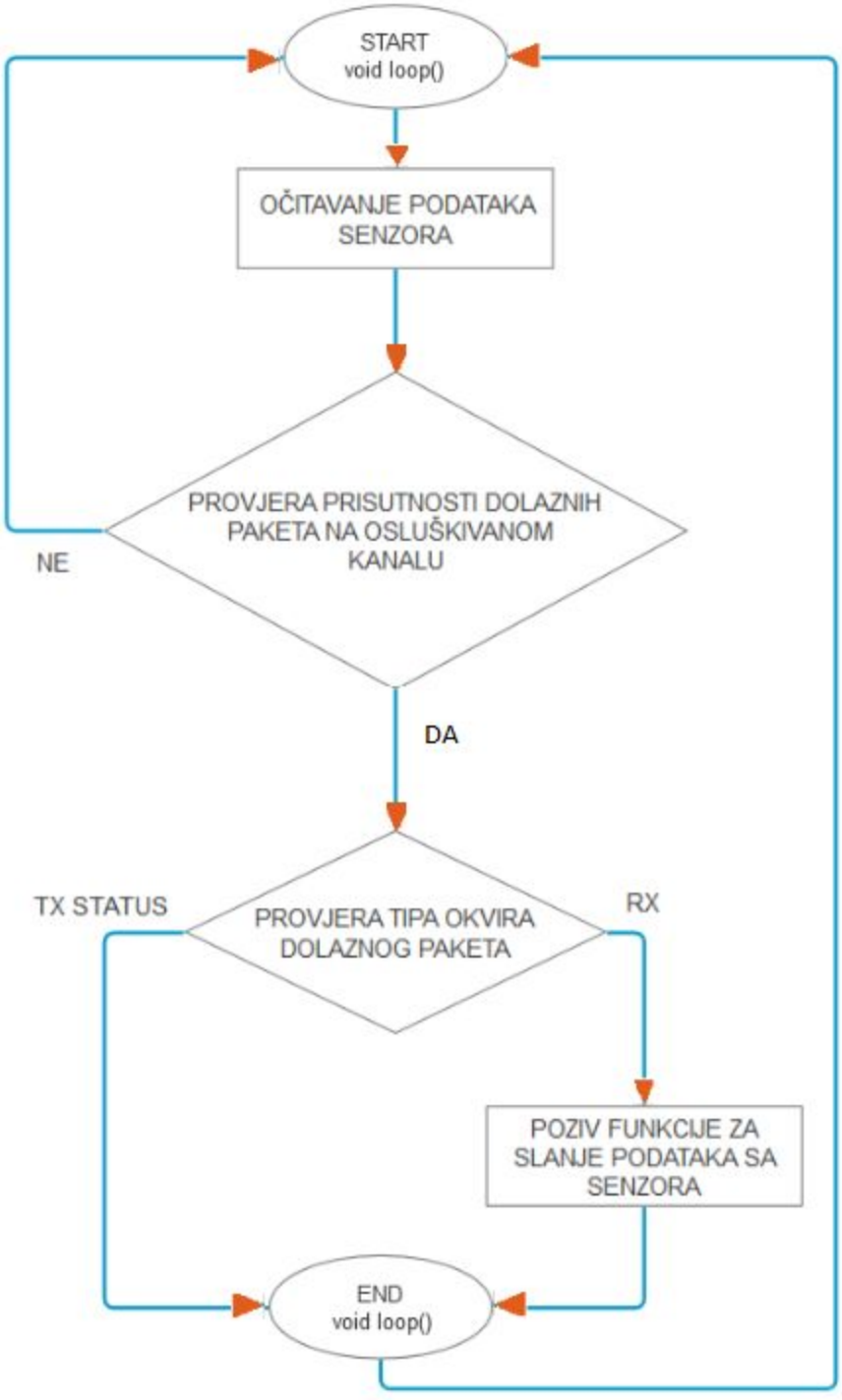

Slika 27. Funkcijski blok dijagram void setup() funkcije koordinatora

Na sljedećoj slici prikazan je funkcijski blok dijagram *void loop()* funkcije koju Arduino mikrokontroler koordinatorskog mrežnog čvora izvršava ciklički.




