ABS sustav kočenja za bicikl

Valenčak, Marin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

ABS SUSTAV KOČENJA ZA BICIKL

Završni rad

Marin Valenčak

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Marin Valenčak
Studij, smjer:	Stručni prijediplomski studij Elektrotehnika, smjer Automatika
Mat. br. pristupnika, god.	A 4628, 27.07.2020.
JMBAG:	0165084791
Mentor:	prof. dr. sc. Tomislav Keser
Sumentor:	prof. dr. sc. Damir Blažević
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	prof. dr. sc. Tomislav Keser
Član Povjerenstva 2:	doc. dr. sc. Tomislav Galba
Naslov završnog rada:	ABS sustav kočenja za bicikl
Znanstvena grana završnog rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Projektirati, izraditi i evaluirati sustav za kočenje prema ABS principu s implementaijom na biciklu. Sustav realizirati uporabom mikroračunalnog sustava, senzora vrtnje kotača te aktuatora za aktivaciju kočionog sustava. Implementirati jednostavni upravljački program za upravljanje sustavom temeljem ABS principa. Provesti evaluaciju implementiranog ustava. (Rezervirano: Marin Valenčak)

Datum ocjene pismenog dijela završnog rada od strane mentora:	11.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	30.09.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	30.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Marin Valenčak
Studij:	Stručni prijediplomski studij Elektrotehnika, smjer Automatika
Mat. br. Pristupnika, godina upisa:	A 4628, 27.07.2020.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: ABS sustav kočenja za bicikl**

izrađen pod vodstvom mentora prof. dr. sc. Tomislav Keser

i sumentora prof. dr. sc. Damir Blažević

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

ABS (engl. *Anti-lock Braking System*) tj. sustav protiv blokiranja kotača je elektronički sustav koji se ugrađuje u motorna vozila kao što su motocikli i automobili i najnovije u električne bicikle. Napravljen je kao mjera sigurnosti koja tijekom naglih kočenja sprječava blokiranje kotača. ABS je dodatna mjera sigurnosti koja omogućava vozaču da zadrži kontrolu nad vozilom tijekom naglih kočenja, čak i u ekstremnim situacijama. Ta stabilnost je ključna pri velikim brzinama ili na zavojima. Najbolje djeluje na skliskim površinama poput mokrog asfalta ili pješčane staze. Omogućava sigurnije zaustavljanje vozila jer sprječava blokiranje kotača. Iako ABS može produljiti zaustavni put i dalje olakšava upravljivost vozila te sprječava nekontrolirana proklizavanja.

ABS ima tri glavne komponente koje zajedno osiguravaju pravilno funkcioniranje sustava. Senzor brzine kotača koji je smješten na svakom kotaču ili diferencijalu vozila i stalno prati brzinu rotacije kotača. Signal s ovih senzora se kontinuirano šalje prema ECU te omogućuje da sustav reagira u stvarnom vremenu. ECU (engl. *Electronic Control Unit*) je „mozak“ ABS sustava, njegova funkcija je primanje signala sa senzora brzine kotača i njihova analiza. On uspoređuje vrijednosti i otkriva znakove blokade kotača. Nakon detekcije blokade, aktivira disk kočnice koji izvršava mehaničko kočenje. Zajednički rad ovih komponenti omogućuje funkcionalnost ABS sustava.

Senzor brzine kotača prati rotaciju kotača i šalje signal ECU-u koji prema tome otkriva ako postoji blokada kotača. Tijekom kočenja ECU uspoređuje brzinu vrtnje svakog kotača. Ako otkrije da jedan ili više kotača usporavaju znatno brže nego ostali aktivira i otpušta kočnice. Aktivacijom ABS-a sustav modulira tlak kočnica te umjesto potpunog blokiranja kočnica sustav otpušta i primjenjuje pritisak kočnice. Ovaj postupak sprječavanja blokadu kotača, pomaže zadržati kontrolu nad vozilom čak i pri snažnom kočenju. Proces je najbolje prikazan kada vozač skreće vozilom u isto vrijeme kada se zaustavlja. Ovim postupkom se osigurava upravljivost i nastavak kretanja unatoč naglom kočenju. Princip ABS na motociklu i e-biciklu je jednak uz promjene za primjenu na dva kotača.

Zadatak se temeljiti na projektiranju i izradi sustava za kočenje prema ABS principu s implementacijom na biciklu. Ključni koraci su odabir komponenata kao što su mikroupravljač, senzori, napajanje i ostalo, izrada fizičke konstrukcije, proces planiranja sheme u programu Kicad, lemljenje komponenti na bušene pločice (eng. *Protoboard*). Implementacija ABS sustava na bicikl se ostvaruje većinom programski s pomoću STM32 mikroupravljača. On uz pomoć

senzora i ostalih komponenti omogućuje detekciju blokade kotača, kontrolu sustava putem prekidača i opcija ručne kočnice koja potpuno blokira kotač kada je potrebno.

