Mikroupravljački sustav za udaljenu detekciju pokreta

Biro, Borna

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:391878

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

MIKROUPRAVLJAČKI SUSTAV ZA UDALJENU DETEKCIJU POKRETA

Završni rad

Borna Biro

Osijek, 2017.

Sadržaj

1. UVOD 1
1.1. Zadatak rada 3
2. TERORIJSKA PODLOGA 4
2.1. Diskretni modulacijski postupci 4
2.2. Senzor udaljenosti 7
2.3. Mjerenje temperature i vlage zraka 10
2.4. Mikroupravljači 14
2.5. Arduino platforma 17
3. IZRADA SHEME I PROGRAMSKE PODRŠKE 21
3.1. Izrada sheme 21
3.2. Izrada programa 24
3.3. Napajanje uređaja i izbor baterije 28
4. PRIKAZ PRAĆENJA RADA ZAVRŠNOG RADA 33
5. ZAKLJUČAK 38
LITERATURA 39
SAŽETAK 41
ABSTRACT 42
ŽIVOTOPIS 43
PRILOG 44

1. UVOD

Zadatak ovog rada jest napraviti mrežu senzora koji detektiraju pokret ili prepreke pomoću ultrazvučnog senzora, mjere temperaturu i vlagu zraka i zatim prikupljene podatke šalju glavnoj jedinici. Glavna jedinica zatim obrađuje te podatke u smislu da provjerava da li se prepreka približila na zadanu udaljenost, te ukoliko je, šalje taj podatak na mobilni uređaj preko Bluetooth™ veze. Mobilni uređaj mora sadržavati Bluetooth™ i aplikaciju za prikaz podataka koji su primljenu preko Bluetooth™ veze. Korištena aplikacija za to je „Serial Bluetooth“ koja se instalira na Android™ uređaj. Platforma senzora i glavne jedinice je bazirana na Arduino Nano mikroupravljačkoj platformi koju je osmislila tvrtka „Arduino“. Senzori koji su korišteni za detekciju pokreta su ultrazvučni detektori udaljenosti, model SRF-05, koji sadrži na sebi predajnik i prijamnik ultrazvučnog signala. Detekcija se bazira na tome da se mjeri vremenski interval od kad je poslan ultrazvučni signal do trenutka kada je isti primljen i ujedno reflektiran od prepreku. Mjerenje vlage i temperature se izvodi pomoću senzora pod nazivom DHT22, a može mjeriti temperaturu u opsegu od -40°C do +80°C, te vlagu u opsegu od 0% do 100% i koristi jedan izvod za prijenos podataka. Prijenos podataka pomoću radio valova izvodi se pomoću modula koji se baziraju na nRF24L01(+) integriranom krugu. Integrirani krug komunicira s mikroupravljačem preko SPI komunikacije, te je zaslužan za postavljanje zaglavlja na poslane podatke koji uključuje 40 bitnu adresu, cikličku redundantnu provjeru podataka, bitove za potvrdu primitka paketa (podataka), bitove za kontrolu paketa. Također radi rentransmisiju paketa, ukoliko dođe do gubitka paketa, kontrolu izlazne snage, podešavanje unutrašnjeg sklopovlja za prijam odnosno predaju paketa, podešavanje brzine prijenosa podataka. Prijenos podataka je podešen na 1 Mb/s, a izlazna snaga na 0 dBm. Kada se senzorske jedinice priključe na izvor napajanja, podešavaju se parametri za nRF24L01(+) integrirani krug, odnosno parametri potrebni za komunikaciju, te se mjeri napon izvora napajanja (litiji ion baterija napona 3.7 V). U slučaju da je napon baterije prenizak za rad, rad senzora se obustavlja, te odmah nakon pokretanja mikroupravljač i nRF24L01(+) rade u režimu niske potrošnje. Ukoliko je napon odgovarajući, započinje proces sinkronizacije u kojem se čeka da glavna jedinica da signal da je aktivna. Taj proces traje 2 minute, te je u tom vremenu nRF24L01(+) postavljen na prijam paketa, a mikroupravljač ulazi u režim rada s niskom potrošnjom u trajanju od 500 milisekundi nakon čega provjerava da li je dospio kakav paket, ako nije, proces se ponavlja, sve dok ne isteknu dvije minute. Ukoliko u tih dvije minute stigne podatak za sinkronizaciju, senzori počinju mjeriti i slati podatke glavnoj jedinici. Ukoliko ne dođe nikakav

podatak, mikroupravljač, postavlja nRF24L01(+) u režim niske potrošnje energije, te je za ponovni proces sinkronizacije potrebno pritisnuti tipkalo za sinkronizaciju. Podaci se sa senzora šalju svakih 250 milisekundi, te se pri tome traži od glavne jedinice potvrda primitka paketa. Ukoliko niti nakon 10 pokušaja retransmisije paketa ne dođe potvrda, senzori se vračaju na početak rada, odnosno na čekanje podatka za sinkronizaciju. Glavna jedinica očekuje podatak svakih nekoliko milisekundi, no ukoliko podatak ne dođe nakon 1.5 sekunde, glavna jedinica smatra da je veza između senzora i nje „pukla“ odnosno da veza ne postoji. Glavna jedinica na sebi sadrži Bluetooth™ modul za komunikaciju koji na strani mikroupravljača koristi UART komunikaciju s brzinom od 9600 bauda.

1.1. Zadatak rada

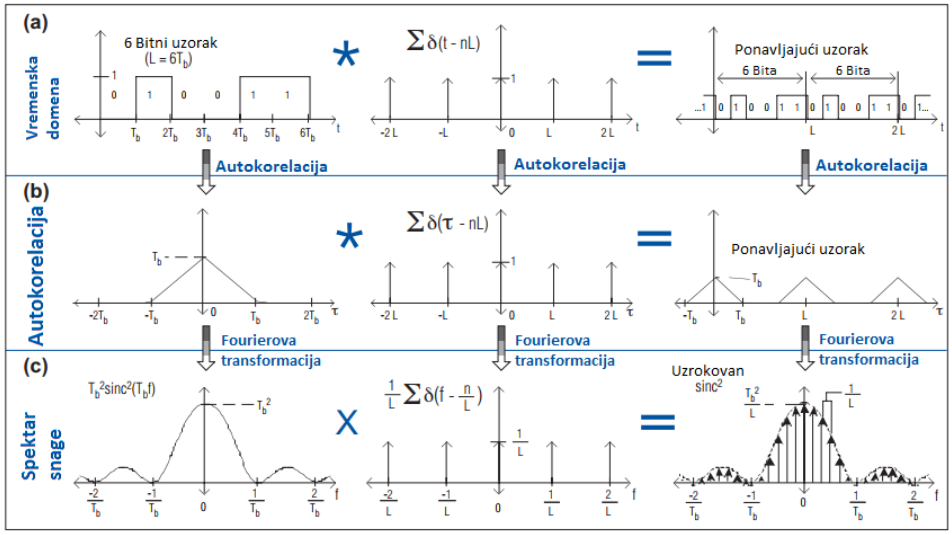
U ovom radu potrebno je napraviti mikroupravljački sustav s mrežom senzora za detekciju pokreta. Mrežu je potrebno napraviti pomoću 3 NRF24L01 komunikacijska modula, od kojih je jedan glavni prijamni uređaj, a dva su odašiljača modula sa senzorima za mjerenje udaljenosti. Glavni modul treba imati mogućnost povezivanja s pametnim telefonom pomoću Bluetooth tehnologije.

2. TERORIJSKA PODLOGA

2.1. Diskretni modulacijski postupci

Za prijenos podataka, koji su diskretni, pomoćnu radio valova, potrebno je koristiti neku vrstu modulacije. Modulacija je postupak pri kojem se sinusnom signalu nosiocu koji je na višoj frekvenciji nego signal ili informacija koju želimo prenijeti mijenjaju neki od parametara (faza, amplituda ili frekvencija) da bi se stvorio novi signal koji u sebi sadrži prvobitnu informaciju s kojim je modularan (oblikovan) taj novi signal. Takav signal se filtrira, zatim šalje na miješalo gdje se miješa s sinusnim signalom frekvencije pogodnom za slanje. Takav signal je vrlo visoke frekvencije. Nakon toga se filtrira i šalje na pojačalo snage. Razlog ovog cijelog postupka je taj da se dobije vrlo efikasan signal, odnosno da je spektralna efikasnost vrlo velika, pa iz toga razloga takav signal neće puno zauzimati u spektru. Ukoliko bi NRZ (eng. *Not return to zero*) format slali bez modulacije, on bi zauzeo veliki dio spektra, što je vidljivo na slici 2.1. (dijagram pod oznakom c) u donjem lijevom kutu). Nadalje, ukoliko se signali šalju pomoću visokofrekventnog signala, mogu se koristiti manje antene što omogućava da uređaji budu kompaktniji i manjih dimenzija.

Blok dijagram jednog klasičnog komunikacijskog sustava gdje se vidi prethodno objašnjeno je vidljiv na slici 2.2.

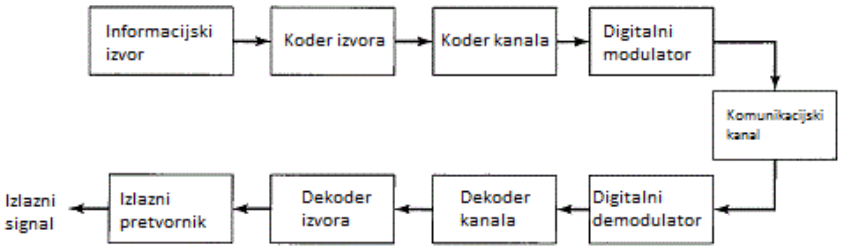


Slika 2.1. Spektar NRZ formata [1]



Slika 2.2. Blok dijagram komunikacijskog sustava zajedno s modulatorom [2]

Razlika između diskretnih modulacijskih postupaka i onih kod kojih se kao informacija koristi kontinuiran signal je taj što se kod diskretnih mogu dodati zaštita koja omogućava da se podaci koji se pošalju zaštite od smetnji, tj. da, ukoliko i dođe do smetnje, bude moguća detekcija pogreške ili pak ispraviti pogrešku. Taj postupak zaštite podataka od pogreške čini koder kanala. Na slici 2.3. je vidljiv njegov položaj u tipičnom komunikacijskom sustavu, on se nalazi na ulazu u modulator, a prethodi mu koder izvora koji smanjuje zalihost podataka, što omogućava veće brzine prijenosa podataka. Takvi postupci nisu mogući kod analognih modulacijskih postupaka koji koriste kontinuirane signale kao informaciju.



Slika 2.3. Diskretni komunikacijski sustav [3]

Kao što je spomenuto, kod modulacija, signalu nosiocu moguće je mijenjati amplitudu, fazu ili pak frekvenciju, stoga postoje:

- FSK (eng. *Frequency Shift Keying*) modulacija – Kao parametar signala nosioca, mijenja se njegova frekvencija
- ASK (eng. *Amplitude Shift Keying*) modulacija – Kao parametar signala nosioca, mijenja se njegova amplituda
- PSK (eng. *Pahse Shift Keying*) modulacija – Kao parametar signala nosioca, mijenja se njegova faza

- QAM (eng. *Quadrature Amplitude Modulation*) modulacija – Kao parametar signala nosioca, mijenjaju mu se faza i amplituda

Kod diskretnih modulacija parametri signala, odnosno amplituda, frekvencija i faza mijenjaju se diskretno, točnije u koracima. Na primjeru 16FSK vidljivo je da ima 16 mogućih stanja frekvencije, što bi značilo da takva modulacija, jednim svojim stanjem reprezentira 4 bita, a frekvencije mogu biti slijedeće:

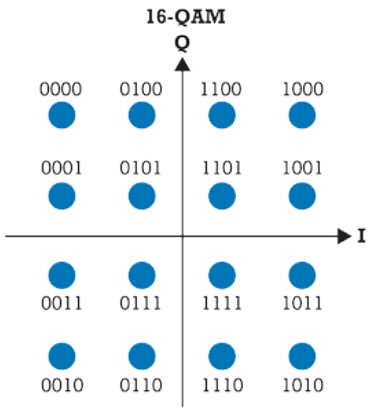
bitovi 0000 = Frekvencija f_0

bitovi 0001 = Frekvencija 1.5 f_0

bitovi 0010 = Frekvencija 2 f_0

i tako nadalje, sve od bitova 1111.

Kod PSK i QAM modulacije, za prikaz svih mogućih stanja faza i amplituda koriste se konstelacijski dijagrami. Oni sadrže sve moguće kombinacije faza i amplituda (kod QAM modulacije) te bitove koji svako od tih stanja reprezentiraju. Primjer jednog konstelacijskog dijagrama za 16QAM modulaciju vidljiv je na slici 2.4.



Slika 2.4. Konstelacijski dijagram za 16QAM modulaciju [4]

Princip rada modulacije kod nRF24L01 integriranog kruga, koji je korišten u ovom radu, je GFSK modulacija. GFSK modulacija (engl. *Gausian Frequency Shift Keying* [5]) je diskretna modulacija kod koje se kao parametar signala nosioca mijenja frekvencija, slično kao kod FSK modulacije, no razlika je u tome što se signal koji sadrži informaciju, prije moduliranja propusti kroz Gaussov filtar, te se takvom signalu smanjuje strmina promjene stanja iz logičke nule u

logičku jedinicu, a time i spektar. Radi toga, promjena frekvencije ne nastupa rapidno, nego postepeno, čime se omogućava da s takvim modulacijskim postupkom zauzima manji dio radio spektra.

2.2. Senzor udaljenosti

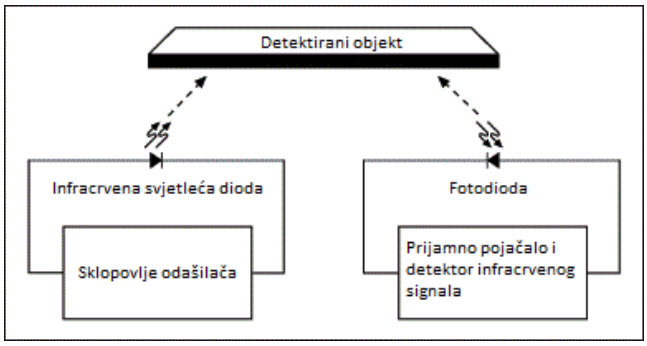
Da bi se mogla izvesti detekcija pokreta, potrebno je mjeriti udaljenost nekog predmeta, te zatim analizirati da li se udaljenost promijenila. Ako je, došlo je do nekog pokreta predmeta, ako nije, predmet stoji. Za mjerenje udaljenosti potreban je sklop koji će udaljenost, koja je fizička veličina, pretvoriti u neku vrstu električkog signala. Taj signal može biti slijedeće:

- iznos struje koji je ovisan o udaljenosti
- iznos napona koji je ovisan o udaljenost
- iznos frekvencije signala koja je ovisna o udaljenosti
- podatak (paket) koji sadrži informaciju o udaljenosti
- impuls, gdje je njegovo trajanje ovisno o udaljenosti

Postoje više vrsta senzora koji se razlikuju po načelima kojim mjere udaljenost, stoga postoje senzori koji mjere udaljenost na slijedeće načine:

- infracrveni senzor udaljenosti
- ultrazvučni senzor udaljenosti
- linearni senzor udaljenosti (pozicije)
- laserski senzori udaljenosti

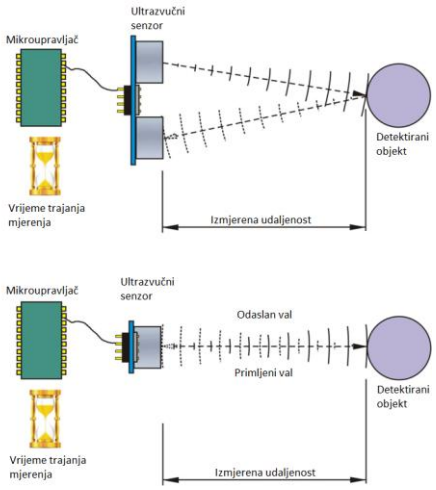
Infracrveni senzori udaljenosti su senzori koji se sastoje od jedne infracrvene svjetleće diode i jedne infracrvene foto diode u istom kućištu. Princip se bazira na tome da se kroz infracrvenu svjetleću diodu propusti pravokutni signal, tj. niz impulsa. Takav snop svijetlosti putovati će do prepreke, zatim će se reflektirati od prepreku i vratiti nazad u infracrvenu foto diodu (vidljivo na slici 2.5.). Takav signal će biti manjeg intenziteta, stoga će udaljenost biti obrnuto proporcionalna s udaljenošću.



Slika 2.5. Princip rada infracrvenog senzora udaljenosti [6]

Na taj način je moguće mjeriti manje iznose udaljenosti, reda do 15 cm, jer iznos reflektiranog snopa svjetlosti na veće udaljenosti biva sve slabiji. Taj je problem moguće riješiti dodavanjem infracrvene diode jačeg intenziteta, no pošto je udaljenost veća, biti će veći i šum u reflektiranoj zraci, što može rezultirati u pogrešnom očitanju udaljenosti.

Zbog svojeg kratkog dosega, nije odgovarajući senzor za ovu namjenu. Stoga se u ovome radu koristi ultrazvučni senzor udaljenosti. Njegov princip rada bazira se na tome da se koriste jedan Piezo ultrazvučni odašiljač i jedan ultrazvučni prijamnik. Kada se pošalje impuls na Piezo ultrazvučni odašiljač, on odašilje zvučni val frekvencije veće od 18kHz (zvuk koji je iznad sluha ljudskoga uha). Takav val ima vrlo uzak snop rasprostiranja. Kada zvučni val naiđe na prepreku, on se reflektira od nju i takav odbijen, vraća se nazad u prijamnik.



Slika 2.6. Princip rada ultrazvučnog senzora udaljenosti [7]

Udaljenost se računa tako što se mjeri vremenski interval između trenutka kada se odaslan ultrazvučni val i trenutka kada je primljen reflektirani ultrazvučni val, te pomoću brzine zvuka. Brzina zvuka je 340 m/s na 15 °C, što znači da zvuk prijeđe put od 340 m u jednoj sekundi. Ako se zna vrijeme u kojem se zvuk prešao određeni put, uz zadanu brzinu zvuka, može se doći do iznosa udaljenosti. Ono što treba napomenuti je to da je vrijeme koje se dobije ovim putem dvostruko veće, jer zvuk putuje prema prepreci, te se zatim reflektira nazad. Pošto je nepoznanica udaljenost od senzora do prepreke, a ne od senzora do prepreke i nazad, vremenski interval koji se dobije, potrebno je podijeliti s 2 da bi se dobila točna udaljenost. Na slijedećem primjeru se može vidjeti objašnjeno.