Slika 28. Funkcijski blok dijagram void loop() funkcije  koordinatorskog čvora

Na sljedećoj slici prikazan je funkcijski blok dijagram *void loop()* funkcije koju Arduino mikrokontroler mjeriteljskog mrežnog čvora izvršava ciklički.




Slika 29. Funkcijski blok dijagram void loop() funkcije  mjernog čvora

### 6.1. Bitne Arduino biblioteke i funkcije

Pri realizaciji bežične senzorske mreže u okviru pisanja programskog koda za Arduino mikrokontrolere korištene su sljedeće biblioteke koje sadrže neke od gotovih setova korištenih funkcija i medtoda:

- *"SimpleZigBeeRadio.h"*
- *"SoftwareSerial.h"*
- "*Adafruit_BME280.h* "
- "*Adafruit_Sensor.h* "

Neke od bitnijih funkcija koje se koriste na strani mjeriteljskih čvorova su :

- "int SENZOR( );" - služi za slanje naredbe za očitanjem koncentracije CO2 na MH-Z19 senzor putem UART sučelja te kao odgovor zaprima rezultat mjerenja u vidu okvira od 9 bajtova

```
int SENZOR(){
  byte cmd[9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79};
  byte response[9]; // polje za odgovor senzora
  //Slanje okvira sa naredbom(upitom)
  Serial.write(cmd, 9); //request PPM CO2
  //praznjenje buffera
  memset(response, 0, 9);
//citanje odgovora
  if (Serial.available() > 0) {
      Serial.readBytes(response, 9); }
  //izoliranje ocitanja koncentracije CO2 u ppm iz odgovora
  ppm_uart = (int)(256 * (int)response[2] + response[3]);
    return ppm_uart;
}
```

- "void SLANJE(int a, int b, int x, uint16_t y );" - služi za paketiziranje mjerenih veličina te identifikatora čvora spremljenih u varijable(koje čine argumente funkcije) u "*TX request*" tip API okvira.

```
void SLANJE( int a,  int b, int x, uint16_t y ){
 //polje korisnog opterecenja unutar API okvira
     uint8_t payload1[5];
         payload1[0] = x  & 0xff;  // identifikator čvora
         payload1[1] = y >> 8  ;   // CO2 viši bajt
         payload1[2] = y & 0xff  ; // CO2 niži bajt
         payload1[3] = a & 0xff  ; // temperatura
         payload1[4] = b & 0xff  ; // relativna vlaznost
 // formiranje API okvira
xbee.prepareTXRequest(koordinator, payload1, sizeof(payload1));
 printPacket( xbee.getOutgoingPacketObject() );
// slanje poruke koordinatoru
 xbee.send();
}
```

### 6.2. Raspberry Pi alati i spremanje podataka u datoteku

Da bi se podatci mjerenih veličina koje Arduino mikrokontroler na koordinatorskom čvoru ispisuje na svoj serijski port putem serijske veze dohvatili na Raspberry Pi računalnu i spremili u datoteku potrebno je instalirati određene programske alate na njega. U nastavku slijede naredbe koje je potrebno upisati u terminal Raspberry Pi računala da bi se alati instalirali.

Ažuriranje operacijskog sustava:

```
sudo apt- get update
```

Instalacija "*Python"* programskog alata:

```
sudo apt- get install python
```

Instalacija "*moreutils"* skupa programskih alata:

```
sudo apt- get install moreutils
```