1.1. Zadatak i struktura rada

Zadatak je napraviti konstrukciju za sve komponente koje se koriste. Glavni dio je smjestiti dvije STM32 pločice u kućište koje se nalazi u okviru. Također treba isplanirati gdje se postavlja servomotor i koračni motor da ne smetaju za vrijeme probnih vožnji i testiranja.

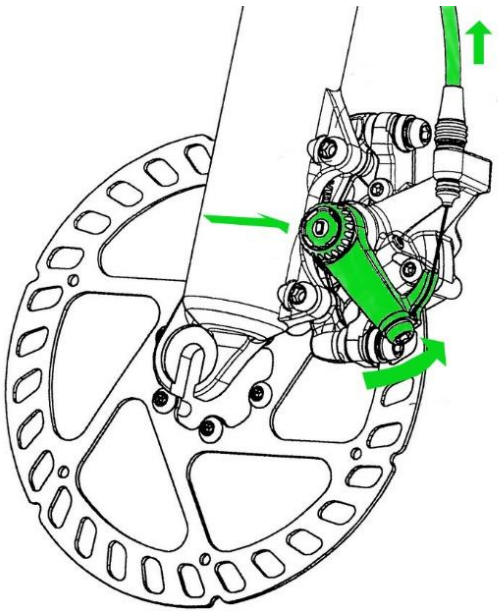
Uvodno poglavlje donosi osnovne informacije o ABS sustavu, njegov širok spektar primjena na vozilima i mogućnost prilagodbe na druge oblike prijevoza kao bicikl. Naglašava se korisnost sustava i sigurnost koja se postiže. Drugo poglavlje se fokusira na teorijski pregled izazova i rješenja koja se javljaju tijekom razvoja. U prijedlogu programskog i sklopovskog rješenja se prikazuju osnovne funkcije i komponente koje se koriste. Treće poglavlje detaljnije opisuje alate i komponente koje se koriste u tazvoju ABS-a. Poglavlje također prikazuje hardverski aspekt sustava, uključujući izradu pločice s električne sheme kao i softversku stranu. Četvrto poglavlje je posvećeno testiranju sustava i analizi rezultata. Opisuje se postupak testiranja u stvarnim uvjetima, rezultati su zabilježeni i uspoređeni. Na temelju testova se predlažu promjene u kodi radi optimizacije performansi.

2. ABS SUSTAV KOČENJA ZA BICIKL

2.1. Teorijski osvrt na problem ABS sustava kočenja

Sustav ABS-a se temelji na preciznosti upravljanja trima glavnim silama koje djeluju na bicikl tijekom kočenja. Da se bicikl opremi s ABS-om mora se postići stanje ravnoteže između tih sila. Sila trenja koja se stvara između diska kočnice i kotača te kotača i podloge. Ona usporava rotaciju kotača. Normalna sila koja djeluje okomito na podlogu težinom bicikla i vozača. Ona povećava pritisak na kotače što čini trenje efikasnijim. Sustav ABS-a uzima u obzir promjene u prianjanju zbog različitih tereta. Sila prianjanja omogućuje kotačima da zadrže kontakt s podlogom. Ona varira ovisno o stanju ceste, kao što su suha, mokra ili skliska površina. Funkcionalnost sustava se postiže integracijom hardverskih i softverskih komponenti.