Na ultrazvučni odašiljač se propusti signal frekvencije 18 kHz. Taj signal, koji se reflektirao o prepreku, vrati se u prijamnik nakon 700 µs (mikro sekundi). Brzina zvuka je 340 m/s [10] pri 15 °C. Traženi podatak je na kojoj udaljenosti se nalazi prepreka.

Δt = 700 µs

v = 340 m/s

d = ?

$$v = \frac{s}{t} \Rightarrow\Rightarrow s = v * t \tag{2-1}$$

Pošto je vrijeme duplo veće, dobiveni vremenski interval dijeli se s 2.

$$s = v * \frac{t}{2} = 340 * \frac{700 * 10^{-6}}{2} = 0.119\, m$$

Rezultat je da se prepreka nalazi na 11.9 cm.

Ovakav senzor je vrlo pogodan za rad jer je jednostavnog principa rada, jednostavne konstrukcije, te može mjeriti udaljenosti do 4 m.

Laserski senzori su vrlo kompleksni, te teško nabavljivi, te mogu mjeriti udaljenosti do nekoliko desetaka metara [8] , što nije potrebno za ovaj rad.

Linearni senzori udaljenosti su u suštini senzori pozicije, čiji se rad zasniva na principu mjerenja otpora. Izmjereni otpor odgovara udaljenosti na kojoj se neki predmet nalazi. Vrlo su jednostavne konstrukcije, a sastoje se od otporničke trake i klizača koji klizi po njoj. Primjer jednog takvog senzora je na slici 2.7.

Također, osim otporničkog (drugim nazivom potenciometarskog) senzora, mogu postojati izvedbe čiji se princip rada bazira na magnetizmu ili na brojanju impulsa. Niti jedan od navedenih linearnih senzora nisu odgovarajući za ovaj rad, jer je prepreku potrebno učvrstiti za klizač, dok je za ovaj rad potrebno bezkontaktno mjerenje udaljenosti.



Slika 2.7. Linearni senzor pozicije (udaljenosti) [9]

Uz uvjete da senzor za mjerenje udaljenosti mora mjeriti barem do 4 metra, mjerenje udaljenosti mora biti beskontaktno i ne smije smetati čovjeku, mora biti jednostavan, lako dostupan, odabrana ultrazvučna metoda mjerenja udaljenosti pomoću SRF-05 senzora. Može mjeriti udaljenost do 4 metra najviše, te 3 cm najmanje s točnošću od 0.1 cm, a kao izlaznu veličinu daje impuls čije trajanje logičke jedinice predstavlja vremenski interval koliko je trebalo da se ultrazvučni signal pošalje, reflektira i vrati nazad i napaja se preko 5 V istosmjernog napona.

2.3. Mjerenje temperature i vlage zraka

Na rad je, osim senzora za mjerenje udaljenosti, dodan i senzor za mjerenje temperature i vlage. Mjerenje temperature se može izvesti na više načina: termopar, termistor, poluvodička dioda, integrirani krug. U nastavku će se opisati svaki od načina pojedinačno.

- Termopar – Bazira se na principu da se dva različita metala na jednom mjestu spoje, a krajevi da im budu slobodni. Postojanjem razlike u temperaturi ta dva različita materijala, na njihovim krajevima se pojavi razlika potencijala, koja je proporcionalna rastu temperature. Pomoću termopara moguće je mjeriti temperature od 95 °C do 1200 °C za K tip [11].

- Termistor – Vrsta otpornika kojem se otpor mijenja u odnosu na temperaturu. Postoje dvije vrste, NTC (engl. *Negative Temperature Coefficient*) kojem kako temperature raste, iznos otpora se smanjuje i PTC (engl. *Positive Temperature Coefficient*) kojem kako temperature raste, raste i iznos otpora. Pretvorbu temperature u neku električnu veličinu, moguće je izvesti na dva načina, da se temperatura pretvara u iznos istosmjernog napona ili da se temperatura pretvara u iznos istosmjerne struje. Na taj način se uz jednu fiksnu veličinu (kod pretvorbe u napon, struja je konstantna, kod pretvorbe u struju, napon je konstantan) može izračunati otpor, nakon čega se preko Steinhart-Hart jednadžbe može dobiti iznos temperature.

Steinhart-Hart jednadžba [12]:

$$\frac{1}{T} = A + B\ln(R) + C[\ln(R)]^3 \qquad (2\text{-}2)$$

gdje je:

T – Temperatura (izražena u Kelvinima)

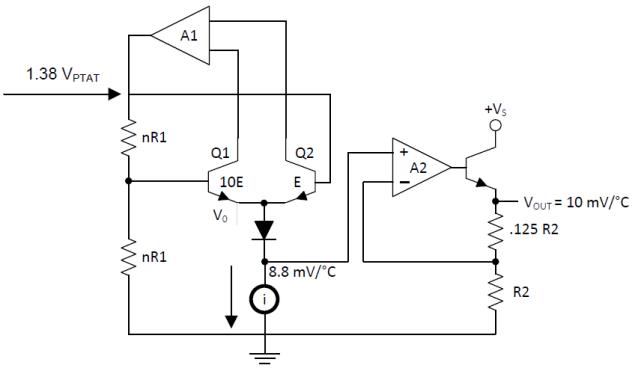
R – Otpor na temperaturi T (izražena u Ohmima)

A, B, C – Steinthart-Hart koeficijenti (najčešće vrijednosti za koeficijente, koji vrijede za veći dio termistora iznose [13]:

$$\mathrm{A} = 0.001129148 \quad \mathrm{B} = 0.000234125 \quad \mathrm{C} = 8.76741 * 10^{-8}$$

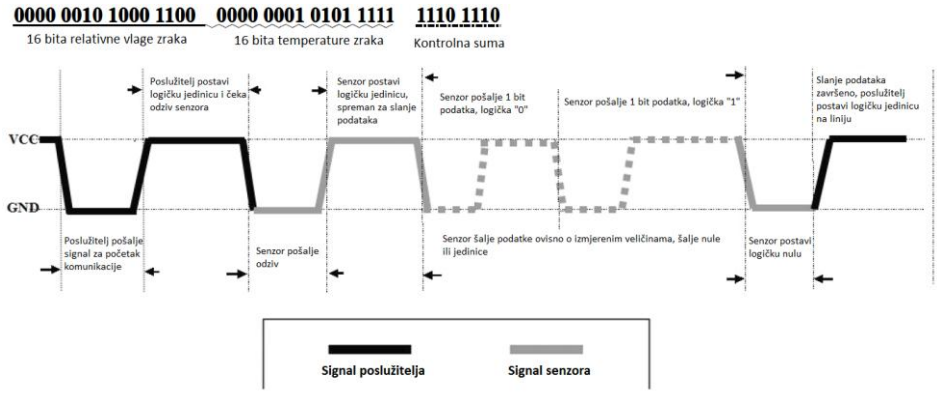
- Dioda – Pomoću obične poluvodičke diode je moguće dobiti iznos temperature tako da se mjeri pad napona na njoj u normalnom području, te se zatim preko *Lookup table* (tablica pretraživanja) može dobiti iznos temperature za zadani napon.

- Integrirani krugovi – Postoje i gotovi poluvodički integrirani krugovi koji temperaturu pretvaraju u iznos istosmjernog napona u nekom odnosu (kao na primjer. LM35DZ kod kojeg je 1 °C = 10 mV) ili daju gotov iznos temperature u podatkovanom obliku (kao na primjer DS18B20, DHT11, DHT22). Unutar tih integriranih krugova, sama pretvorba temperature u neku električnu veličinu se izvodi na prethodno navedene načine (pomoću temistora ili diode), samo što je reprezentacija podataka drugačija, što je vidljivo sa slike 2.8. koja prikazuje unutrašnji spoj temperaturne sonde LM35DZ firme Texas Instruments™.

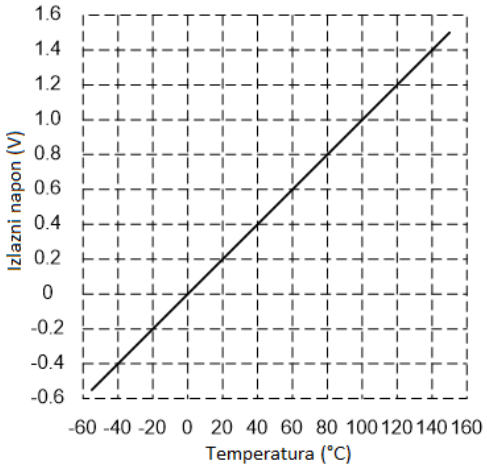


Slika 2.8. Blok dijagram integriranog kruga (temperaturne sonde) LM35DZ

Reprezentacija, odnosno prikaz izmjerene temperature može biti različit od sonde do sonde. na slikama 2.9. i 2.10. se vidi kakav izlaz daju sonde LM35DZ i DHT22 u ovisnosti o temperaturi.



Slika 2.9. Izlazni signal sa sonde DHT22 [15]



Slika 2.10. Izlazni signal s sonde LM35DZ [14]

Mjerenje vlage se zasniva na principu da postoji element koji je osjetljiv na vlagu, čime se u ovisnosti količine vlage u zraku, mijenja njegov otpor. Pošto su to vrlo visoke vrijednosti otpora, potrebno je visoki otpor pretvoriti u iznos malog otpora u nekom linearnom omjeru da ne bi došlo do utjecaja na mjerenu veličinu. Zatim je potrebno taj otpor pretvoriti u napon ili struju ili pak pomoću digitalnog sklopovlja (uz pomoć analogno-digitalnog pretvarača) u podatak. Također, moguće je da se umjesto element koji iznos vlage pretvara u iznos otpora koristi element koji iznos vlage u zraku pretvara u električni kapacitet, te se zatim iznos kapaciteta preračunava u iznos vlage.

Radi jednostavnosti mjerenje vlage se izvodi pomoću DHT22 sonde koja osim mjerenja temperature, može mjeriti i vlagu, a princip mjerenja temperature se izvori pomoću temistora, dok se mjerenje vlage izvodi na prethodno naveden način, pomoću elementa koji pretvara iznos vlage u iznos električnog kapaciteta. Prednost ovog načina jest taj da se pomoću jednog utrošenog izvoda na mikroupravljaču može istovremeno mjeriti i temperatura i vlaga, što pojednostavljuje sklopovlje. Ujedno, sonda u sebi ima sve potrebno sklopovlje koje je već tvornički kalibrirano da daje zadanu točnost. Osnovni podaci o sondi su vidljivi na tablici 2.1..

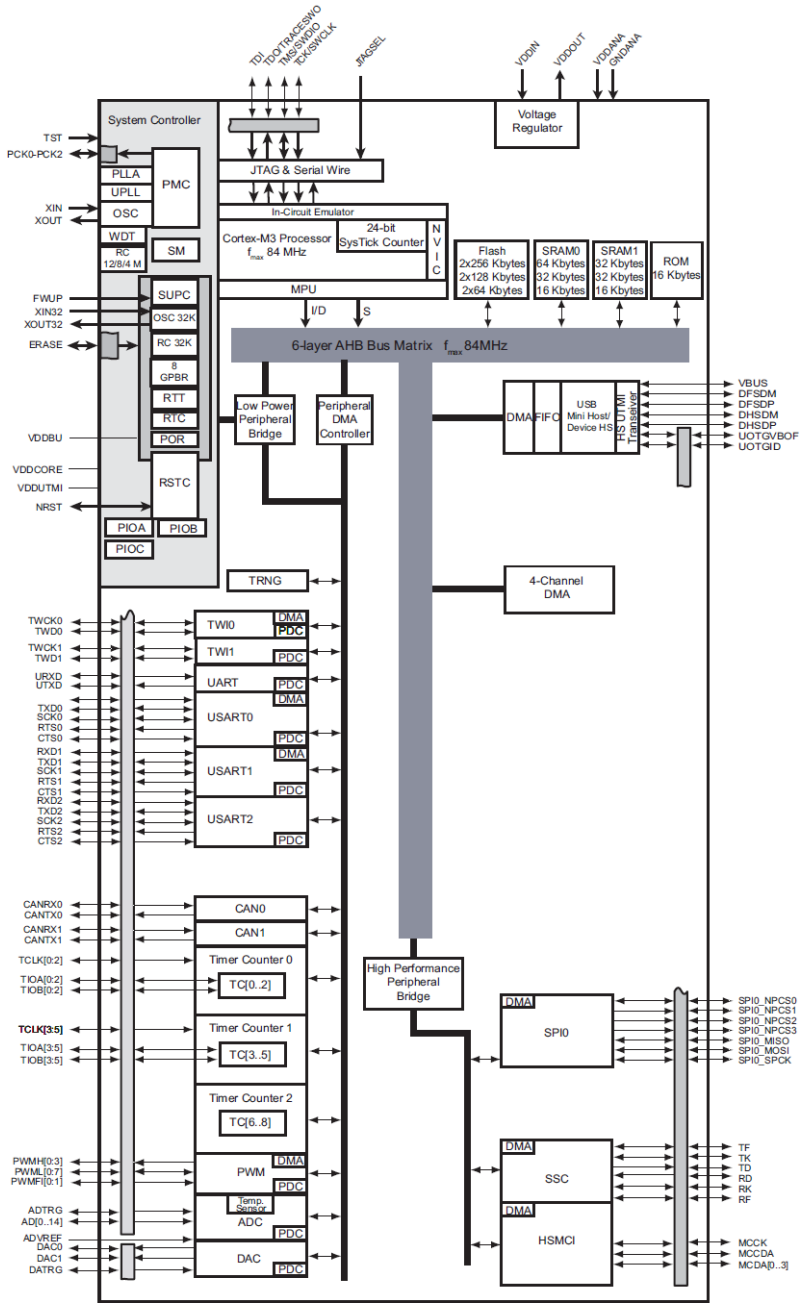
Tablica 2.1. Osnovni tehnički podaci o sondi DHT22 [15]

Model	AM2302
Napon napajanja	3.3 V – 5.5 V DC
Izlazni signal	Digitalni signal preko jednožične sabirnice
Osjetni element	Polimer kondenzator osjetljiv na vlagu
Radni opseg	Vlaga 0-100% RH (relativne vlage zraka); Temperatura -40 ~ 80 °C
Točnost	Vlaga ±2% RH (Max ±5 % RH); Temperatura ±0.5 °C
Rezolucija ili osjetljivost	Vlaga 0.1 % RH; Temperatura 0.1 °C
Ponovljivost	Vlaga ±1 % RH; Temperatura ±0.2 °C
Histereza vlage	±0.3 % RH
Dugoročna stabilnost	±0.5 % RH / godišnje
Zamjenjivosti	Potpuno zamjenjiv

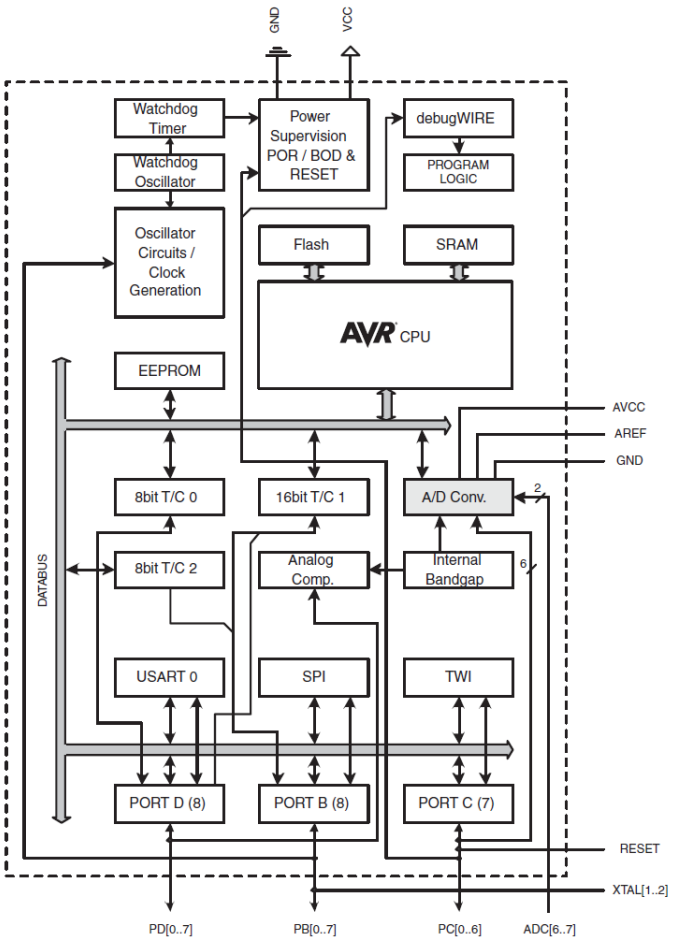
2.4. Mikroupravljači

Pošto je potrebno da sustav bude samostalan, odnosno da korisnik ne treba podešavati nikakve napredne parametre, a da rezultati koje dobije budu njemu jasni, krajnji i točni, te je isto tako potrebo da sustav bude malih dimenzija i da se napaja pomoću baterije, najoptimalnije rješenje je da se kao glavna jedinica koja komunicira s senzorima i s radio prijamnikom senzora (odašiljača) i glavne jedinice (prijamnika) koristi mikroupravljač.

Mikroupravljač je vrlo malo, jednostavno građeno računalo. Sastoji se od centralne procesorske jedinice (engl. *CPU, Central Processing Unit*), koja u sebi sadrži upravljačku jedinicu (engl. *Control Unit*), memoriju i aritmetičko-logičku jedinicu (engl. *ALU, Artihmetic-Logic Unit*). [16] ALU jedinica omogućava da pomoću mikroupravljača možemo izvršavati zbrajanje, oduzimanje, pomak ulijevo (engl. *Shift Left*), pomak udesno (engl. *Shift Right*), komparacija i druge operacije. Osim toga, mikroupravljač u sebi još ima ugrađene razne registre, koji omogućavaju mikroupravljaču neke od operacija nad podacima, UART (eng. *Universal asynchronous receiver-transmitter*) komunikaciju, SPI (eng. Serial Peripheral Interface) komunikaciju, programsku memoriju, RAM (eng. *Random Access Memory*) memoriju, EEPROM (eng. *Electricaly Erasable Read Only Memory*) memoriju. Moderni mikroupravljači, koji su novijeg datuma u sebi još mogu sadržavati i razne registre s kojima mogu izvršiti komunikaciju sa nekim od vanjskih jedinica kao što je USB (eng. *Universal Serial Bus*) komunikacija, te mogu sadržavati *timere*, *interrupte*, DMA (engl. *direct memory access*), analogno-digitalni i digitalno-analogni pretvarač i sl. Mikroupravljač je ustvari mikroprocesor zajedno s dodatnim ulazno-izlaznim jedinicama. Na slici 2.11. je vidljiva arhitektura modernijeg mikroupravljača, dok na slici 2.12. je vidljiva arhitektura mikroupravljača korištenog u ovome radu.