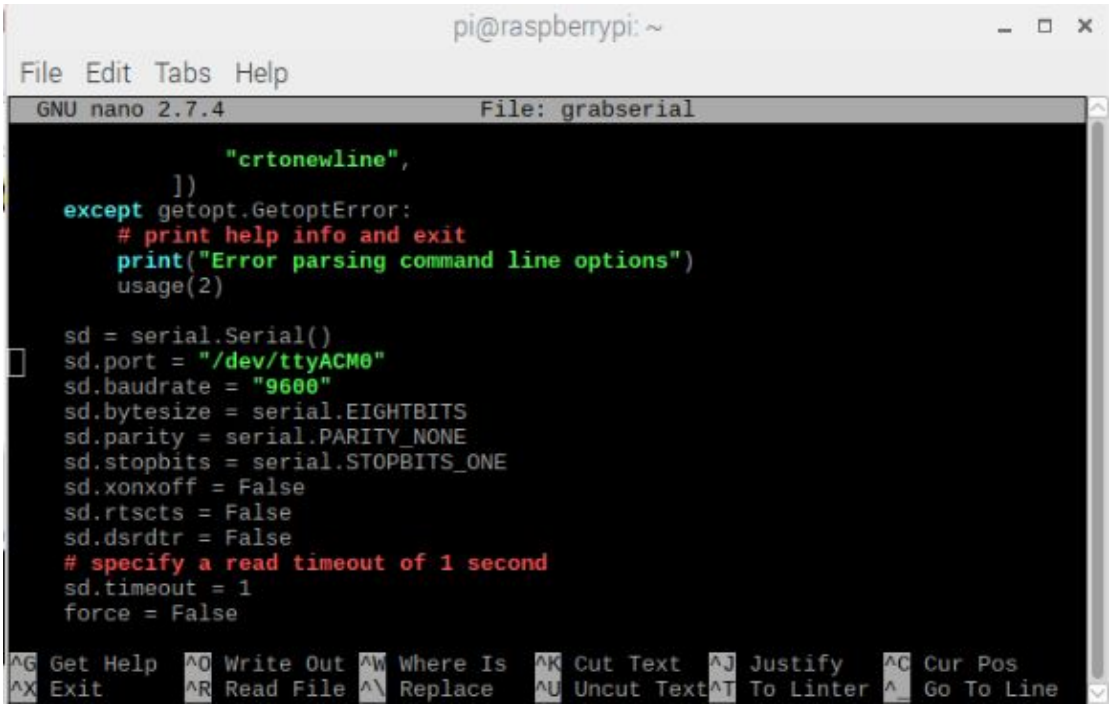
Preuzimanje "*GrabSerial*" programa za prisluškivanje serijskog porta sa *GitHub* servisa:

```
wget
https://raw.githubusercontent.com/tbird20d/grabserial/master/grabseri
al grabserial
```

Konfiguriranje postavki serijske komunikacije unutar "*GrabSerial*" programa

```
nano grabserial
```

Nakon upisa prethodne naredbe potrebno je u programskoj datoteci pronaći postavke za *baud* brzinu i naziv USB uređaja(u ovom slučaju "*ttyACM0"*) koji se prisluškuje kao što je prikazano na sljedećoj slici.

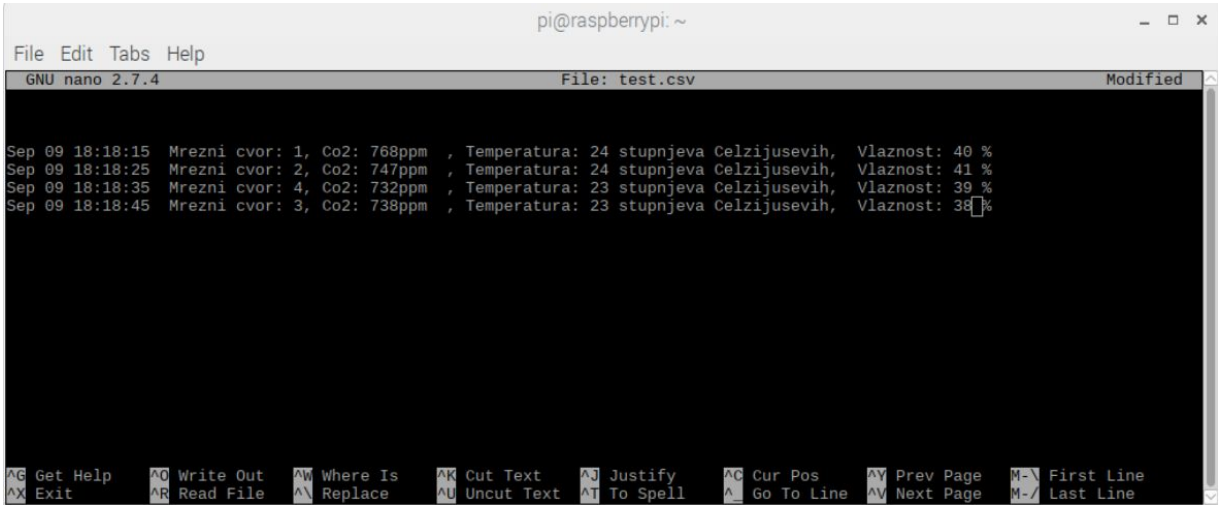



Slika 30. Postavljanje *baud* brzine i odabir USB uređaja

Pokretanje programa koji osluškuje serijski port i očitane podatke upisuje u "*test.csv*" tekstualnu datoteku uz vremenski žig za mjerenja koja su dokumentirana u jednom ciklusu:

```
python grabserial | ts > test.csv
```