Slika 2.11. Arhitektura mikroupravljača ATMEL SAM3X8E (ARM Arhitektura) [18]



Slika 2.12. Arhitektura mikroupravljača ATMEGA328P (Advanced RISC arhitektura) [17]

Postoji više vrsta arhitekture mikroupravljača, a najčešće korištene su RISC (eng. *Reduced Instruction Set Computer*) i Harvard. Mikroupravljač korišten u ovom radu koristi RISC arhitekturu. Princip rada takve arhitekture je taj da se u memoriji nalaze sve instrukcije zajedno s svim potrebnim podacima (početne vrijednosti varijabli, korisnički podaci, i dr.). Zatim se ti podaci šalju preko podatkovane sabirnice na dekoder instrukcija (instrukcijski dekoder, engl. *Instruction Decoder*) i zatim se pomoću njega postavljaju registri za primitak podataka. Nakon toga što ta instrukcija bude gotova, vrijednost programskog brojača (engl. *Program Counter*) se povećava ja jedan, što ukazuje da se čita vrijednost sa slijedeće memorijske adrese na kojoj se nalazi nova instrukcija i tako se cijeli proces ponavlja. Radio toga što ATMEGA328P ima registre koji omogućavaju SPI i UART komunikaciju, koja će omogućavati komunikaciju s radio modulom baziranom na nRF24L01 integriranom krugu i s Bluetooth™ modulom, ima 32 KByte programske memorije, 2 KByte RAM memorije i 1 KByte EEPROM memorije, mogućnost

režima rada niske potrošnje, analogno-digitalni pretvarač za mjerenje napona baterije, a ujedno ima cijelu gotovu platformu preko kojeg se može programirati, on je odabran za ovaj rad.

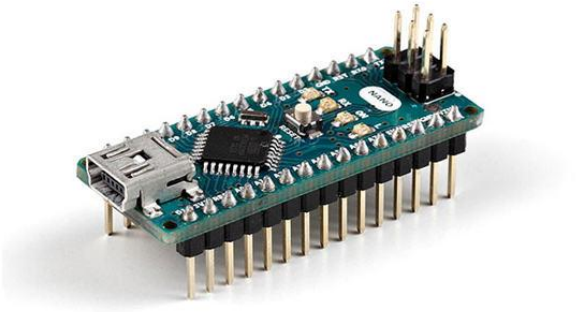
2.5. Arduino platforma

Arduino je razvojno sučelje osmišljeno radi pojednostavljenog programiranja ATMEL mikroupravljača. Bazirano je na Java™ programskoj podršci. Sama Arduino platforma je *Open Source* odnosno, platforma otvorenog koda kako što se tiče *softwarea* tako i *hardwarea*. Početak samog Arduino projekta je 2003. godina kada je grupa, tada studenata, željela osmisliti jeftinu i jednostavnu platformu za mikroupravljače. Željeli su osmisliti software-sku podršku pomoću koje je moguće pisati program za mikroupravljač, program za programiranje i *debugging,* te ujedno i hardware-sku podršku koja bi se sastojala od samog fizičkog programera i pločice s mikroupravljačem. Prvi Arduino je Arduino RS232 koji se programirao preko RS-232 računalnog sučelja, nakon čega je slijedio Arduino Diecimila koji je imao USB programator integriran na pločici. Sam princip programiranja Arduina Diecimila je identičan onome koji se koriti danas, a to je da se na pločici nalazi USB to UART pretvarač koji je spojen na mikroupravljač. Mikroupravljač koji se koristi na pločici mora na sebi imati već prethodno isprogramiran engl. *Bootloader*, odnosno dio koda koji će primati podatke s UART-a i na taj način će programirati mikroupravljač. U suštini, on na taj način programira samog sebe. Na kraju procesa programiranja, mikroupravljač se ponovno pokrene (engl. *Reset*) i kreće izvoditi program koji mu je zadan.

Mikroupravljač koji je korišten na prvoj Arduino pločici je ATMEGA8-16P koji je bio tada jedan od jačih, lakše nabavljivih i cijenom prihvatljivim mikroupravljačem. Danas postoji više Arduino pločica, a neke od njih su Uno, Nano, Micro koje su bazirane na ATMEGA328P mikroupravljaču, Arduino Mega koji je baziran na ATMEGA1280 ili ATMEGA2560 mikroupravljaču.

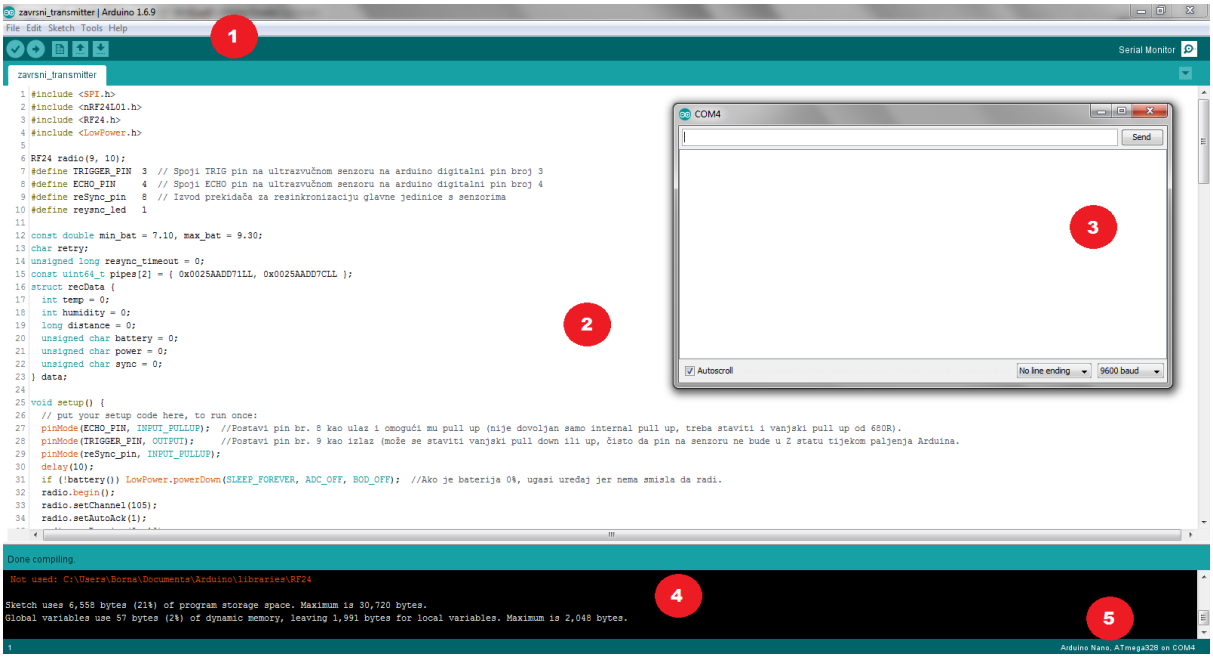
Što se tiče programske podrške, ona se sastoji od Arduino IDE razvojnog okruženja, koje se sastoji od *editora,* programera i *debuggera* (*Serial Monitora* preko kojeg korisnik može ispisivati razne podatke od strane mikroupravljača na svoje računalo i na taj način može pratiti tok programa ili pak stanja pojedinih parametara i varijabli).

Slika 2.13. prikazuje Arduino Nano razvojnu pločicu, koja je korištena u ovome radu radi svojih malih dimenzija i niske potrošnje.



Slika 2.13. Izgled Arduino Nano razvojne pločice [19]

Na slici 2.14. vidljiv je izgled Arduino IDE razvojnog okruženja.



Slika 2.14. Izgled Arduino IDE razvojnog okruženja

Arduino IDE razvojno okruženje sastoji se od:

1. Izborničke trake – Pomoću nje možemo otvarati izbornike u kojima možemo odabirati model ploče, *port* za programiranje, podešavati postavke okruženja, podešavati postavke programatora, način programiranja, izvršavati *Autoformat* i druge postavke.

2. *Editor* – Prostor koji je namijenjen za pisanje programskog koda koji želimo da mikroupravljač izvršava. S lijeve strane se nalaze redni broj reda, a ukoliko dođe do sintaktičke greške tijekom kompajliranja programa, kompajler taj red u editoru označi crvenom bojim. Programski jezik koji se koristi za programiranje Arduino pločice je C/C++.

3. *Serial Monitor* – Prozor koji se aktivira pritiskom na gumb koji se nalazi u gornjem desnom kutu. U njemu se ispisuju podaci koje mikroupravljač (Arduino pločica) šalje računalu putem UART sučelja. Za uspješnu komunikaciju između računala i Arduino pločice potrebno je dobro podesiti Baud Rate. Predefinirana vrijednost je 9600.

4. Informacijska traka – Prikazuje što se trenutno događa, da li je kompajliranje uspješno ili je naišlo na grešku, te ujedno ispisuje što je greška. Ukoliko je sve prošlo bez greške, iznad nje se ispisuje *Compile Complete* ili *Upload Complete*.

5. Traka koja prikazuje trenutno odabranu Arduino pločicu i trenutno odabran port za programiranje.

Kada se pokrene sučelje, ukoliko nije otvoren novi, prazni dokument u *editoru,* potrebno je kliknuti na File > New. Nakon toga se otvara novi prozor koji u *editoru* ima dvije bitne funkcije; *void setup* i *void loop*. *Void setup* je funkcija u koju mikroupravljač ulazi nakon što se pokrene. Ta funkcija se izvršava samo jednom i to prilikom pokretanja, stoga je korisna ukoliko trebamo podesiti neke parametre programa prije nego sam program započne.

Void loop je funkcija u koju upisujemo program koji želimo da se obavlja. Taj program će se ponavljati kada dođe do kraja te funkcije. Dakle, nakon izvršenja zadnje funkcije u *void loop* funkciji, program se vraća na početak te iste *void loop* funkcije. Bez te dvije osnovne funkcije, nije moguće isprogramirati Arduino pločicu, stoga svaki program mora imati te dvije funkcije.

Zatim, potrebno je odabrati odgovarajuću Arduino pločicu koju želimo programirati. To se izvodi na način da se na izborničkoj traci odabere Tools > Bord te se zatim odabere pločica.

Nakon toga se priključuje Arduino pločica na računalo i odabire *port* na način da se odabere Tools > Port.

Kada je sve to podešeno, može se početi s pisanjem programa koje se izvodi u *editoru*. Nakon što je kod odnosno program napisan, potrebno ga je kompajlirati, što se izvodi na način da se u izborničkoj traci odabere Sketch > Verify/Complie ili se klikne na kvačicu koja se nalazi ispod izborničke trake u gornjem lijevom kutu. Ako sve prođe bez greške, potrebno je program poslati na Arduino pločicu, što se izvodi na način da se iz izborničke trake odabere Sketch > Upload ili se klikne na gumb u obliku strelice usmjerene u desnu stranu koja se nalazi ispod izborničke trake u gornjem lijevom kutu. Ako je sve bilo uspješno, program na mikroupravljaču će se početi izvoditi, a na informacijskoj traci će biti ispisano *Upload Complete*.

3. IZRADA SHEME I PROGRAMSKE PODRŠKE

3.1. Izrada sheme

Cijeli rad će se sastojati od 3 dijela; dva senzora koji mjere udaljenost, te temperaturu i vlagu zraka (odašiljači) i od glavne jedinice s Bluetooth™ modulom (prijamnik). Sva tri dijela će u sebi imati Arduino Nano s ATMEGA328P mikroupravljačem. Osim toga, senzori će sadržati ultrazvučni senzor udaljenosti, DHT22 senzor vlage i temperature zraka. Također, sadržavat' će i nRF24L01(+) radio primo-predajući modul koji radi na 2.4 GHz.

Pošto je napon napajanja tog modula 3.3 V istosmjerno, potrebno je spustiti napon baterije od 7 V (koji dolazi iz DC-DC pretvarača) na odgovarajuću razinu. U tu svrhu koristi se AMS1119-3V3 linearni regulator napona (engl. *LDO Voltage regulator*). Prednost korištenja linearnog regulatora napona umjesto regulatora napona koji radi u prekidačkom režimu rada je to što neće stvarati smetnje i šumove u napajanju, što je iznimno važno za radio modul. Nedostatak je taj što mu je faktor iskorištenja lošiji nego kod regulatora koji radi u prekidačkom režimu rada.

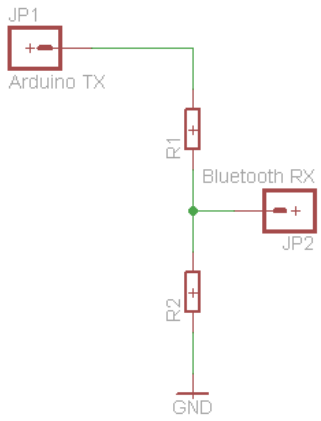
Također, za pravilan rad potrebni su kondenzatori u napajanju koji filtriraju napon napajanja i uklanjaju neželjene šumove visoke frekvencije. Osim toga, senzorska jedinica (odašiljač) na sebi će sadržavati i prekidač koji će služiti za aktivaciju sinkronizacije s glavnom jedinicom.

Sinkronizacija radi na način da senzorske jedinice (odašiljači) čekaju da dobiju podatak od glavne jedinice (prijamnik), te zatim počinju mjeriti i slati podatke. Ukoliko se nakon dvije minute ne primi paket s sinkronizacijskim podatkom, sinkronizacija se poništava i senzorske jedinice ulaze u režim rada niske potrošnje energije, gdje se gasi radio primo-predajnik. Da bi se radio primopredajnik ponovno aktivirao i omogućila sinkronizacija, potrebno je pritisnuti tipkalo i držati ga dok se svjetleća dioda ne upali.

U prilogu, moguće je naći shemu senzorske jedinice (Prilog 1).

Kada je u pitanju glavna jedinica, ona se sastoji od radio primo-predajničkog modula koji u sebi ima integrirano pojačalo snage i niskošumno pojačalo. To omogućava glavnoj jedinici (prijamniku) veću osjetljivost, a ujedno i veću snagu odašiljanja. Također i ova jedinica ima na sebi linearni regulator napona na 3.3 V istosmjerno, kao i kondenzatore na filtriranje napona napajanja i uklanjanje šumova visokih frekvencija, no također sadrži i Bluetooth™ modul, koji omogućava komunikaciju s pametnim telefonom, te *buzzer* zvučnik.

Bluetooth™ modul radi na naponu napajanja od 5 V, no amplituda signala koji ulaze u njega ne smije biti veća od 3.3 V jer bi moglo doći do oštećenja. Stoga potrebno je napraviti tzv. *Level Converter* (pretvarač naponskih razina), odnosno elektronički sklop koji će signal amplitude od 5 V, pretvoriti u signal amplitude od 3.3 V. To je bitno samo od RX (eng. *Receive*, prijamna linija) izvoda na Bluetooth™ modulu. Takav sklop je najjednostavnije izvesti pomoću dva otpornika koji čine djelitelj napona.



Slika 3.1. *Level Converter* za Bluetooth™ modul

Potrebno je proračunati vrijednosti otpornika. Potrebno je odabrati jedan fiksan otpornik, što će u ovom slučaju biti R_2 te će iznositi 10 kΩ. Iznos drugog otpornika će se dobiti računskim putem.

Napon na otporniku R_2 treba biti 3.3 V, stoga se može izračunati struja kroz taj otpornik:

$$I = \frac{U_{R_2}}{R_2} = \frac{3.3}{10000} = 0.33 \text{ mA} \tag{3-1}$$

Ta ista struja prolazi i kroz R_1, a napon na njemu je

$$UTX - UR2 = 5 - 3.3 = 1.7 \text{ V}. \tag{3-2}$$

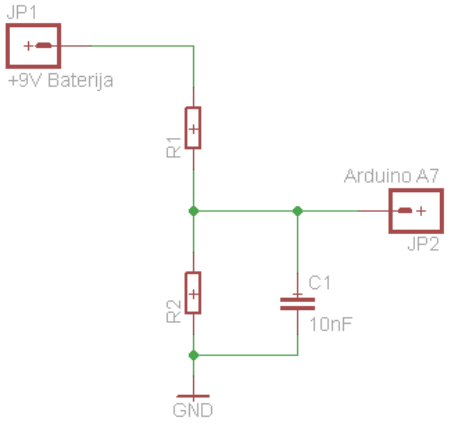
Stoga je moguće izračunati njegov otpor kao:

$$R_1 = \frac{U_{R_1}}{I} = \frac{1.7}{0.33 * 10^{-3}} = 5151.51 \ \Omega \tag{3-3}$$

Najbliža vrijednost otpornika je 5.6 kΩ. Dakle, za R_1 iznosi 5.6 kΩ.

Cijela shema glavne jedinice (odašiljača) može se pronaći u prilogu (Prilog 2), a izgled završenog rada u cijelosti nalazi se također u prilogu (Prilog 6).

Također, svaki od modula ima mogućnost mjerenja napona baterije. Napon baterije, s kojom se sklopovi napajaju iznosi 3.7 V. A/D pretvarač, koji se nalazi u mikroupravljaču, može mjeriti napone do 1.1 V, stoga je potrebno napon baterije spustiti na odgovarajuću naponsku razinu. To se također može izvesti pomoću djelitelja napona uz to što mu se paralelno otporniku s kojim se mjeri napon dodaje kondenzator koji filtrira visoke frekvencije, te omogućava stabilno i mirno čitanje napona. Odabrano je da najveći dopušteni napon koji se može mjeriti bude 11 V, da korisnik cijeli uređaj može napajati pomoću 9 V „blok“ baterije, ukoliko je potrebno.



Slika 3.2. Djelitelj napona za A/D pretvornik

Odabiremo jedan fiksan otpornik, u ovom slučaju R_2 koji iznosi 100 kΩ. Pošto je poznato da na njemu smije biti napon od najviše 1.1 V, možemo izračunati struju kroz njega.

$$I = \frac{U_{R_2}}{R_2} = \frac{1.1}{10000} = 110\ \mu\text{A} \tag{3-4}$$

Ta ista struja teče i kroz otpornik R_1, pad napona na njemu se dobije kao

$$U_{R1} = U_{Bat} - U_{R1} = 11 - 1.1 = 9.9\ \text{V} \tag{3-5}$$

Sada se zna napon i struja na njemu, stoga je moguće izračunati njegov otpor

$$R_1 = \frac{U_{R_1}}{I} = \frac{9.9}{110 * 10^{-6}} = 90\ \text{k}\Omega \tag{3-6}$$

Najbliži otpornik tome je od 100 kΩ, stoga se odabire taj otpornik kao R_1.

Potrebno je još izračunati omjer ulaznog i izlaznog napona, jer će to biti potrebno kasnije u programu ra izračunavanje

$$U_{\text{Omjer}} = \frac{U_{\text{Bat}}}{U_{R_2}} = \frac{U_{R_1} + U_{R_2}}{U_{R_2}} = \frac{U_{R_1}}{U_{R_2}} + \frac{U_{R_2}}{U_{R_2}} = 1 + \frac{U_{R_1}}{U_{R_2}} = 1 + \frac{I * R_1}{I * R_2} = 1 + \frac{R_1}{R_2} = 1 + \frac{100}{10} = 11 \tag{3-7}$$

Važno je napomenuti da je kao napon baterije uzet napon od 11 V. Razlog tomu je što napon pune baterije od 9 V, koja je neopterećena, može doseći više od 10 V. Ukoliko se napaja pomoću litiji ion baterije, napon prazne baterije iznosi 2.8 V, dok napon pune 4.2 V.

3.2. Izrada programa

Program za mikroupravljač je pisan u C / C++ programskome jeziku, koristeći Arduino IDE razvojno okruženje. Svaka jedinica ima svoj program, no senzorska jedinica (odašiljači) imaju identičan program, te je jedina razlika u adresi, tako da se ta dva odašiljača mogu razlikovati. Pri izradi programa korištene su biblioteke, koje su prethodno nađene na internetu. U pitanju su slijedeće biblioteke:

- NRF24L01.h i RF24.h – Biblioteke koje omogućavaju rad s nRF24L01(+) radio modulom [21]
- LowPower.h – Biblioteka koja omogućuje da mikroupravljač radi u režimu rada niske potrošnje [23]
- DHT.h – Biblioteka koja omogućava komunikaciju s DHT22 senzorom vlage i temperature [22]

U slijedećem dijelu, biti će opisani važniji dijelovi koda glavne jedinice (prijamnika), a potpuni kod, zajedno s komentarima može se pronaći u prilogu (Prilog 3). Također tijek programa (engl. *flow chart*) nalazi se u prilogu (Prilog 5).

Nakon što se uveze biblioteka u program potrebno je definirati izvode koji će se koristiti za komunikaciju s nRF24L01 radio modulom i za UART komunikaciju, odrediti adrese komunikacije, te odrediti izgled paketa.

```
RF24 radio(9,10);
SoftwareSerial serial(4, 5);
const uint64_t pipes[2] = { 0x0025AADD71LL, 0x0025AADD7CLL };
struct recData {
  int temp;
  int vlaga;
  long udaljenost;
  unsigned char baterija;
  unsigned char sinkronizacija;
} paket;
```

Programski kod 3.1. Postavke izvoda, adrese i paketa

Nakon toga, potrebno je podesiti radio primo-predajnik. Potrebno mu je podesiti postavke kanala, automatske potvrde primitka, veličinu paketa, adrese i adrese pojedinog komunikacijskog kanala, kao i retransmisiju:

```
radio.begin();
radio.setChannel(105);
radio.setAutoAck(1);
radio.setRetries(2, 10);
radio.setPayloadSize(sizeof(recData));
radio.openWritingPipe(pipes[1
radio.openReadingPipe(1, pipes[0]);
radio.openReadingPipe(2, pipes[1]);
```

Programski kod 3.2. Postavke radio primo-predajnika

Nadalje, potrebno je otkriti da li je pristigao kakav paket, te analizirati taj pristigli paket podataka. Ako je izmjerena udaljenost izmjerena od strane senzora manja od one koja je podešena kao okidna udaljenost, treba vidjeti koji ju je senzor aktivirao, te treba vidjeti da li prepreka stoji, približava se ili se udaljava.

```
if (radio.available(&rxPipeNo))
  {
    proximityAlarm = 0;
    while (radio.available())
    {
      radio.read(&paket, sizeof(recData));
    }
    distance[rxPipeNo - 1] = double(paket.udaljenost) / 100;
    bat[rxPipeNo - 1] = paket.baterija;
    temperature[rxPipeNo - 1] = float(paket.temp) / 10;
    humidity[rxPipeNo - 1] = float(paket.vlaga) / 10;

    if (alarmDistance > distance[rxPipeNo - 1] && distance[rxPipeNo - 1] >
0) {
      proximityAlarm = rxPipeNo;
      if (distanceOld[rxPipeNo - 1] - distance[rxPipeNo - 1] > 2)
      {
        pokret = 1;
      } else if (distanceOld[rxPipeNo - 1] - distance[rxPipeNo - 1] < -2)
{
        pokret = 3;
      } else {
        pokret = 2;
      }
      distanceOld[rxPipeNo - 1] = distance[rxPipeNo - 1];
    }
 }
```

Programski kod 3.3. Prijam i analiza pristiglog paketa (vezano za detekciju udaljenosti)

Ukoliko postoji prepreka na udaljenosti manjoj od okidne, potrebno je te podatke reprezentirati korisniku preko Bluetooth™ veze i zvučnog signala, ako ga je korisnik omogućio.

```
if (proximityAlarm != 0 && (unsigned long)(millis() - alarmTimeout) > 500) {
   alarmTimeout = millis();
   serial.print(F("UPOZORENJE - Senzor "));
   serial.println(proximityAlarm, DEC);
   serial.print(F("Prepreka "));
   switch (pokret) {
     case 1:
       serial.print(F("se priblizava"));
       break;
     case 2:
       serial.print(F("stoji"));
       break;
     case 3:
       serial.print(F("se udaljava"));
       break;
   }
   serial.print(F(" - "));
   serial.print(distance[proximityAlarm - 1], 2);
   serial.println(F("cm"));
   if (buzzer) tone(6, 750 + (250 * (4 - pokret)), 50);
   proximityAlarm = 0;
 }
```

Programski kod 3.4. Reprezentacija podataka o udaljenosti prepreke korisniku

Također, korisnik može podešavati neke od postavki slanjem određenih kodova preko Bluetooth™ veze u obliku ASCII koda. Svaka komanda započinje s „*“, a završava s „#“. Tako postoje slijedeće komande:

- *B000# / *B111# - Onemogući / omogući zvučni signal

- *DXXX# - Podešavanje okidne udaljenosti detekcije prepreke, gdje XXX označava željenu udaljenost detekcije prepreke u centimetrima
- *R1111# - Slanje resinkronizacijskog signala svim senzorima (odašiljačima)

Nadalje, slijede neki bitni dijelovi koda za senzorsku jedinicu (odašiljač). Cijeli kod, zajedno s komentarima, nalazi se u prilogu (Prilog 4).

Nakon uvoza biblioteka, potrebo je nadjenuti nazive pojedinim izvodima sa mikroupravljača, radi lakšeg praćenja i programiranja koda.

```
RF24 radio(9, 10);
#define OKID_PIN  3
#define ECHO_PIN  4
#define DHTPIN 2
#define DHTTYPE DHT22
#define reSync_pin   8
#define reysnc_led   1
DHT dht(DHTPIN, DHTTYPE);
```

Programski kod 3.5. Imenovanje i definiranje izvoda za komunikaciju s senzorima i radio modulom

Nakon toga potrebno je odrediti adrese za komunikaciju. Oba senzora koriste isti program, jedina razlika je u rednome broju senzora. U slučaju da se radi o senzoru 1, tada bi varijabla redniBrojSenzora imala vrijednost jedan.

```
const uint64_t pipes[2] = { 0x0025AADD71LL, 0x0025AADD7CLL };
const char redniBrojSenzora = 2;
```

Programski kod 3.6. Definiranje adresa za komunikaciju i rednog broja pojedinog senzora

Mjerenje temperature i vlage se vrši na slijedeći način.

```
int vlaga_ = int(dht.readHumidity() * 10);
int temperatura_ = int(dht.readTemperature() * 10);
if (isnan(vlaga_) || isnan(temperatura_))
{
  paket.vlaga = 0;
  paket.temp = 0;
  return 0;
} else {
  paket.vlaga = vlaga_;
  paket.temp = temperatura_;
  return 1;
```

Programski kod 3.7. Mjerenje temperature i vlage pomoću DHT22 senzora

Slijedeće, potrebno je izvesti mjerenje udaljenosti pomoću SRF-05 ultrazvučnog senzora za udaljenost. Naredni kod opisuje kako se to izvodi.

```
unsigned long pocetno_vrijeme, zavrsno_vrijeme;
char greska = 0;
digitalWrite(OKID_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(OKID_PIN, LOW);
unsigned long timeout = millis();
do {
  if ((unsigned long)(millis() - timeout) > 30) {
    greska = 1;
    break;
  }
}
while (!digitalRead(ECHO_PIN));
pocetno_vrijeme = micros();
timeout = millis();
if (greska) return 0;
do {
  if ((unsigned long)(millis() - timeout) > 30) {
    greska = 1;
    break;
  }
}
while (digitalRead(ECHO_PIN));
zavrsno_vrijeme = micros();
if (greska) return 0;
double distance = (double(zavrsno_vrijeme - pocetno_vrijeme) * 0.017);
return (distance);
```

Programski kod 3.8. Mjerenje udaljenosti pomoću SRF-05 ultrazvučnog senzora

Nakon što su podaci prikupljeni, slanje paketa glavnoj jedinici vrši se na način da se prvo provjerava da li je linija, odnosno veza slobodna tako da se mjeri kolika je snaga signala nosioca na tom kanalu. Ako je preslab ili ga nema, paket se šalje.

```
radio.powerUp();
radio.startListening();
while (radio.testCarrier()) {
  delay(random(100));
}
radio.stopListening();
ok = radio.write(&paket, sizeof(recData));
```

Programski kod 3.9. Slanje paketa glavnoj jedinici

3.3. Napajanje uređaja i izbor baterije

Jedan od uvjeta je da uređaj (sklop) bude prijenosan, što znači da je potrebno da se napaja baterijom. Neki od uvjeta koji se stavljaju na izbor baterije su slijedeći:

- veliki kapacitet
- male dimenzije
- dovoljno visok radni napon
- baterija mora biti punjiva

Iz slijedećih uvjeta, mogu se pronaći baterije s različitim tehnologijama, no niti jedna je odgovara na sve uvjete. Ukoliko se uzme NiMH (eng. *Nickel–metal hydride battery*) tehnologija, odgovarat' će radni napon (9 V „Blok“ NiMH baterija), punjiva je, malih dimenzija, no ne odgovara kapacitetom, pošto su takve baterije kapaciteta reda 200 mAh. Ukoliko se uzme prosječna potrošnja sklopa kao

$$I_{sr} = \frac{(I_{rada} * t_{rada}) + (I_{mirovanja} * t_{mirovanja})}{t_{rada} + t_{mirovanja}} = \frac{(40mA * 50ms) + (15mA * 250ms)}{50ms + 250mS} = 19.16667\ mA \quad (3\text{-}8)$$

Kapacitet baterije se računa kao umnožak struje koja teče kroz bateriju i vremenskog intervala u kojem se to događa.

$$C_{baterije} = I_{trošenja} * t_{trošenja} \quad (3\text{-}9)$$

Pošto bateriju nikada nije poželjno isprazniti do kraja, nego joj ostaviti dio kapaciteta, ne možemo iskoristiti sav njen kapacitet, već samo dio, praktično oko 80 % njenog kapaciteta se može iskoristiti.

$$C'_{baterije} = C_{baterije} * 0.8 \quad (3\text{-}10)$$

Stoga

$$C_{baterije} * 0.8 = I_{trošenja} * t_{trošenja} \rightarrow t_{trošenja} = \frac{C_{baterije} * 0.8}{I_{trošenja}} = \frac{200mAh * 0.8}{19.16667\ mA} = 8.34\ h \quad (3\text{-}11)$$

Dakle, s tom baterijom, u idealnom slučaju, autonomija sklopa iznosi 8 sati.

Drugi odabir su NiCD (eng. *Nickel Cadmium*) baterije. No, problem kod takvih baterija jest to što koriste kadmij, koji je toksičan, stoga su takve baterije teško dostupne.

Zadnja alternativa su Litiji Ion baterije. Imaju dosta veliki kapacitet (prosječna 18650 baterija ima 2500 mAh), relativno su malih dimenzija, punjive su, no radni napon je manji od onog što je sklopu potrebno. Sklopu je potrebno najmanje 6.5 V za rad. Taj napon se dobije tako što Arduino radi na 5 V, a prije njega se nalazi regulator (AMS1117-5.0), koji da regulira napon,

ulazni napon mora biti 1.5 V veći od izlaznog. Dakle, treba barem dvije serijski spojene baterije. No, to nije lako izvedivo, pošto dvije baterije nisu jednake, pa će se dogoditi da ni naponi na njima nisu jednaki, pa će baterije pod teretom, puniti jedna drugu, što može dovesti do toga da napon na jednoj bateriji dostigne napon veći od dozvoljenih 4.2 V. Taj problem se može riješiti korištenjem strujnog kruga za balansiranje (ujednačavanje) napona pojedine baterije, ali takvi sklopovi su teško dostupni ili su vrlo komplicirani za izradu.

U radu su korištene dvije litijeve baterije u paraleli, što povećava njihov kapacitet, dakle 5000mAh. Stoga, može se izračunati autonomija (trajanje vijeka baterije):

$$C_{baterije} * 0.8 = I_{trošenja} * t_{trošenja} \rightarrow t_{trošenja} = \frac{C_{baterije} * 0.8}{I_{trošenja}} = \frac{5000mAh * 0.8}{19.16667\,mA} = 209\,h \quad (3\text{-}12)$$

U stvarnosti, trajanje će biti nešto manje, no i dalje usporedivo veće nego kod NiMh baterija.

Drugi način kako se može riješiti problem niskog napona litiji ion baterije jest da se koristi DC-DC pretvarač (eng. *DC-DC Booster*) koji će napon s litijevih baterija povisiti na 6.5V.

Upravo na taj način je ovdje riješen problem niskog napona koji se dobiva iz litijevih baterija. U tu svrhu, korišten je MC34063 integrirani krug koji omogućuje povisivanje napona, smanjivanje napona i inverziju napona, pomoću zavojnice i diode. Za potpuni rad potrebno je izračunati vrijednosti otpornika u povratnoj vezi, otpornika koji određuje maksimalnu struju, te vrijednost zavojnice i izlaznog kondenzatora. Potrebno je da sklop za povećavanje napona daje izlazni od 7V, izlaznu struju od najviše 150 mA, radi na frekvenciji od 80 kHz, te daje šum ne veći od 40mV.

Proračun za pojedine komponente [24] (nalazi se u tehničkim specifikacijama integriranog kruga MC34063):

$$t_{on}/t_{off} = \frac{V_{out} + V_F - V_{in(\text{minium})}}{V_{in(minimun)} - V_{sat}} = \frac{7 + 0.2 - 3}{3 - 0.5} = 1.68 \quad (3\text{-}13)$$

$$t_{on} + t_{off} = \frac{1}{f} = \frac{1}{80\,kHz} = \frac{1}{80 * 10^3} = 1.25 * 10^{-5} s = 12.5\,\mu s \quad (3\text{-}14)$$

$$t_{off} = \frac{t_{on} + t_{off}}{\frac{t_{on}}{t_{off}} + 1} = \frac{12.5 * 10^{-6}}{1.68 + 1} = 4.66\ \mu s \tag{3-15}$$

$$t_{on} = \left(t_{on} + t_{off}\right) - t_{off} = 12.5\mu s - 4.66\mu s = 7.84\mu s \tag{3-16}$$

$$C_T = 4.0 * 10^{-5} * t_{on} = 4.0 * 10^{-5} * 7.84 * 10^{-6} = 313.6\ pF \tag{3-17}$$

$$\begin{aligned} I_{peak\ (switc\ h)} &= 2 * I_{out\ (\text{maximum }\ )} * \left(\frac{t_{on}}{t_{off}} + 1\right) \\ &= 2 * 150 * 10^{-3} * (1.68 + 1) = 0.804\ A \end{aligned} \tag{3-18}$$

$$R_{SC} = \frac{0.3}{I_{peak\ (switc\ h)}} = \frac{0.3}{0.804} = 0.3\ \Omega \tag{3-19}$$

$$\begin{aligned} L_{(\text{minimum }\ )} &= \left(\frac{V_{in\,(minimum\ )} - V_{sat}}{I_{peak\ (switc\ h)}}\right) * t_{on\,(\text{maximum }\ )} \\ &= \left(\frac{3 - 0.5}{0.804}\right) * 7.84 * 10^{-6} = 20\ \mu H \end{aligned} \tag{3-20}$$

$$C_O = 9 * \frac{I_{out} * t_{on}}{V_{ripple}} = 9 * \frac{0.15 * 7.84 * 10^{-6}}{0.04} = 280\ \mu F \tag{3-21}$$

$L_{minimum}$ predstavlja zavojnicu koja će se upotrijebiti, odnosno njezin minimalni induktivitet, C_O predstavlja kondenzator na izlazu, a C_T kondenzator koji određuje frekvenciju sklopa (spojen je na izvod broj 3 integriranog kruga MC34063). Potrebno je još izračunati otpornike u povratnoj vezi, koji se računaju kao

$$|V_{out}| = 1.25 * \frac{R_2}{R_1} \tag{3-22}$$

odnosno, ako se traži omjer otpornika

$$\frac{R_2}{R_1} = \frac{|V_{out}|}{1.25} = \frac{7}{1.25} = 5.6 \tag{3-23}$$

Uz fiksiran otpornik R_2 = 10000 Ω, R_1 dobije se kao

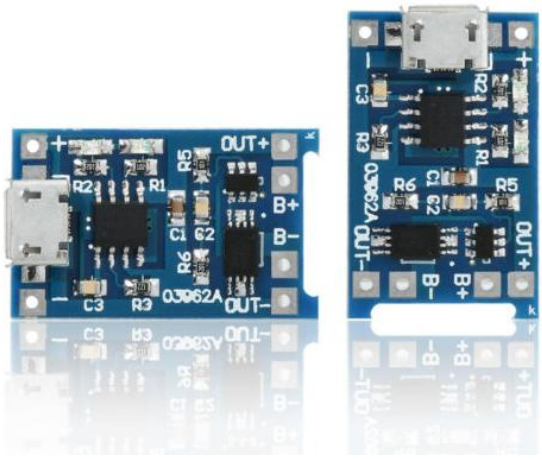
$$R_1 = \frac{R_2}{5.6} = \frac{10000}{5.6} = 1785\ \Omega \approx 2200\ \Omega \tag{3-24}$$

Odabran je najbliži slijedeći otpornik, a to je otpornik od 2.2 kΩ. Rezultat toga će biti nešto manje različitiji napon on od onoga što je odabrano, no pošto nije potrebno precizno napajanje od 7V, može se uzeti slijedeći najbliži otpornik. Isto vrijedi i za zavojnicu, odabrana je on 10μH,

jer je slijedeća najbliža, najčešće korištena, zavojnica. Iako je po proračunu izlazni kondenzator 280 μF, a njemu najbliži 330 μF, odabran je 470 μF, jer se eksperimentalno pokazalo da 330 μF i dalje ima veliki šum. No, te vrijednosti nisu kritične, stoga, ukoliko se želi bolja filtracija s manje šuma, mogu se odabrati i veći iznosi kapaciteta kondenzatora.

Shema završenog sklopa DC-DC pretvarača, kao i njegov nacrt tiskane pločice nalazi se u prilogu (prilog 7).