Sadržaj tekstualne datoteke prikazan je na slici **31.**




Slika 31. Sadržaj tekstualne datoteke nakon jednog mjernog ciklusa

# 7. ZIGBEE SIGURNOST I ZAŠTITA PODATAKA

ZigBee standard uključuje napredne sigurnosne mehanizme za uspostavljanje i transport mrežnih ključeva te povjerljivost i integritet razmjene samih podataka. Enkripcija se zasniva na modelu koji koristi 128-bitni ***AES*** **( eng.** ***Advanced Encryption Standard*****)** sigurnosni ključ. Za navedene usluge na mreži je zadužen podsloj aplikacijskog sloja (“*APS”*). ZigBee protokol zasnovan je na temelju modela "otvorenog povjerenja". To znači da si svi slojevi protokola vjeruju u međusobnoj komunikaciji, a kriptografska zaštita javlja se samo pri komunikaciji između pojedinih uređaja u mreži. Svaki protokolni sloj je zasebno odgovoran za sigurnost okvira koji mu pripadaju. Moguće je razlikovati dvije vrste sigurnosnih ključeva, mrežni ključ i ključ veze.

***ZigBee Trust Center*** je aplikacija koja se pokreće na uređaju koji ima legitimitet za naspram ostalih uređaja unutar ZigBee mreže za distribuciju sigurnosnih ključeva u svrhu upravljanja mrežnim i krajnjim aplikacijama. *ZigBee Trust Center* može biti koordinatorski uređaj na mreži ili neki drugi kojega je koordinator ovlastio adresirajući ga kao takvog.

## 7.1. Implementacija sigurnosnih mehanizama

Omogućavanjem sigurnosti u mreži *XBee Zigbee* uređaja, isti stječu mrežni ključ kada se pridruže mreži. Prijenos podataka uvijek je šifriran mrežnim ključem i po želji se može daljnje kriptirati pomoću APS ključa veze.

Konfiguracijski parametar **“*Encryption Enable”* (*EE*)** mora biti postavljen na vrijednost:**”1”** izdavanjem *AT* naredbi ili putem *XCTU* konfiguracijskog okruženja na svim uređajima u mreži. Promet u mreži je takovim konfiguriranjem kriptiran mrežnim ključem između pojedinih mrežnih skokova.

Koordinator odabire mrežni sigurnosni ključ za mrežu pomoću parametra ***“Network Encryption Key (NK)”***. Pri zadanoj postavci (***“NK = 0”*****)**, koordinator će odabrati slučajni mrežni ključ.

Usmjerivači i krajnji uređaji s omogućenom zaštitom(***"EE = 1"***) stječu mrežni ključ kada se pridruže mreži. Ako zajednički ključ veze predefiniran od strane koordinatora i ostalih uređaja isti dobivaju mrežni ključ šifriran ključem veze.

Koordinator također mora odabrati ključ veze ***ZigBee Trust Center*** uređaja, koristeći parametar ključa šifriranja("***KY"***). Ako je postavljen na ***"KY = 0"*** (zadano), koordinator odabire slučajan ključ veze i mrežni ključ se distribuira svim uređajima u mreži nezaštićen. Takva praksa nije preporučena jer je dovoljno znati "*PAN ID"* mreže da bi se zaobišli svi sigurnosni mehanizmi.

Ako koordinator koristi unaprijed konfiguriran ključ veze postavljen kao ***"KY> 0"***, koordinator neće slati mrežni ključ nezaštićen već će se samo uređaji s ispravno unaprijed konfiguriranom ključem veze moći pridružiti mreži.