Pošto su baterije punjive, potrebno je izraditi sklop koji će puniti litij ion baterije. Sklop je u obliku gotovog modula koji se može naći na tržištu, a baziran je na TP4056 integriranom krugu. On omogućuje punjenje litiji ion baterije, pritom, kontrolirajući napon i struju u različitim fazama punjenja baterije, kao i prikaz stadija punjenja baterije. Također, modul na sebi sadrži DW01 integrirani krug koji služi za praćenje napona na bateriji i struju kroz bateriju i štiti bateriju od kratkog spoja, preopterećenja, preniskog ili previsokog napona.

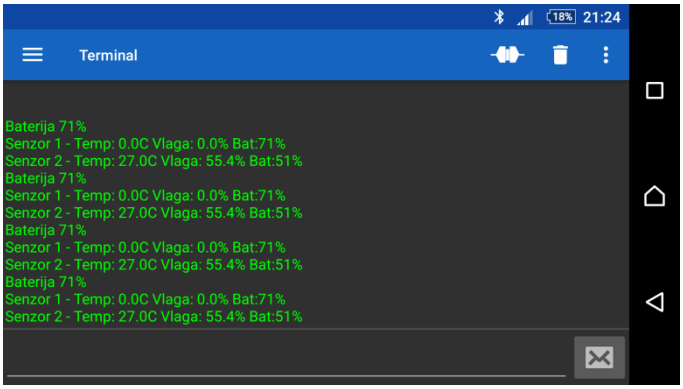


Slika 3.3. Modul za punjenje litiji ion baterija s zaštitom baziran na TC4056 i DW01 integriranom krugu [25]

4. PRIKAZ PRAĆENJA RADA ZAVRŠNOG RADA

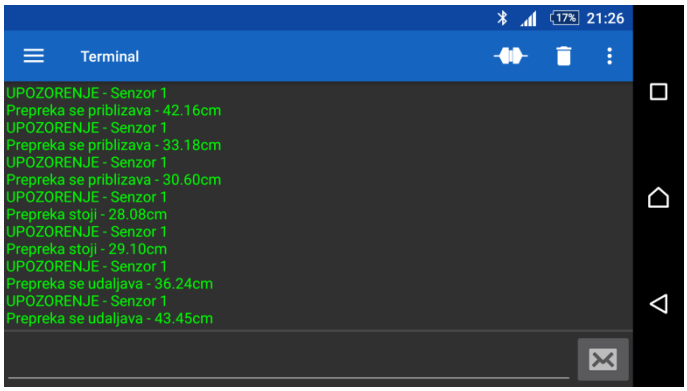
U daljnjem dijelu rada biti će prikazane sve opcije i mogućnosti glavne jedinice i senzorskih jedinica.

Prije svega, potrebo je otići na trgovinu aplikacija na pametnom telefonu i instalirati aplikaciju pod nazivom „Serial Bluetooth“. Zatim, potrebno je priključiti napajanje na način da se sama litiji ion baterija spoji s modulom za punjenje i zaštitu litiji ion baterije, zatim se taj modul spoji na sklop za povećanje napona napajanja, a izlaz iz njega se spaja u jedinice (senzorske jedinice, te zatim glavna jedinica). Taj postupak je potrebno samo učiniti tijekom prvog korištenja. Nakon toga, pokrenuti prethodno instaliranu aplikaciju, te upariti Bluetooth™ uređaj s pametnim telefonom, to se radi na način da se odabire gumb s tri vodoravne linije koji se nalazi u gornjem lijevom kutu, te odabere *Bluetooth devices* i odabere Bluetooth™ uređaj. Naziv mu može biti „Arduino“, „HC-05“ ili „HC-06“, a lozinka „0000“ ili „1234“. Odabrati gumb za spajanje koji se nalazi gore, prvi po redu. Nakon toga se počinju prikazivati podaci s glavne jedinice u prozoru.



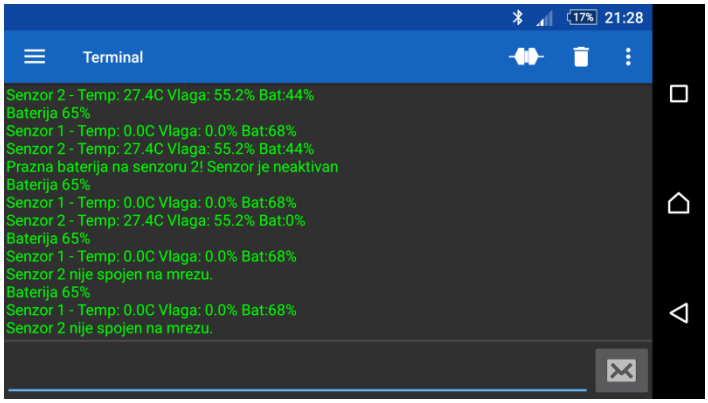
Slika 4.1. Prikaz podataka kada nema nikakve prepreke

Klasični prikaz prikazuje podatke o trenutnoj razini baterije glavne jedinice, temperaturu, vlagu i razinu baterije svakog od senzora.



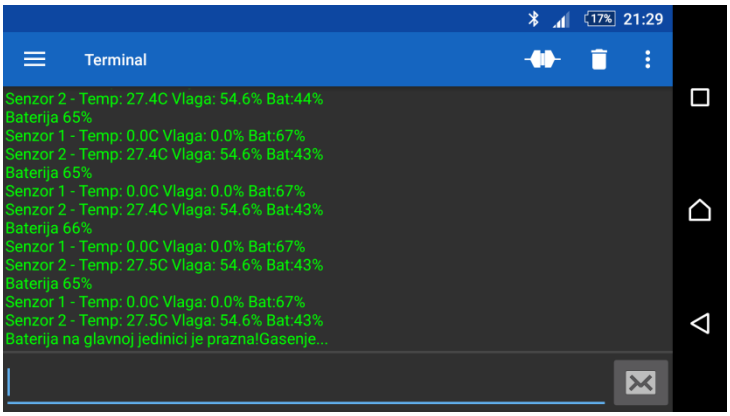
Slika 4.2. Prikaz kada se jednom od senzora približi prepreka

Ukoliko se približi prepreka na udaljenost manje od one koja je podešena, prikaz se mijenja, te se sada ispisuje na kojem je senzoru prepreka otkrivena, da li se prepreka udaljava, približava ili stoji, te na kojoj se udaljenosti prepreka trenutno nalazi.



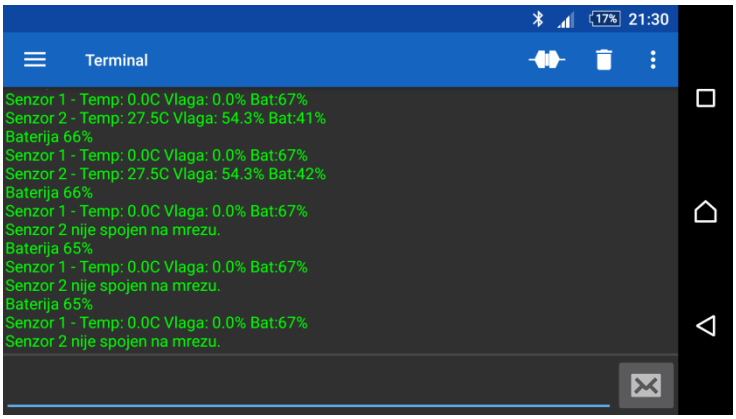
Slika 4.3. Prikaz kada se na jednom od senzora isprazni baterija

Kada se na jednom od senzora isprazni baterija, prikaže se na kojem se senzoru ispraznila baterija, te taj senzor prestaje s radom.



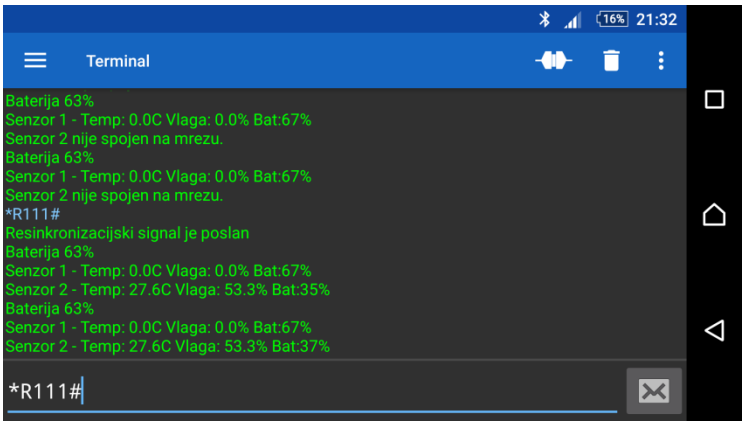
Slika 4.4. Prazna baterija na glavnoj jedinici

U situaciji kada se na glavnoj jedinici isprazni baterija ona to daje do znanja korisniku ispisom i zvučnim signalom (ako je dopušten). Nakon toga prestaje s radom.



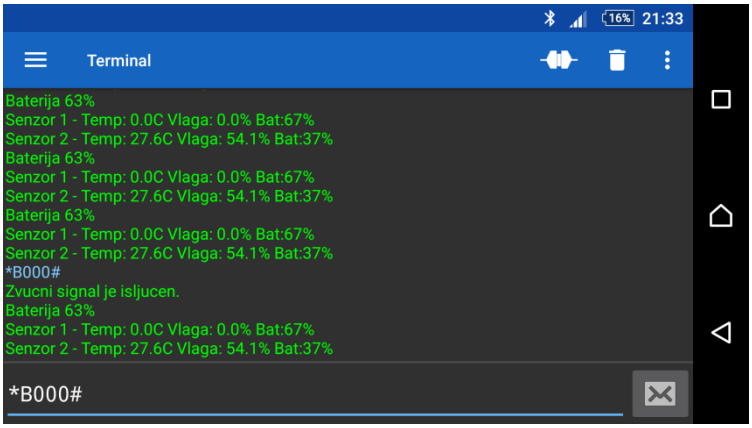
Slika 4.5. Gubljenje veze s jednim od senzora

U slučaju kada se dogodi prekid veze, a to se događa ukoliko glavna jedinica nije primila podatak od nekog od senzora u vremenskom intervalu od 1.5 sekunde, nastane slijedeći prikaz u kojem nam glavna jedinica daje do znanja s kojim je senzorom izgubljena veza.



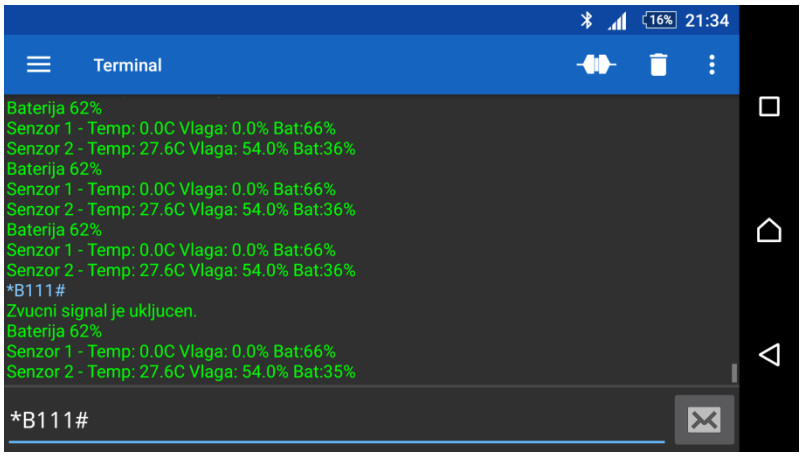
Slika 4.6. Slanje naredbe (komande) za resinkronizaciju

U situaciji kada dođe do gubitka veze, potrebno je pokušati ponovno ostvariti vezu na način da se pošalje resinkronizacija. To se vrši upisom naredbe *R111# u terminal.



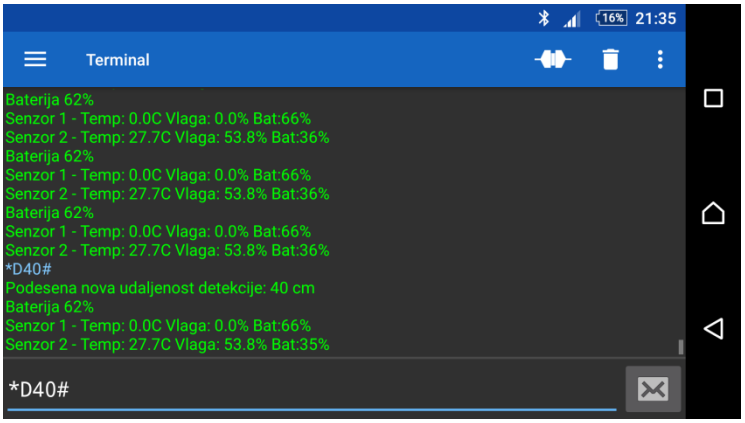
Slika 4.7. Slanje naredbe za gašenje zvučnog signala

U nekim situacijama, može biti potrebno da se onemogući staranje zvučnog signala pri nekom upozorenju. To se može učiti slanjem naredbe *B000# u terminal.



Slika 4.8. Omogućavanje zvučnog signala

Prethodno onemogućen zvučni signal upozorenja, može biti ponovno aktiviran s naredbom *B111#. Prilikom nekog upozorenja, pojavit' će se i zvučni signal s glavne jedinice.



Slika 4.9. Podešavanje okidne udaljenosti, udaljenosti detekcije

Da bi senzor reagirao na neku prepreku, potrebno je podesiti na koju udaljenost da senzor šalje upozorenje. Naredba koju je potrebno poslati jest *Dxxx#, gdje xxx označava udaljenost detekcije prepreke u centimetrima. Dakle ukoliko se želi da se detektira prepreka na 120 cm ili manje, potrebo je poslati naredbu *D120#.

5. ZAKLJUČAK

Mreža senzora s nRF24L01(+) modulima uspješno je napravljena. Omogućava mjerenje udaljenosti prepreke, kao i detekciju pokreta putem bežičnog prijenosa podataka. Također, omogućava da se glavna jedinica, preko Bluetooth™ veze, poveže s pametnim telefonom. Pametni telefon, za prikaz podataka i slanje naredbi (komandi) treba imati instaliranu aplikaciju „Serial Bluetooth“ ili neki slični *Bluetooth Terminal*. Sustav omogućava da se detektira prepreka, pokret, te da se odredi da li se ta prepreka približava, udaljava ili stoji. Također, korisnika može upozoravati zvučnim signalom, ako je to korisnik postavkom omogućio. Osim slanja podataka vezanih za udaljenost, omogućeno je mjerenje i slanje podataka o temperaturi i vlazi zraka, što je izvedeno pomoću DHT22 senzora. Ujedno, sustav prati stanje baterije pojedine senzorske jedinice (odašiljača) i glavne jedinice (prijamnika), šalje status baterije korisniku na pametni telefon, te ga upozorava ukoliko je jedna od baterija prazna. Sustav može prepoznati kada je došlo do gubitka veze između senzorske jedinice i glavne jedinice, te o tome obavijestiti korisnika. Prilikom slanja paketa, postoji sustav protiv sudara paketa (kolizije paketa), na način da prije slanja osluškuje da li je linija slobodna. Mreža senzora, odnosno sustav ima urađeni algoritam za uštedu energije, što omogućava dulji vijek baterije. Domet mreže senzora je oko 20 metara ukoliko ima prepreka, te preko 50 metara ukoliko postoji radiooptička vidljivost. Korisnik može podešavati postavke kao što su zvučno upozorenje, te podešavanje okidne udaljenosti, nakon koje se detektira pokret ili prepreka.

Neka od mogućih poboljšanja su izrada vlastite aplikacije za ovu mrežu senzora bazirana na Android™ platformi, omogućavanje promjene kanala mreže senzora, podešavanje izlazne snage svakog pojedinog dijela mreže (odašiljača i prijamnika), podešavanje okidne udaljenosti za svaki senzor posebno, korekcija brzine zvuka u ovisnosti o vlazi zraka i temperaturi, dodavanje svjetlosnog signala uz zvučni u trenutku upozorenja.

LITERATURA

[1] http://pdfserv.maximintegrated.com/en/an/AN3455.pdf - Pristupljeno dana 21.6.2017.

[2] http://www.ni.com/newsletter/51446/en/ - Pristupljeno dana 21.6.2017.

[3] http://www.ni.com/white-paper/14940/en/ - Pristupljeno dana 21.6.2017.

[4] http://www.lightwaveonline.com/articles/print/volume-30/issue-5/features/which-optical-modulation-scheme-best-fits-my-application.html - Pristupljeno dana 23.6.2017.

[5] http://www.rfwireless-world.com/Terminology/GFSK-vs-FSK.html - Pristupljeno dana 23.6.2017.

[6] https://www.maximintegrated.com/en/app-notes/index.mvp/id/4622 - Pristupljeno dana 24.6.2017.

[7] http://pubs.sciepub.com/automation/3/3/6/ - Pristupljeno dana 24.6.2017.

[8] https://www.sick.com/us/en/product-portfolio/distance-sensors/long-range-distance-sensors/c/g168297 - Pristupljeno dana 24.6.2017.

[9] https://www.megatron.de/en/products/potentiometric-position sensors/potentiometric-linear-transducer-mbx.html - Pristupljeno dana 24.6.2017.

[10] https://en.wikipedia.org/wiki/Speed_of_sound - Pristupljeno dana 24.6.2017.

[11] http://www.thermometricscorp.com/thermocouple.html - Pristupljeno dana 24.6.2017.

[12] https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation - Pristupljeno dana 24.6.2017.

[13] https://gist.github.com/100ideas/1119533 - Pristupljeno dana 24.6.2017.

[14] http://sensorxyz.blogspot.hr/2016/03/lm35-precision-centrigrade-temperature.html - Pristupljeno dana 24.6.2017.

[15] https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf - Pristupljeno dana 24.6.2017.

[16] https://www.tigoe.com/pcomp/code/controllers/all-about-microcontrollers/ - Pristupljeno dana 24.6.2017.

[17] http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf - Pristupljeno dana 24.6.2017.

[18] http://www.atmel.com/Images/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf - Pristupljeno dana 24.6.2017.

[19] http://www.arduino.org/products/boards/arduino-nano - Pristupljeno 24.6.2017.

[20] http://www.ti.com/lit/ds/symlink/lm35.pdf - Pristupljeno dana 24.6.2017

[21] https://github.com/nRF24/RF24 - Pristupljeno dana 28.6.2017.

[22] https://github.com/adafruit/DHT-sensor-library - Pristupljeno dana 28.6.2017.
[23] https://github.com/rocketscream/Low-Power - Pristupljeno dana 28.6.2017.
[24] https://www.onsemi.com/pub/Collateral/MC34063A-D.PDF - Pristupljeno dana 6.9.2017.
[25] http://www.ebay.com/itm/2PCS-5V-Micro-USB-1A-18650-Lithium-Battery-Charging-Board-Charger-Module-ID-/311889725430?hash=item489e1387f6:g:U6UAAOSwIQdZNkEJ - Pristupljeno dana 6.9.2017.

SAŽETAK

Tema ovog rada je bila osmisliti mrežu senzora koji se baziraju na radio primo-predajniku s nRF24L01(+) integriranim krugom. Potrebno je bilo detektirati da postoji prepreka, odnosno pokret, te preko Bluetooth™ veze obavijestiti korisnika putem pametnog telefona. Detekcija pokreta je izvršena pomoću SRF-05 ultrazvučnog senzora udaljenosti , te se prati da li je izmjerena udaljenost koja je primljena paketom na glavnu jedinicu veća ili manja od one koja je podešena kao okidna udaljenost. Ako je manja, korisnika se obavještava putem zvuka i informacije prikazane na pametnom telefonu. Mreža senzora je bežična i prijenosna, dakle napaja se preko baterije. Također, ugrađeno je i mjerenje temperature i vlage zraka pomoću DHT22 senzora temperature i vlage koji koristi jednožičanu komunikaciju s mikroupravljačem. Pošto su jedinice bežične, napajanje baterijom, potrebno je i pratiti stanje baterije i upozoriti korisnika ako su prazne, što je izvedeno pomoću djelitelja napona i analogno-digitalnog pretvarača ugrađen u mikroupravljač. Korisnik ima mogućnosti podešavati postavke slanjem naredbe preko Bluetooth™ veze.

Ključne riječi: Arduino, nRF24L01, senzor udaljenosti, Bluetooth, bežična mreža, detekcija pokreta

ABSTRACT

Task of this project was to make a sensor network that are based on radio transceiver with nRF24L01(+) integrated circuit. It is necessary to detect whether an obstacle or motion exist and then over a Bluetooth connection notify a user using his Smartphone. Motion detection is done by using SRF-05 ultrasonic distance sensor and it monitors measured distance that is received by main unit, sent by sensor unit and comparing whether is that distance bigger or smaller of trigger distance, which is set by user. If it's smaller, user gets notified by sound on main unit and information shown on his Smartphone. Sensor network is wireless and portable, so it gets supplied from DC battery. Oslo, there is a temperature and humidity measurement of air, done with DHT22 air temperature and humidity sensor. Because, all units are wireless and portable , supplied by DC battery, it is necessary to monitor battery voltage and notify a user if batteries are getting low, which is done by using a voltage divider and analog-to-digital converter that is built-in inside a microcontroller. User can change setting sending a command over Bluetooth network.

Title: Microcontroller system for remote motion detection

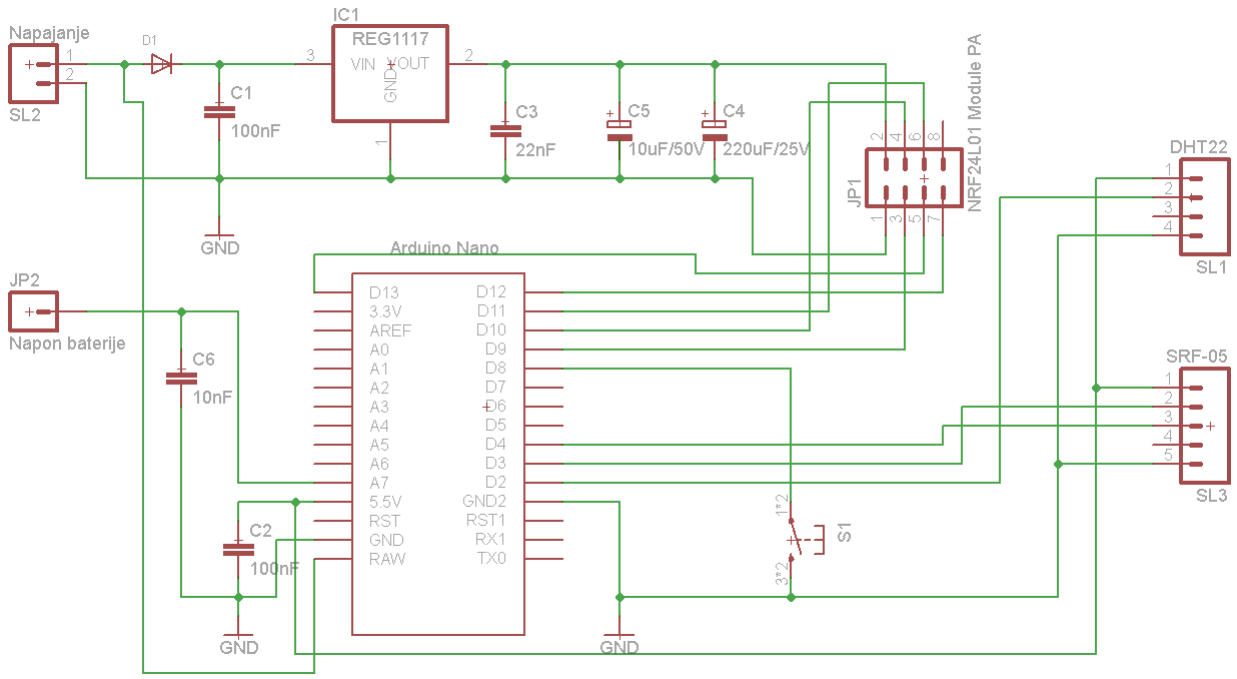
Keywords: Arduino, nRF24L01, distance sensor, Bluetooth, wireless network, motion detection

ŽIVOTOPIS

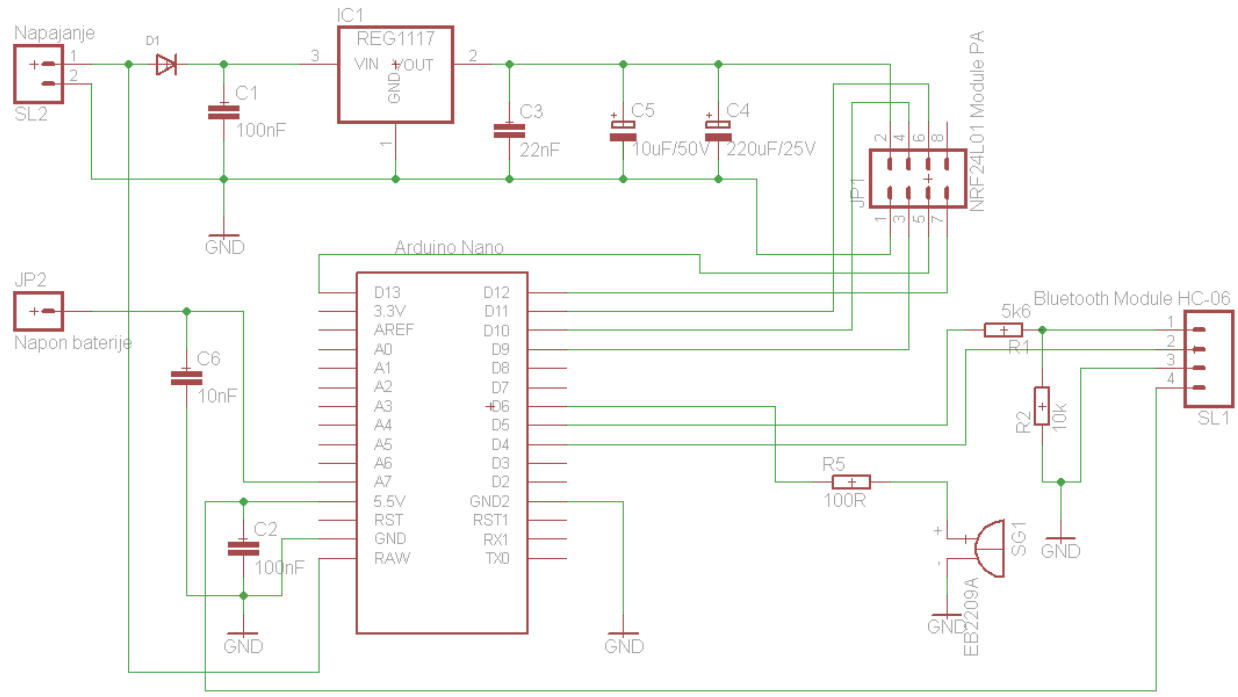
Borna Biro rođen je 13.7.1994. u Osijeku. Pohađao je osnovnu školu Ivana Kukuljevića u Belišću i završio ju 2009. godine. Iste godine upisao je Srednju Školu Valpovo u Valpovu, smjer elektrotehničar. Za vrijeme pohađanja srednje škole sudjelovao je na dvije smotre radova, prva je bila državna smotra radova održana 2012. godine u Selcu, na kojoj se predstavio s projektom „Audio Pojačalo Snage“ te je osvojio drugo mjesto. Druga je bila 2013. godine, održana u Zagrebu na FER-u, na koju se predstavio s projektom „Kompenzator jalove snage“. Iste te godine je završio srednju školu, te polagao državnu maturu. Nakon polaganja državne mature, upisao je elektrotehnički fakultet u Osijeku (Fakultet elektrotehnike, računarstva i informacijskih tehnologija).

Borna Biro

PRILOG



Prilog 1 Shema senzorske jedinice (odašiljača)



Prilog 2 Shema glavne jedinice (prijamnika)