***APS*** **enkripcija** je neobavezni sloj sigurnosti koji koristi ključ veze za šifriranje korisnog opterećenja podataka. Za razliku od mrežne enkripcije koja se dešifrira i šifrira na osnovi pojedinog mrežnog skoka, *APS* enkripciju dekriptira samo odredišni uređaj.

APS enkripcija omogućuje se zasebno za svaki pojedinačni okvir zahtjeva transmisijskog API paketa na način da se u *"**Options**"* polju istog unese vrijednost *"**0x20**"*. Omogućivanje APS enkripcije smanjuje maksimalnu nosivost okvira za devet bajtova.

# 8. ZAKLJUČAK

Koncept umreženog svijeta objekata koji automatizmima izvršavaju međusobnu interakciju te interakciju sa ljudima značajan je i sve više raširen aspekt pri razvijanju novih tehnologija. Interaktivan okoliš ostvaren umrežavanjem "pametnih" objekata pružio bi čovjeku automatizirano svakodnevno okruženje koje se intuitivno prilagođava njegovim potebama. Bežične senzorske mreže su stožerne sastavnice u ostvarivanju takvih sustava jer omogućavaju uređajima unutar istih da međusobno razmjenjuju prikupljene podatke iz okoline na osnovu kojih su definirane njihove daljnje radnje.

Pri preslikavanju projektirane mreže u fizičku domenu primjećeno je da konfiguracijski *software* RF modula, Digi XCTU iz neutvrđenih razloga ponekad ne obnaša zadane funkcije. Problem je riješen povremenim korištenjem starije, jednostavnije inačice konfiguracijskog okruženja.

ZigBee tehnički standard omogućuje stvaranje pouzdanih ekonomičnih i robusnih sustava na vrlo jednostavan način te je jedna od najpogodnijih tehnologija za realiziranje bežičnih senzorskih mreža i gotovo idealno odgovara na sve zahtjeve na iste. U budućnosti, najozbiljniji konkurent za preuzimanje dijela tržišta koji koji se oslanja na ZigBee tehnologiju ima novopečeni *Bluetooth Mesh* bežični standard koji konceptualno, osim u kapacitetnim mogućnostima, višestruko premašuje mogućnosti ZigBee tehnologije.

# LITERATURA

[1] E. Generalic, https://glossary.periodni.com/glosar.php?hr=Beerov+zakon pristupljeno: srpanj 2019

[2] Robert Faludi:Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2010.

[3] https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf pristupljeno: srpanj 2019

[4] http://www.pocket-lint.com/news/129857-zigbee-and-others-explained pristupljeno: srpanj 2019

[5] https://en.wikipedia.org/wiki/Zigbee ,pristupljeno: srpanj 2019

[6] https://hr.wikipedia.org/wiki/ZigBee ,pristupljeno: srpanj 2019

[7] https://github.com/ericburger/simple-zigbee ,pristupljeno: srpanj 2019

[8] http://www.cs.odu.edu/~cs752/papers/zigbee-001.pdf ,pristupljeno: srpanj 2019

[9] https://learn.sparkfun.com/tutorials/sparkfun-bme280-breakout-hookup-guide/all , pristupljeno: srpanj 2019

[10] http://www.science.smith.edu/~jcardell/Courses/EGR328/Readings/XBeeCookbook.pdf , pristupljeno: srpanj 2019

[11] https://courses.csail.mit.edu/6.857/2017/project/17.pdf , pristupljeno: srpanj 2019

[12]https://www.winsen-sensor.com/d/files/PDF/Infrared%20Gas%20Sensor/NDIR%20CO2%20SENSOR/MH-Z19%20CO2%20Ver1.0.pdf , pristupljeno: srpanj 2019

[13] https://elinux.org/Grabserial , pristupljeno: srpanj 2019

# POPIS SLIKA

Slika 1. Prikaz bežične senzorske mreže implementirane u Internet Of Things(IOT) koncept 6