Prilog 3 Programski kod glavne jedinice (prijamnika)

```
#include <SPI.h>                          //Uvoz biblioteke za SPI komunikaciju (koristi se za
komunikaciju s nRF24L01 integriranim krugom)
#include <nRF24L01.h>                     //Uvoz biblioteka za rad i upravljanje s nRF24L01
integriranim krugom
#include <RF24.h>
#include <LowPower.h>                     //Uvoz biblioteke za omogućavanje korištenja rada s
malom potrošnjom, energije kod mikroupravljača
#include <SoftwareSerial.h>               //Uvoz biblioteke za UART komunikaciju baziranoj na
softwareskoj platformi
#include <EEPROM.h>                       //Uvoz biblioteke za EEPROM (trajnu) memoriju

RF24 radio(9, 10);                        //Definiriranje izvoda s kojim ce nRF24L01 biti spojen
s mikroupravljačem
SoftwareSerial serial(4, 5);              //Definiranje izvoda za RX i TX kod UART komunikacije
(RX, TX)
const uint64_t pipes[2] = { 0x0025AADD71LL, 0x0025AADD7CLL };     //Definiranje adresa kod
nRF24L01, koje su potrebne za komunikaciju. Adrese za umrežene uređaje se smiju razlikovati samo
u zadnjem byteu
struct recData {                                                  //Struktura podataka, odnosno
paket podataka koji se šalje glavnoj jedinici
  int temp;
  int vlaga;
  long udaljenost;
  unsigned char baterija;
  unsigned char sinkronizacija;
} paket;
double min_bat = 0, max_bat = 0;    //Definiranje najvećeg i najmanjeg napona baterije. Potrebno
kod računanja postotka

float temperature[2], humidity[2];        //Varijable za spremanje podataka o vremenu sa senzora
temperature i vlage obadviju senzorskih jedinica
double alarmDistance;                     //Definiranje varijable za podešavanje udaljenosti na
kojoj da senzor reagira
double distance[2], distanceOld[] = {alarmDistance, alarmDistance};           //Varijable za
udaljenost oba senzora za udaljenost i varijable koje pamte zadnju vrijednost udaljenosti
(potrebno za detekciju pomaka)
unsigned char bat[3], dataRec = 0, proximityAlarm = 0, pokret = 2;            //Pokret = 2 -
Prepreka miruje, Pokret = 1 - Prepreka se približava, Pokret = 3 - Prepreka se udaljava
unsigned long timer1 = 0, timer2 = 0, weatherTimeout = 0, alarmTimeout = 0;   //Varijable koje se
koriste za istek vremena
char active = 0, praznaBaterija = 0;                          //Varijable koje govore koji su
senzori aktivni na mreži (u bitovima B000000BA, A-senzor 1, B-senzor2) i varijabla za oznaku koja
jedinica ima praznu bateriju
unsigned char buzzer;                                         //Varijabla koja omogućava ili
onemogućava zvučni signal
char command[10];                                             //Varijabla u koju se sprema komanda
koja je poslana preko Bluetooth(TM) modula

void setup() {
  serial.begin(9600);                                 //Podesi brzinu prijenosa podataka na UARTu,
koristi se za Bluetooth
  pinMode(6, OUTPUT);                                 //Podesi izvod 6 na mikroupravljaču na
izlaz (izvod za buzzer)
  unsigned int napon_temp = napon_baterije()*1000;    //Izmjeri napon baterije u milivoltima
  if (napon_temp > 3100 && napon_temp < 4400) {       //Ako je napon između 3.1V i 4.4V, onda se
uređaj napaja preko litijeve baterije
    min_bat = 3.10;                                   //Minimalni radni napon litijeve je oko
2.8V, no radi očuvanja baterije, prekida se s radom ranije
    max_bat = 4.20;                                   //Maksimalni napon litijeve baterije. Oboje
potrebno kod računanja postotoka baterije
  }else if(napon_temp > 7000 && napon_temp < 10000) { //Ako je napon između 7V i 10V, uređaj se
napaja preko NiMH 9V baterije
    min_bat = 7.10;                                   //Minimalni i maksimalni napon baterije,
potrebno za izračunavanje postotka baterije
    max_bat = 9.50;
  }
  if (!status_baterije()) low_bat();                  //Provjeri da li je baterija slaba, ako je
ugasi jedinicu i obavjesti korisnika o tome

  read_parameters();                                  //Ako je s baterijom sve u redu, očitaj
postavke iz EEPROM memorije

  radio.begin();                                      //Inicijaliziraj bibiloteku za nRF24L01
radio primopredajnik
```

```
  radio.setChannel(105);                                  //Postavi odgovarajući kanal za
komunikaciju
  radio.setAutoAck(1);                                    //Postavi automatsku potvrdu primitka
paketa
  radio.setRetries(2, 10);                                //Postavi nRF24L01 automatsku retransmisiju,
10 pokušaja u razmaku od 2 ms
  radio.setPayloadSize(sizeof(recData));                  //Postavi duljinu paketa koji se šalje
  radio.setPALevel(RF24_PA_MAX);
  radio.openWritingPipe(pipes[1]);                        //Podesi adresu komunikacijskog kanala
(pipe) za slanje podataka
  radio.openReadingPipe(1, pipes[0]);                     //Podesi adresu i redni broj
komunikacijskog kanala (pipe) za primanje podataka - Senzor 1
  radio.openReadingPipe(2, pipes[1]);                     //Podesi adresu i redni broj
komunikacijskog kanala (pipe) za primanje podataka - Senzor 2
  randomSeed(analogRead(A0));                             //Podesi izvor odakle će se uzimati
nasumične vrijednosti (potrebno za retransmisiju podataka, no ne nRF24L01 automatsku
retransmisiju)
  radio.startListening();                                 //Prebaci nRF24L01 u režim rada za prijam
podataka
  sync_tx();                                              //Pošalji sinkronizacijski signal na
senzorske jedinice dajući im do znanja da se glavna jedinica upalila i da je spremna za primanje
podataka

  delay(100);                                             //Stanka dok se ne primi podatak, nije
neophodna

  /*double voltage = read_battery();
    if (voltage > 3 && voltage < 5.5) {
    max_bat = 4.2;
    min_bat = 3.4;
    }*/
}

void loop() {
  uint8_t rxPipeNo;                                       //Varijabla za spremanje podatka koji
senzor je posalo podatak
  bat[2] = status_baterije();                             //Očitaj postotak baterije i spremi ga u
varijablu u kojoj se pamte stanja svih baterija
  if (bat[2] == 0) {                                      //Ako se baterija isprazni tokom rada,
obavijesti korisnikam, ugasi nRF i ATMEGA za stalno.
    serial.print(F("Baterija na glavnoj jedinici je prazna!"));
    serial.print(F("Gasenje..."));
    if (buzzer) tone(6, 700, 500);
    delay(500);                                           //Pauza dok buzzer ne odsvira signal i
dok se sve ne pošalje
    radio.powerDown();
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
  }

  if (serial.available() > 4) bluetooth_rx();             //Ako ima nešto primljeno s Bluetooth(TM)
i veće je od 4 znaka, prvovjeri da li primljena neka komanda

  if (radio.available(&rxPipeNo))                         //Provjeri da li ima pristiglih paketa.
Za Senzor 1: rxPipeNo = 1, za Senzor 2: rxPipeNo = 2
  {
    proximityAlarm = 0;                                   //Poništi varijablu za alarm (alarm da li
je prepreka bliže nego što je zadano u distanceAlarm varijabli)
    while (radio.available())                             //Sve dok ima nešto u privremenoj
memoriji u raadio primopredajniku, čitaj podatke i spremaj ih u strukturu, stvarajući tako paket
    {
      radio.read(&paket, sizeof(recData));
    }
    distance[rxPipeNo - 1] = double(paket.udaljenost) / 100;      //Rasporedi vrijednosti iz
paketa u njihove varijable u ovisnosti koji je senzor što poslao
    bat[rxPipeNo - 1] = paket.baterija;
    temperature[rxPipeNo - 1] = float(paket.temp) / 10;
    humidity[rxPipeNo - 1] = float(paket.vlaga) / 10;

    if (alarmDistance > distance[rxPipeNo - 1] && distance[rxPipeNo - 1] > 0) {         //Ako je
izmjerenja udaljenost manja od one što je podešena kao udaljenost alarma, zapamti koji je senzor
aktivirao to
      proximityAlarm = rxPipeNo;
      if (distanceOld[rxPipeNo - 1] - distance[rxPipeNo - 1] >
2)                      //Provjeravaj da li se prepreka udaljava, približava ili stoji.
Uspoređuje se predhodna očitana udaljenost i trenutna
      {
```

```
        pokret =
1;                                                                  //Prepreka se približava
      } else if (distanceOld[rxPipeNo - 1] - distance[rxPipeNo - 1] < -2) {
        pokret =
3;                                                                  //Prepreka se udaljava
      } else {
        pokret =
2;                                                                  //Prepreka stoji
      }
      distanceOld[rxPipeNo - 1] = distance[rxPipeNo - 1];                                 //Na
kraju je ta izmjerena udaljenost, stara (predhodna udaljenost)
    }

    if (paket.baterija == 0) praznaBaterija = rxPipeNo;          //Ako je primljen podatak da je
baterija na nekom senzoru 0%, upozori korisnika na to. Zapamti koji senzori ima slabu bateriju.

    if (rxPipeNo == 1) {                                        //Ako je paket primljen s senzora 1,
obnovi vrijeme trajanja vremensog ograničenja aktivnosti pojedinog senzora
      timer1 = millis();
      active |= 1;                                              //Postavi vrijednost bita varijable za
prikaz aktivnosti pojedinog senzora na mreži, u ovom slučaju senzora 1
    }
    if (rxPipeNo == 2) {                                        //Ako je paket primljen s senzora 2,
obnovi vrijeme trajanja vremensog ograničenja aktivnosti pojedinog senzora
      timer2 = millis();
      active |= 1 << 1;                                         //Postavi vrijednost bita varijable za
prikaz aktivnosti pojedinog senzora na mreži, u ovom slučaju senzora 2
    }
  }

  if (((active >> 0) & 1) && ((unsigned long)(millis() - timer1) > 1500)) active &= ~(1 <<
0);  //Ako nije došao nikakav paket sa senzora 1 više od 1.5 sekunde, smatraj senzor neaktivnim
(veza je pukla)
  if (((active >> 1) & 1) && ((unsigned long)(millis() - timer2) > 1500)) active &= ~(1 <<
1);  //Ako nije došao nikakav paket sa senzora 2 više od 1.5 sekunde, smatraj senzor neaktivnim
(veza je pukla)

  if (praznaBaterija != 0) {                                    //Ako je vrijednost varijable za
detekciju koji senzor ima praznu bateriju različita od nule (ako je vrijednost 1, senzor 1 ima
praznu bateriju, ako 2 -> senzor 2)
    serial.print("Prazna baterija na senzoru ");              //Obavijesti korisnika porukom i
zvučnim signalom (ako je omogućen)
    serial.print(praznaBaterija, DEC);
    serial.println("! Senzor je neaktivan");
    if (buzzer) tone(6, 700, 500);
    praznaBaterija = 0;                                         //Poništi varijablu da se ne ispisuje
stalno, jer će se senzor ionako nakon poslanog podatka o praznoj bateriji ugasiti
  }

  if (proximityAlarm == 0 && (unsigned long)(millis() - weatherTimeout) > 2000 && (unsigned
long)(millis() - alarmTimeout) > 2000) { //Ako nema nikakve detektirane prepreke, prošlo je više
od 2 sekunde i ako je prošlo
    weatherTimeout =
millis();
              //2 sekunde nakon aktivacije alarma za prepreku ispiši podatke o vremenu i
bareriji
    printWeather();
                                    //te poništi varijable za predhodnu udaljenost
    distanceOld[0] = alarmDistance;
    distanceOld[1] = alarmDistance;
  }

  if (proximityAlarm != 0 && (unsigned long)(millis() - alarmTimeout) > 500) {  //Ako je neki od
senzora detektirao prepreku, prikaži koji senzor je to detektirao, da li se prepreka udaljava,
približuje ili stoji, te
    alarmTimeout = millis();                                                          //na kojoj se
trenutno udaljenosti nalazi prepreka
    serial.print(F("UPOZORENJE - Senzor "));
    serial.println(proximityAlarm, DEC);
    serial.print(F("Prepreka "));
    switch (pokret) {
      case 1:
        serial.print(F("se priblizava"));
        break;
      case 2:
        serial.print(F("stoji"));
        break;
```

```
      case 3:
        serial.print(F("se udaljava"));
        break;
    }
    serial.print(F(" - "));
    serial.print(distance[proximityAlarm - 1], 2);
    serial.println(F("cm"));
    if (buzzer) tone(6, 750 + (250 * (4 - pokret)), 50);      //Ako je omogućeno, napravi zvučni
signal, viši ton, prepreka se približava, niži prpreka se uadljava, ton frekvencije između ta dva,
prepreka stoji
    proximityAlarm = 0;                                       //Poništi varijablu za prikaz koji
je senzor detektirao prepreku
  }
}


void printWeather(void)                                       //Funkcija koja ispisuje kolika je
izmjerena vlaga, temperatura zraka, stanje baterije svakog senzora (odašiljača) i status baterije
glavne jedinice
{
  serial.print(F("Baterija "));
  serial.print(bat[2], 1);
  serial.println("%");
  for (int i = 0; i < 2; i++) {
    serial.print(F("Senzor "));
    serial.print(i + 1, DEC);
    if ((active >> i) & 1) {                                  //Ako je trenutno promatrani senzor
aktivan, ispiši podatke
      char serialTemp[50];                                    //Sve podatke složi preko jedne
sprintf funkcije, radi uštede RAMa i FLASHa, decimalne vrijednosti se ispisuju preko integer-a
      sprintf(serialTemp, " - Temp: %d.%dC Vlaga: %d.%d%% Bat:%d%% ", int(temperature[i]),
abs(int(temperature[i] * 10)) % 10, int(humidity[i]), abs(int(humidity[i] * 10)) % 10, bat[i]);
      serial.println(String(serialTemp));
    } else {
      serial.println(" nije spojen na mrezu.");               //Ako nije, ispiši da taj senzor nije
aktivan
    }
  }
}

double napon_baterije(void)                                   //Funkcija koja čita i vraća iznos napona
baterije u voltima
{
  analogReference(INTERNAL);                                  //Podesi referentni napon da se može
izvršiti komparacija po nećemu što je neovisno o naponu baterije
  double adcA7 = 0;                                           //Inicaliziraj varijablu za vrijednost
napona u bitovima (0-1023)
  for (int i = 0; i < 8; i++) {                               //Izvedi računanje srednje vrijednosti
napona, čitaj napon osam puta u intervalima od 600 mikrosekundi
    adcA7 += (double)analogRead(A7);
    delayMicroseconds(600);
  }
  return ((adcA7 / 8 / 1023 * 1.1 / 0.09090909)*1.081);   //Izračujan napon pomoću znanja omjera
naponskog djelitelja (0.9090909), iznosa napona referentne vrijednosti i broju bita A/D
pretvornika.
}                                                          //1.081 je saktor korekcije, tj. točnost mjerenja
je bila 8.1% netočna u odnosu na stvarnu vrijednost (Eksperimentalno određeno).

char status_baterije(void)                                    //Funkcija koja vraća iznos postatak
ostakta energije u bateriji na temelju očitanog napona baretije
{
  double volt = (napon_baterije() - min_bat) / (max_bat - min_bat) * 100;     //Izračunaj
postotak baterije na temelju napona baterije
  if (volt > 100) volt = 100;                                                 //Postotak ne može
biti veći od 100%, dakle ako slučajno i je, vrati ga nazad na 100%
  if (volt < 0) volt = 0;                                                     //Postotak baterije
ne može biti manji od 0%, dakle ako slučajno i je, vrati ga nazad na 0%
  return (char)volt;
}

void sync_tx(void)                  //Funkcija daje do znanja da je glavna jedinica priključena
na mrežu, te da senzori mogu početi slati podatke
{
  radio.stopListening();            //Prebaci radio primopredajnik u režim rada za slanje
podataka
  paket.sinkronizacija = 'S';       //Postavi vrijednost varijable za sinkronizaciju
  for (int j = 0; j < 2; j++) {
```

```
    radio.openWritingPipe(pipes[j]);  //Otvori komunikacijski kanal (pipe) za slanje podataka na
zadanoj adresi (adresa senzora 1, te zatim senzora 2)
    delay(200);                                          //Kašnjenje između slanja
sinkronizacije senzora 1 i senzora 2
    for (int i = 0; i < 5; i++)
    {
      if (radio.write(&paket, sizeof(recData))) break;    //Pošalji najviše pet puta paket
(sinkronizaciju). Ako se dobije povrda, paket je primljen ako ne ponavljaj.
      delay(20);                                         //Kašnjenje između slanja
    }
  }
  //data.address = 2;
  radio.startListening();                                //Nakon slanja sinkronizacije, radio
primopredajnik se prebacuje u režim rada primanja poadtaka
  paket.sinkronizacija = 0;                              //Poništi varijablu za sinkronizaciju
  delay(200);                                            //Kašnjenje dok se senzori podese, nije
obavezna
}

void low_bat() {                                         //Funkcija koja korisnika obavještava
porukom i zvučnim signalom (ako je dozvoljen) da je baterija glavne jedinice prazna i zatim ju
gasi
  serial.println("Niska razina baterije!");
  serial.println("Gasenje...");
  if (buzzer) tone(6, 1000, 500);
  delay(500);
  LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
}

void read_parameters(void)                              //Čitanje parametara iz EEPROM (trajne)
memorije u kojoj podaci ostanu nakon gašenja napajanja
{
  buzzer = EEPROM.read(0);                              //Očitaj podatak iz EEPROMa vezan za
postavke buzzera
  if (buzzer > 1) {                                     //Ako je podatak veći od jedan (dakle
podatak može biti ili nula ili jedan) podatak je nevaljan
    buzzer = 1;                                         //Snimi valjan podatak u EEPROM
    EEPROM.write(0, buzzer);
  }
  alarmDistance = double(EEPROM.read(1) << 8 | EEPROM.read(2));   //Udaljenost je tipa integer,
dakle dva bytea, pa zauzimaju dvije zasebne adrese, koje kada se očitavaju, moraju nazad spojiti
u jedan integer
  if (alarmDistance > 400 || alarmDistance < 5)                    //Ako je podatak ne valjan,
snimi novi, valjan podatak.
  {
    alarmDistance = 100;
    EEPROM.write(1, highByte(int(alarmDistance)));                 //Integer je dva bytea, pa ga
je za spremanje u EEPROM potrebno razdvojiti
    EEPROM.write(1, lowByte(int(alarmDistance)));
  }
}

void bluetooth_rx(void)
{
  uint8_t n = 0;                                            //Varijabla koja služi kao brojač
elemenata (simbola)
  char temp_serial[] = {"                                      "};  //Privremena varijabla za
prikupljanje znakova s UARTa, max do 40 znaka
  do {
    temp_serial[n] = char(serial.read());                   //Čitaj što imaš u UART RXu sve dok
ne naletiš na znak koji označava kraj komande ili dok ne popuniš cijelu privremenu varijablu za
UART RX
    if (temp_serial[n] == '#') break;
    n++;
    delay(1);                                               //Pošto je BAUD 9600, što je
relativno sporo (uspoređujući 16MHz takta), pričekaj malo da se UART RX napuni. Nije
najefikasnija metoda, ali radi.
  } while (n < 40);
  char start_s = -1, stop_s = -1;                           //Definiraj varijable koje će
označavati na kojem mjestu počinje, a na kojem završava komanda u privremenoj varijabli za UART
RX.
  for (int i = 0; i < sizeof(temp_serial); i++)
  {
    if (temp_serial[i] == '*') start_s = i;                 //Kad naletiš na znak koji označava
početak komande zapamti to mjesto u polju.
```

```
    if (temp_serial[i] == '#') stop_s = i;                    //Isto tako, kad naletiš na znak
koji označava kraj komande, zapami to mijesto (nije potrebno pošto je taj znak uvijek zadnji, no
ipak za svaki slučaj).
  }
  if (start_s != -1 && stop_s != -1)                          //Ako su varijable promijenjile
svoje početne (inicijalizirane) vrijednosti, dakle pronašli smo mjesta gdje komanda počinje i
završava
  {
    for (int i = start_s; i < stop_s - 1; i++)         //Kopiraj samo komandu u poseban string
    {
      command[i - start_s] = temp_serial[i + 1];
    }
  } else {
    serial.flush();                                           //Tu ostaviti flush() jer garant je
primio nekakve gluposti koje nemaju veze s komandom.
  }
  char cmd = command[0];                                      //Spremi slovo komande u zasebnu
varijablu
  char cmd_data[9];                                           //Varijabla za analizu brojeva (nju
ćemo koristiti kako bi iz stringa dobili broj)

  for (int i = 1; i < 9; i++)                                 //Pomakni string za jedno mjesto u
lijevo, jer atoi, da prepozna int, mora počinjati s brojem ili +/- znakom, a ovdje počinje s
slovom.
  {
    cmd_data[i - 1] = command[i];
  }
  switch (cmd)
  {
    case 'D':                                                 //Ako je komanda 'D', to označava
da će se udaljenost za detekciju promijeniti i da broj iza te komande označava na koju će se novu
udaljenost podesiti
      {
        int setDistance = atoi(cmd_data);
        serial.print("Podesena nova udaljenost detekcije: "); //Prikaži porukom da je došlo do
promjene udaljenosti detekcije
        serial.print(setDistance, DEC);
        serial.println(" cm");
        alarmDistance = setDistance;
        EEPROM.write(1, highByte(int(alarmDistance)));        //Spremi novu udaljenost detekcije
u trajnu EEPROM memoriju
        EEPROM.write(2, lowByte(int(alarmDistance)));
        break;
      }

    case 'B':                                                 //Ako je primljena komanda 'B',
znači da slijedi postavka buzzera, ako je broj nakon komande nula, buzzer je ugašen, ako je veći
od 1, onda je upaljen
      {
        if (atoi(cmd_data)) {
          buzzer = 1;
        } else {
          buzzer = 0;
        }
        serial.print(F("Zvucni signal je "));                 //Obavijesti korisnika o novoj
postavci
        buzzer ? serial.print(F("ukljucen")) : serial.print(F("isljucen"));
        serial.println(".");
        EEPROM.write(0, buzzer);                              //Spremi postavku u EEPROM
      }
      break;

    case 'R':                                                   //Ako je primljena komanda 'R'
znači da korisnik traži da se izvede resinkronizacija senzora s glavnom jedinicom
      serial.println(F("Resinkronizacijski signal je poslan")); //Obavijesti korisnika o primitku
komande
      sync_tx();
      break;

    default:
      serial.println(F("Unesena komanda nije valjana!"));       //Ukoliko ništa od toga nije
prepoznato, komanda koja je unešena nije valjana!
      break;
  }
  strcpy(command, "         ");                               //Nakon ispisa, očisti string da
bude spreman za novu komandu
}
```

Prilog 4 Programski kod senzorske jedinice (odašiljača)

Napomena: Programi obje senzorske jedinice su slični, razlikuju se samo u vrijednosti varijable redniBrojSenzora.

```
#include <SPI.h>                          //Uvoz biblioteke za SPI komunikaciju (koristi se za komunikaciju s nRF24L01 integriranim krugom)
#include <nRF24L01.h>                     //Uvoz biblioteka za rad i upravljanje s nRF24L01 integriranim krugom
#include <RF24.h>
#include <LowPower.h>                     //Uvoz biblioteke za omogućavanje korištenja rada s malom potrošnjom, energije kod mikroupravljača
#include <DHT.h>                          //Uvoz biblioteke za senzor vlage i temperature zraka

RF24 radio(9, 10);                        //Definiriranje izvoda s kojim ce nRF24L01 biti spojen s mikroupravljačem
#define OKID_PIN  3                       //Spoji TRIG pin na ultrazvučnom senzoru na arduino digitalni pin broj 3
#define ECHO_PIN  4                       //Spoji ECHO pin na ultrazvučnom senzoru na arduino digitalni pin broj 4
#define DHTPIN 2                          //Koji se izvod na mikroupravljaču koristi za komunikaciju s DHT22 senzorom vlage i temperature
#define DHTTYPE DHT22                     //Koji model senzora za temperaturu i vlagu se koristi - DHT 22  (AM2302)
#define reSync_pin   8                    //Izvod prekidača za resinkronizaciju glavne jedinice s senzorima
#define reysnc_led   1                    //Izvod za svijetleću diodu koja označava da je pokrenut postupak resinkronizacije s glavnom jedinicom

double min_bat = 0, max_bat = 0;    //Definiranje najvećeg i najmanjeg napona baterije. Potrebno kod računanja postotka

DHT dht(DHTPIN, DHTTYPE);                 //Podešavanje biblioteke, postavljanje odgovarajućeg modela senzora i izvoda

const uint64_t pipes[2] = { 0x0025AADD71LL, 0x0025AADD7CLL };     //Definiranje adresa kod nRF24L01, koje su potrebne za komunikaciju. Adrese za umrežene uređaje se smiju razlikovati samo u zadnjem byteu
const char redniBrojSenzora = 2;                                  //Označava da li je ovaj program namjenjen za senzor broj 1 ili senzor broj 2, pošto se isti program koristi za oba senzora, razlika je samo u adresama
char ponovni_pokusaj;                                             //Varijabla koja pamti broj pokušaja retransmisije
unsigned long resync_timeout = 0;                                 //Varijabla koja pamti vrijeme i trajanje resinkronizacijskog postupka
struct recData {                                                  //Struktura podataka, odnosno paket podataka koji se šalje glavnoj jedinici
  int temp = 0;
  int vlaga = 0;
  long udaljenost = 0;
  unsigned char baterija = 0;
  unsigned char sinkronizacija = 0;
} paket;

void setup() {
  pinMode(ECHO_PIN, INPUT_PULLUP);              //Postavi pin br. 4 kao ulaz i omogući mu pull up (poželjno je staviti i vanjski pull up od 680R).
  pinMode(OKID_PIN, OUTPUT);                    //Postavi pin br. 3 kao izlaz (može se staviti vanjski pull down ili up, čisto da pin na senzoru ne bude u Z statu tijekom paljenja Arduina.
  pinMode(reSync_pin, INPUT_PULLUP);            //Postavi izvod broj 8 na Arduino pločici kao ulaz i omogući pull up
  delay(10);                                    //Pričekaj malo dok se sve ne stabilizira, pa zatim pročitaj stanje baterije

  unsigned int napon_temp = napon_baterije()*1000;    //Izmjeri napon baterije u milivoltima
  if (napon_temp > 3100 && napon_temp < 4400) {       //Ako je napon između 3.1V i 4.4V, onda se uređaj napaja preko litijeve baterije
    min_bat = 3.10;                                   //Minimalni radni napon litijeve je oko 2.8V, no radi očuvanja baterije, prekida se s radom ranije
    max_bat = 4.20;                                   //Maksimalni napon litijeve baterije. Oboje potrebno kod računanja postotoka baterije
  }else if(napon_temp > 7000 && napon_temp < 10000) { //Ako je napon između 7V i 10V, uređaj se napaja preko NiMH 9V baterije
    min_bat = 7.10;                                   //Minimalni i maksimalni napon baterije, potrebno za izračunavanje postotka baterije
    max_bat = 9.50;
  }
```

```
  if (!status_baterije()) LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);  //Ako je baterija
0%, ugasi uređaj jer nema smisla da radi.

  dht.begin();                                    //Inicijaliziraj biblioteku za senzor vlage i
temperature zraka

  radio.begin();                                          //Inicijaliziraj biblioteku za rad s
nRF24L01
  radio.setChannel(105);                                  //Postavi odgovarajući kanal za
komunikaciju
  radio.setAutoAck(1);                                    //Postavi automatsku potvrdu primitka
paketa
  radio.setRetries(2, 10);                                //Postavi nRF24L01 automatsku
retransmisiju, 10 pokušaja u razmaku od 2 ms
  radio.setPayloadSize(sizeof(recData));                  //Postavi duljinu paketa koji se šalje
  radio.setPALevel(RF24_PA_MAX);
  radio.openWritingPipe(pipes[redniBrojSenzora - 1]);     //Podesi adresu komunikacijskog kanala
(pipe) za slanje podataka
  radio.openReadingPipe(0, pipes[redniBrojSenzora - 1]);  //Podesi adresu i redni broj
komunikacijskog kanala (pipe) za primanje podataka
  randomSeed(analogRead(A0));                             //Podesi izvor odakle će se uzimati
nasumične vrijednosti (potrebno za retransmisiju podataka, no ne nRF24L01 automatsku
retransmisiju)
  radio.startListening();                         //Prebaci nRF24L01 u režim rada za prijam
podataka
  //Serial.begin(9600);                           //Brzina UARTa - Potrebno samo kod debugginga!
  cekaj_mastera();                                //Pričekaj signal za sinkronizaciju od glavne
jedinice (prijamnika podataka)

}

void loop() {
  bool ok = false;                                        //Inicijalizacija varijable za
uspješnost slanja paketa
  paket.baterija = status_baterije();                     //Pročitaj status baterije u
postocima
  paket.udaljenost = long(mjeri_udaljenost() * 100);      //Pročitaj udaljenost s senzora za
mjerenje udaljenosti i vrati podatak u centimetrima
  ocitaj_meteo_podatke();                                 //Očitaj temperaturu i vlagu sa
senzora
  radio.powerUp();                                        //Pokreni nRF24L01 radio
primopredajnik
  radio.startListening();                                 //Prebaci nRF24L01 u režim rada za
prijam podataka
  while (radio.testCarrier()) {                           //Provjeri da li je "linija" slobodna
(da li na liniji postoji signal nosioc). Ako je, šalji, inače čekaj
    delay(random(100));                                   //Čekaj neko nasumično vrijeme koje
nije veće od 100 milisekundi
  }
  radio.stopListening();                                  //Ako je linija čista šalji paket
  ok = radio.write(&paket, sizeof(recData));              //Pošali paket i spremi uspješnost
slanja u varijablu
  delay(2);                                               //Daj vremena nRF24L01 da pošalje
podatak, te ga zatim ugasi
  radio.powerDown();                                      //Prebaci nRF24L01 u režim niske
potrošnje
  delay(2);                                               //Daj vremena da arduino izbasi sve
iz registara za SPI komunikaciju, te ga zatim stavi u low power mode
  if (!paket.baterija) {                                  //Ako se baterija isprazni tokom rada,
ugasi nRF i ATMEGA za stalno.
    radio.powerDown();
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
  }
  ok ? ponovni_pokusaj = 10 : ponovni_pokusaj--;          //Ako druga strana nije primila
podatak, pokušaj ponovno poslati podatak i tako 10 puta, nakon čega odustani i ugasi se
  if (ponovni_pokusaj == -1) cekaj_mastera();
  delay(10);
  LowPower.powerDown(SLEEP_250MS, ADC_OFF, BOD_OFF);      //Ugasi ATMEGA328P (mikroupravljač)
na 250 milisekundi, odnosno, prebaci ga u režim niske potrošnje energije

}

double mjeri_udaljenost(void)                             //Funkcija za mjerenje udaljenosti
pomoću ultrazvučnog senzora
{
  unsigned long pocetno_vrijeme, zavrsno_vrijeme;         //Inijcjaliziraj vrijable za trenutak
kada je posan signal i kada je primljen signal
```

```
  char greska = 0;                                          //Inicijaliziraj varijablu za
uspješnost mjerenja udaljenosti
  digitalWrite(OKID_PIN, HIGH);                             //Pošalji impuls od 10 milisekundi na
okidni ulaz senzora udaljenosti
  delayMicroseconds(10);
  digitalWrite(OKID_PIN, LOW);

  unsigned long timeout = millis();                         //Varijabla koja prati vrijeme
impulsa
  do {
    if ((unsigned long)(millis() - timeout) > 30) {         //Čekaj sve dok je Echo izvod
(povratni izvod) na senzoru za mjerenje udaljenosti 0
      greska = 1;                                           //Ako prođe više od 30 milisekundi i
nema logičke jedinice, senzor ili ne postoji ili je neispravan, označi neuspjeh u mjerenju
      break;
    }
  }
  while (!digitalRead(ECHO_PIN));

  pocetno_vrijeme = micros();                               //Spremi treutno vrrijeme, vrijeme
kada se pojavila logička jedinica na ulazu (Echo izvodu)
  timeout = millis();

  if (greska) return 0;                                     //Ako postoji neuspjeh u mjerenju,
odustani od nastavka mjerenja

  do {
    if ((unsigned long)(millis() - timeout) > 30) {         //Čekaj sve dok je Echo izvod na
senzoru za mjerenje udaljenosti 1
      greska = 1;                                           //Ako prođe više od 30 milisekundi i
nema logičke nule, senzor ili ne postoji ili je neispravan, označi neuspjeh u mjerenju
      break;
    }
  }
  while (digitalRead(ECHO_PIN));

  zavrsno_vrijeme = micros();                   //Spremi trenutak kada se pojavila logička nula
na Echo izvodu senzora

  if (greska) return 0;                         //Ako postoji neuspjeh u mjerenju, odustani, ako
je sve u redu, izračunaj udaljenost pomoću pravila i formulom opisanom u teroriji

  double distance = (double(zavrsno_vrijeme - pocetno_vrijeme) * 0.017);  //Udaljenost[m] =
vrijeme trajanja logičke jedinice[sec] * Vzvuka/2 => ugaljenost [cm] = 1E-6 * 340/2 * 100 = 0.017
  return (distance);                                                      //Vrati rezultat u
cetimetrima
}

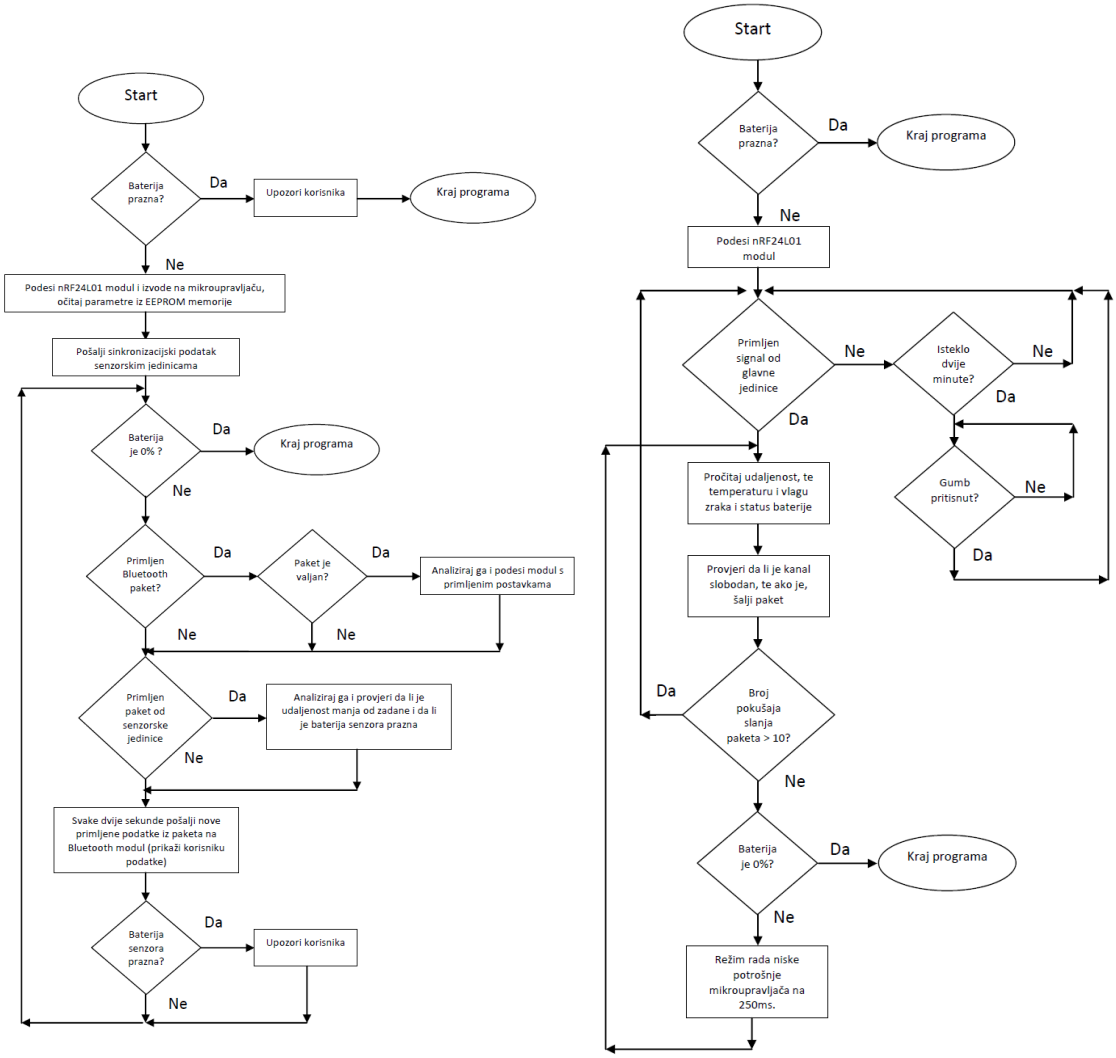
bool ocitaj_meteo_podatke() {                               //Funkcija koja čita temperaturu i
vlagu zraka s DHT22 senzora
  int vlaga_ = int(dht.readHumidity() * 10);                //Pročitaj temperaturu s senzora i
pomakni decimalnu točku jedno mjesto u desno (da se ukloni decimalni dio broja)
  int temperatura_ = int(dht.readTemperature() * 10);       //Pročitaj vlagu s senzora i pomakni
decimalnu točku jedno mjesto u desno (da se ukloni decimalni dio broja)
  if (isnan(vlaga_) || isnan(temperatura_))                 //Ako ima greške tijekom čitanja,
vrijednosti će biti NaN - Not a Number
  {
    paket.vlaga = 0;                                        //Tada su svi iznosi jednaki nuli,
ukazivajući na to da je senzor ili neispravan ili ne postoji
    paket.temp = 0;
    return 0;                                               //Vrati oznaku da je nastao neuspjeh
u čitanju podataka
  } else {                                                  //Ako nisu NaN, već su neki brojevi,
prebaci iznose temperature i vlage u paket za slanje
    paket.vlaga = vlaga_;
    paket.temp = temperatura_;
    return 1;                                               //Vrati oznaku da je nastao uspjeh u
čitanju podataka
  }
}

double napon_baterije(void)                                 //Funkcija koja čita i vraća iznos
napona baterije u voltima
{
  analogReference(INTERNAL);                                //Podesi referentni napon da se može
izvršiti komparacija po nećemu što je neovisno o naponu baterije
```

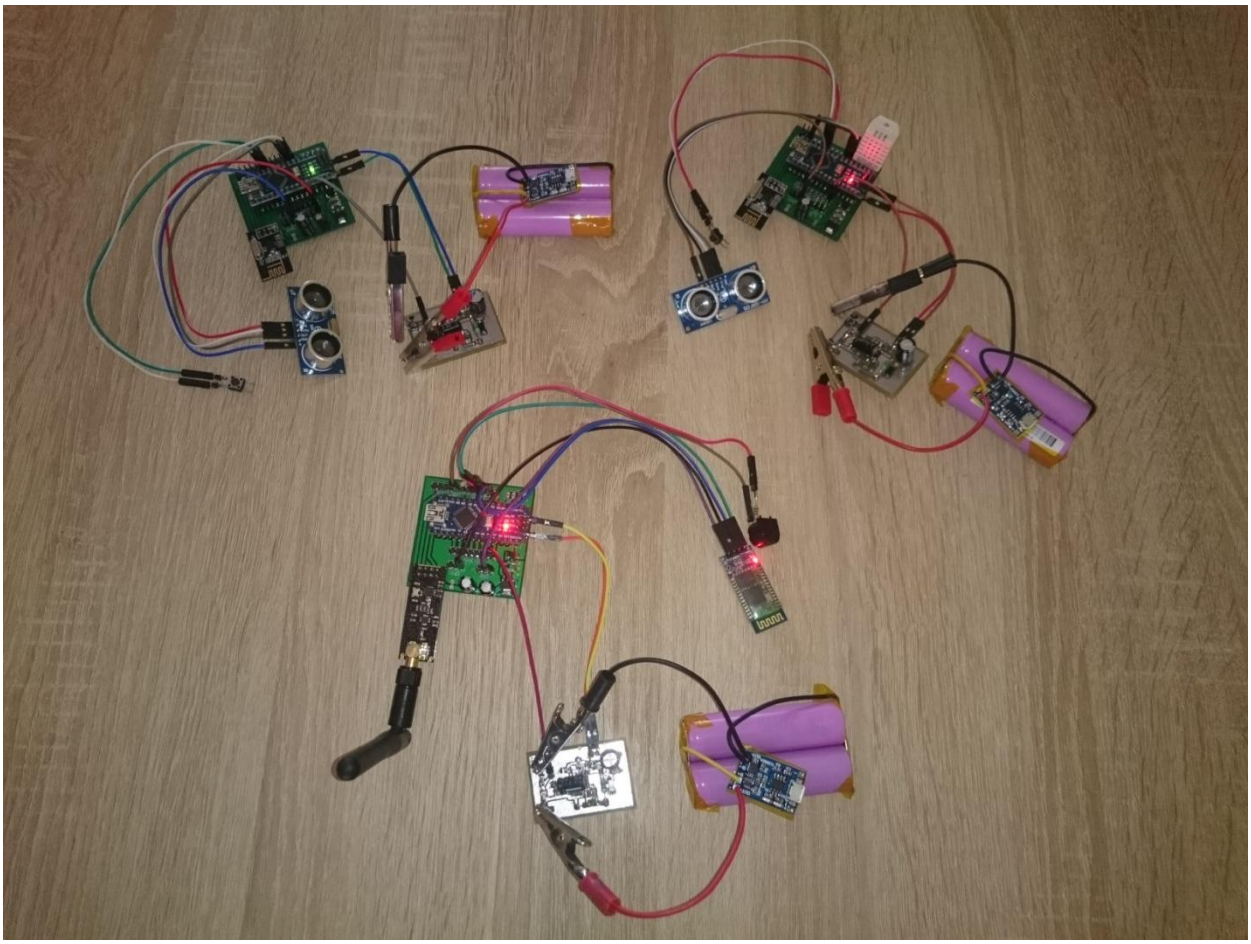
```
  double adcA7 = 0;                                          //Inicaliziraj varijablu za
vrijednost napona u bitovima (0-1023)
  for (int i = 0; i < 8; i++) {                              //Izvedi računanje srednje
vrijednosti napona, čitaj napon osam puta u intervalima od 600 mikrosekundi
    adcA7 += (double)analogRead(A7);
    delayMicroseconds(600);
  }
  return ((adcA7 / 8 / 1023 * 1.1 / 0.090909090909)*1.081);         //Izračujan napon pomoću
znanja omjera naponskog djelitelja (0.9090909), iznosa napona referentne vrijednosti i broju bita
A/D pretvornika
}                                                  //1.081 je saktor korekcije, tj. točnost
mjerenja je bila 8.1% netočna u odnosu na stvarnu vrijednost (Eksperimentalno određeno).

char status_baterije(void)                                   //Funkcija koja vraća iznos postatak
ostakta energije u bateriji na temelju očitanog napona baretije
{
  double volt = (napon_baterije() - min_bat) / (max_bat - min_bat) * 100;   //Izračunaj postotak
baterije na temelju napona baterije
  if (volt > 100) volt = 100;                                                //Postotak ne može
biti veći od 100%, dakle ako slučajno i je, vrati ga nazad na 100%
  if (volt < 0) volt = 0;                                                    //Postotak baterije
ne može biti manji od 0%, dakle ako slučajno i je, vrati ga nazad na 0%
  return (char)volt;
}

void cekaj_mastera() {                                       //Funkcija koja čeka sinkronizacijski
signal od glavne jedinice i vrši proceduru sinkroniziranja
  radio.startListening();                                    //Prebaci radio primopredajnik u način
rada za primanje podataka (paketa)
  pinMode(reysnc_led, OUTPUT);               //Postavi izvod za svjetleću diodu na izlaz
  resync_timeout = millis();                                 //Inicijaliziraj varijablu za vrijeme
trajanja sinkronizacije - Postupak traje 2 minuta
  delay(2);                                                  //Daj vremena da Arduino stigne sve
podatke poslati iz registara prije nego li uđe u Low Power
  do {
    digitalWrite(reysnc_led, LOW);                           //Svjetlećom diodom označi da je u
tijeku procedura za sinkronizaciju
    LowPower.powerDown(SLEEP_500MS, ADC_OFF, BOD_OFF);     //Radi uštede energije, ugasi
mikroupravljač na 500 milisekundi tijekom sinkronizacije
    if (radio.available()) {                                 //Ako je radio nešto primio, pokupi taj
pajet i analiziraj ga
      while (radio.available())
      {
        radio.read(&paket, sizeof(recData));
      }
    }
    //Ako je prošlo 2 minute, a sinkronizacija nije uspješna, ugasi radio primopredajnik
    if ((unsigned long)(millis() - resync_timeout) > 461) //"100mS" millis() = 26sec u stvarnosti
-> 120sec u stvarnosti = 461mS millis(). Razlog zašto je takvo vrijeme je taj što se tijekom low
power modea gasi oscilator
    { //uključujući i timer1 koji je zaslužan za rad millis funkcije
      radio.powerDown();                                     //Ugasi radio primopredajnik
      digitalWrite(reysnc_led, HIGH);                        //Ugasi svjetleću diodu, označavajući
da je proces sinkronizacije završen
      delay(2);
      do {
        LowPower.powerDown(SLEEP_250MS, ADC_OFF, BOD_OFF);  //Čekaj da se pritisne gumb za
ponovnu resinkronizaciju, a do tada povremeno ugasi mikroupravljač radi uštede energije
      } while (digitalRead(reSync_pin));
      radio.powerUp();                                       //Ako se pritisla tipka za ponovnu
sinkronizaciju, upali radio i ponovi cijeli postupak sinkronizacije u trajanju od 2 minute
      radio.startListening();
      resync_timeout = millis();
    }
    delay(2);
  } while (paket.sinkronizacija != 'S');                     //Ako pristigli paket ima podatak za
sinkronizaciju, počni s radom
  radio.stopListening();                                     //Prebaci radio primopredajnik u
režim rada za slanje padataka
  paket.sinkronizacija = 0;                                  //Poništi podatak za sinkronizaciju
  ponovni_pokusaj = 10;                                      //Podesi broj dozvoljenih
retransmisija prije nego odašiljač odluči da je veza između njega i glavne jedinice pukla i vrati
se na ovu funkciju
  digitalWrite(reysnc_led, HIGH);                            //Ugasi svjetleću diodu za
označavanje postupka sinkronizacije
  delay(2);
}
```



Prilog 5 Dijagrami toka programa glavne (lijevo) i senzorske jedinice (desno)



Prilog 6 Slika gotovog rada

Prilog 7 Shema i nacrt tiskane pločice DC – DC pretvarača