Slika 2. Spektar elektromagnetskog zračenja 7

Slika 3. Zakon obrnutih kvadrata 8

Slika 4. Prikaz funkcijski različitih ZigBee uređaja u mreži 11

Slika 5. Pojednostavljeni prikaz stoga ZigBee protokola 12

Slika 6. Prikaz podržanih topologija ZigBee mreže 14

Slika 7. Prikaz izgleda XBee S2c radio modula 18

Slika 8. Nazivi i raspored pin-ova na XBee S2c radio modulima 20

Slika 9. Dijagram protoka podataka kroz sustav u okruženju UART sučelja 21

Slika 10. Primjer UART podatkovnog paketa: 0x1F (decimalni broj "31") 22

Slika 11. Prikaz izgleda XCTU korisničkog sučelja 23

Slika 12. Prikaz odabira API režima rada unutar XCTU korisničkog sučelja 24

Slika 13. Nazivi i raspored pinova na “ATmega328p” 36

Slika 14. Prikaz izgleda Arduino NANO platforme 36

Slika 16. Prikaz izgleda “BME280” 39

Slika 17. Prikaz mogućih I2C adresnih konfiguracija putem kratkospojnika 39

Slika 18. Prikaz izgleda ”MH-Z19” osjetnika 40

Slika 19. Prikaz primjera Beer-Lambertova zakona 42

Slika 20. Prikaz topologije ostvarene bežične senzorske mreže 45

Slika 21. Prikaz izgleda arhitekture i pokazne sheme spajanja koordinatorskog mrežnog čvora 46

Slika 22. Prikaz izgleda arhitekture i pokazne sheme spajanja mjeriteljskog mrežnog čvora 47

Slika 23. Zajedničke konfiguracijske mrežne postavke svih RF modula u mreži 48

Slika 24. Pojedinačne adresne konfiguracijske postavke svih RF modula u mreži 49

Slika 25. Protok podataka kroz različita sučelja u sustavu 50

Slika 26. Ispis inicijalizacije mreže sa koordinatorskog čvora 51

Slika 27. Funkcijski blok dijagram void setup() funkcije koordinatora 52

Slika 28. Funkcijski blok dijagram void loop() funkcije koordinatorskog čvora 53

Slika 29. Funkcijski blok dijagram void loop() funkcije mjernog čvora 54

Slika 30. Postavljanje baud brzine i odabir USB uređaja 58

Slika 31. Sadržaj tekstualne datoteke nakon jednog mjernog ciklusa 58

# POPIS TABLICA

Tablica 1. Usporedba nekih popularnih bežičnih tehnologija 8

Tablica 2. Tipovi i klase ZigBee uređaja 9

Tablica 3.Usporedba ZigBee uređaja različitih po ovlastima unutar mrežnog sloja 14

Tablica 4. Korištene hardware komponente 16

Tablica 5. Tehničke karakteristike XBee S2c RF modula 18

Tablica 6. Neke od najvažnijih AT naredbi 24

Tablica 7. Osnovna struktura XBee API okvira 26

Tablica 7.1. Neke od najvažnijih AT naredbi 27

Tablica 8. AT command (immediate) tip API okvira 28

Tablica 9. AT command response tip API okvira 29

Tablica 10. TX request tip API okvira 30

Tablica 11. TX response tip API okvira 32

Tablica 12. RX received tip API okvira 32

Tablica 13. Tehničke karakteristike “Arduino NANO” 36

Tablica 13.1. Tehničke karakteristike “BME280” 37

Tablica 14. Tehničke karakteristike “MH-Z19” 40

Tablica 15. UART postavke 42

Tablica 16. Prikaz okvira naredbe 42

Tablica 17. Prikaz okvira odgovora **43**

# Prilog 1 - Korišteni programski kodovi za Arduino mikrokontrolere

## 1. Arduino programski kod koordinatora

```
#include <SimpleZigBeeRadio.h>
#include <SoftwareSerial.h>

 //Kreiranje XBee objekta
 SimpleZigBeeRadio xbee = SimpleZigBeeRadio();
 //Kreiranje SoftwareSerial objekta
 SoftwareSerial xbeeSerial(10, 11); // (RX=>DOUT, TX=>DIN)
 //Kreiranje SimpleZigBeeAddress objekta, buduci da nismo zadali
nikakvu vrijednost konstruktor dodjeljuje broadcast adresu
 SimpleZigBeeAddress broadcast = SimpleZigBeeAddress();
 // Paket koji se salje:
 SimpleZigBeePacket zbp = SimpleZigBeePacket();

 //Pomocne varijable za pracenje proteklog vremena
 unsigned long previousMillis1 = 0;
 unsigned long previousMillis2 = 0;
 unsigned long interval1 = 60000;
 unsigned long interval2 = 10000;
 //Ostale pomocne varijable
 int broj = 0;
 int novi_broj = 1;
 int kontrola = 0;
 uint8_t adresa_16_64[10][6]; //Adresna matrica za skladistenje
adresa uredjaja na mrezi
```

```
 void setup() {
  //iniciranje komunikacije na serijskim portovima
  Serial.begin( 9600 );
  xbeeSerial.begin( 9600 );
  //postavljanje serijskog porta za XBee radio modul
  xbee.setSerial( xbeeSerial );
  // osiguravanje povratne informacije sa statusom proteklog
prijenosa (primanje TX statusnih paketa)
  xbee.setAcknowledgement(true);
  // Slanje "ND" AT naredbe za prikupljanje adresa uredjaja na mrezi
  AT_ZAHTJEV_ND();
  uint8_t exFrame[] = {
0x10,0x00,0x00,0x13,0xA2,0x00,0x00,0x00,0x00,0x00,0xff,0xfe,0x00,0x00
,0xff,0xff };
  zbp.setFrameData(0, exFrame, sizeof(exFrame));
     }


  void loop(){
    while( xbee.available() ){
      xbee.read();
      if( xbee.isComplete() ){
        //PROVJERA TIPA PAKETA
        if( xbee.isATResponse() ){
          uint8_t frameID = xbee.getIncomingFrameID();
          uint16_t at_naredba = xbee.getATResponseCommand();
          uint8_t atStat = xbee.getATResponseStatus();
          uint8_t atLength = xbee.getATResponsePayloadLength();
```

```
          //Ukoliko je tip dolaznog okvira odgovor na AT naredbu
          // slijedi spremanje adresa u matricu
          if((at_naredba == 0x4E44)&&(atStat==0x0)){
           for(int j=6; j<=9; j++){
           adresa_16_64[broj][j-6]=  xbee.getATResponsePayload(j);
          if (j>=8)
          {adresa_16_64[broj][j-4] = xbee.getATResponsePayload(j-8);
}
          }
           broj++;
           novi_broj = broj;
           kontrola = novi_broj;
          }
           //Ukoliko je tip dolaznog okvira RX slijedi ispis
primljenih
           //podataka sa senzora na serijski port
            }else if( xbee.isRX() ){

         Serial.print(" Mrezni cvor: ");
         Serial.print(xbee.getRXPayload(0) );

         Serial.print(", Co2: ");
         Serial.print(xbee.getRXPayload(1)*256+xbee.getRXPayload(2));
         Serial.print("ppm  ,");

         Serial.print(" Temperatura: ");
         Serial.print(xbee.getRXPayload(3) );
         Serial.print(" stupnjeva Celzijusevih");
```

```
        Serial.print(",  Vlaznost: ");
        Serial.print(xbee.getRXPayload(4) );
        Serial.print(" %");

        Serial.println();
        kontrola++;
       }
      }
     }

       unsigned long currentMillis = millis();
    // Provjera ciklickog vremenskog uvjeta(intervala) za
    //ponovno sakupljanje adresa mreznih uredjaja
   if(currentMillis - previousMillis1 > interval1) {
     previousMillis1 = currentMillis;
     broj = 0;
     AT_ZAHTJEV_ND();
     currentMillis = millis();
     }
    // Provjera ciklickog vremenskog uvjeta(intervala) za
    //poziv funkcije koja proziva mjeriteljske cvorove i zahtjeva
podatke od njih
     else if( ( currentMillis - previousMillis2 > interval2) &&
(kontrola != 0 ) )
     {
        previousMillis2 = currentMillis;
        PROZIVANJE();
       }
       delay (10);
```

```
        }

  // Funkcija za prozivanje mjernih mreznih cvorova
void PROZIVANJE(){
for( int k = 0; k < novi_broj; k++ ){

          zbp.setFrameData( 6, adresa_16_64[k][0] );
          zbp.setFrameData( 7, adresa_16_64[k][1] );
          zbp.setFrameData( 8, adresa_16_64[k][2] );
          zbp.setFrameData( 9, adresa_16_64[k][3] );
          zbp.setFrameData( 10, adresa_16_64[k][4] );
          zbp.setFrameData( 11, adresa_16_64[k][5] );
          xbee.send( zbp );
          }
          kontrola = 0;
           delay(10);
         }

  //Funkcija koja izdaje AT naredbu za prikupljanje adresa uredjaja
na mrezi
void AT_ZAHTJEV_ND(){
    xbee.prepareATCommand('ND');
    xbee.send();
}
```

**2. Arduino programski kod mjeriteljskog čvora**

```
#include <SimpleZigBeeRadio.h>
#include <SoftwareSerial.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

  //Kreiranje XBee objekta
  SimpleZigBeeRadio xbee = SimpleZigBeeRadio();
  //Kreiranje SoftwareSerial objekta
  SoftwareSerial xbeeSerial(10, 11); // (RX=>DOUT, TX=>DIN)
  //Kreiranje SimpleZigBeeAddress objekta
  SimpleZigBeeAddress64 koordinator =
SimpleZigBeeAddress64(0x00000000,0x00000000);
 //identifikator rednog broja mjeriteljskog čvora
 //Kreiranje BME280 objekta
 Adafruit_BME280 bme; // I2C

 int cvor = 1;
 //varijable za spremanje mjerenih veličina
uint16_t ppm_uart;
 int temp;//Stores temperature value
 int vlaznost;
 int tlak;

  void setup() {
```

```
  //iniciranje komunikacije na serijskim portovima
  Serial.begin( 9600 );
  xbeeSerial.begin( 9600 );
  //postavljanje serijskog porta za XBee radio modul
  xbee.setSerial( xbeeSerial );
  //osiguravanje povratne informacije sa statusom proteklog prijenosa
//(primanje TX statusnih paketa)
   xbee.setAcknowledgement(true);
  }



   void loop() {
  while( xbee.available() ){

      xbee.read();
      if( xbee.isComplete() ){

        //PROVJERA TIPA PAKETA

   if( xbee.isRX() ){
    //poziv SENZOR funkcije za CO2 mjerenje
         SENZOR();
         // Mjerenje temperature i relativne vlaznosti sa BME280
         bme.begin();
         temp =  bme.readTemperature();
         vlaznost = bme.readHumidity();
         delay(10);
              //poziv SLANJE funkcije za formiranje i slanje okvira
//             sa mjerenim velicinama
```

```
            SLANJE(cvor, ppm_uart, temp, vlaznost);
       }
      }
     }
     delay(10);
  }

// int SENZOR( );” - služi za slanje naredbe za očitanjem
koncentracije CO2 na MH-Z19
// senzor putem UART sučelja te kao odgovor zaprima rezultat
// mjerenja u vidu okvira od 9 bajtova
int SENZOR(){
  byte cmd[9] = {0xFF,0x01,0x86,0x00,0x00,0x00,0x00,0x00,0x79};
  byte response[9]; // polje za odgovor senzora
  //Slanje okvira sa naredbom(upitom)
  Serial.write(cmd, 9); //request PPM CO2
  //praznjenje buffera
  memset(response, 0, 9);
//citanje odgovora
  if (Serial.available() > 0) {
      Serial.readBytes(response, 9); }
  //izoliranje ocitanja koncentracije CO2 u ppm iz odgovora
  ppm_uart = (int)(256 * (int)response[2] + response[3]);
    return ppm_uart;
}


// void SLANJE(int a, int b, int x, uint16_t y );” - služi za
// paketiziranje mjerenih veličina te identifikatora čvora
```

```
// spremljenih u varijable(koje čine argumente funkcije)
// u  “TX request” tip API okvira.
void SLANJE(  int x, uint16_t y, int a, int b  ){
          //polje korisnog opterecenja unutar API okvira
uint8_t payload1[5];
          payload1[0] = x  & 0xff; // identifikator čvora
          payload1[1] = y >> 8  ;  // CO2 viši bajt
          payload1[2] = y & 0xff  ; // CO2 niži bajt
          payload1[3] = a & 0xff  ; // temperatura
          payload1[4] = b & 0xff  ;// relativna vlaznost
          // formiranje API okvirA
  xbee.prepareTXRequest(koordinator, payload1, sizeof(payload1));
   // slanje poruke koordinatoru
  xbee.send();
}
```