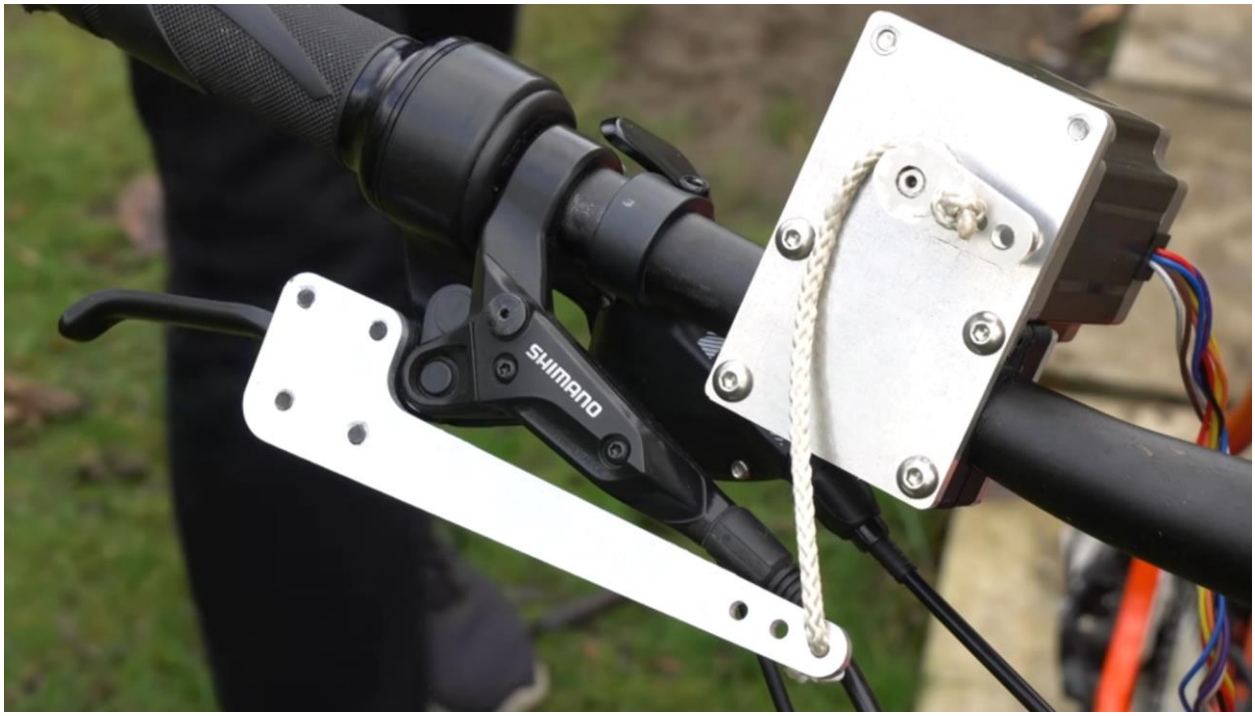
Problem stvara čovjek koji je van kontrole sustava. Pritiskom ručke za kočenje na upravljaču bicikla uzrokuje sajlu da povuče kočione čeljusti da stisnu kočione pločice uz diskove koji pritom stvaraju trenje i zaustavljaju bicikl kao što se vidi na slici 2.1. Takav princip kočenja je specifičan za disk kočnice koje se koriste. Ta sila usporava bicikl pretvarajući kinetičku energiju u toplinsku energiju.



Sl. 2.1. Prikaz diska kočnice

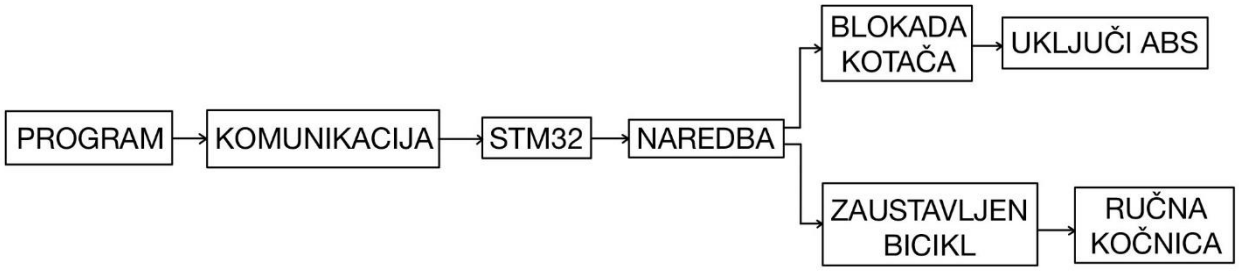
Problem je preslab koračni motor koji ne može svladati silu vozača koji pritišće kočnicu. On se rješava dodavanjem poluge na upravljaču bicikla. Poluga povećava silu što omogućuje da motor

povuče kočnicu iz vozačeve ruke. Ovim se smanjuje opterećenje na motor te se zadržava funkcionalnost. Primjer poluge je prikazan na slici 2.2.



Sl. 2.2. Princip poluge [1]

Za sustav s ABS-om koristi se magnet koji je montiran na žbicu i magnetski senzor koji kada dođe u kontakt s magnetom šalje signal koji se koristi za utvrđivanje brzine kotača. Prilikom blokade kotača sustav ABS-a se uključuje te poluga koja je montirana na upravljaču bicikla pojačava silu motora da otpusti ručku za kočenje. Ovim dodatkom se povećava kontrola nad vozilom i sigurniji zaustavni put neovisno o površini. Vrsta podloge utjeće na silu prianjanja koja se javlja između te površine i gume bicikla.

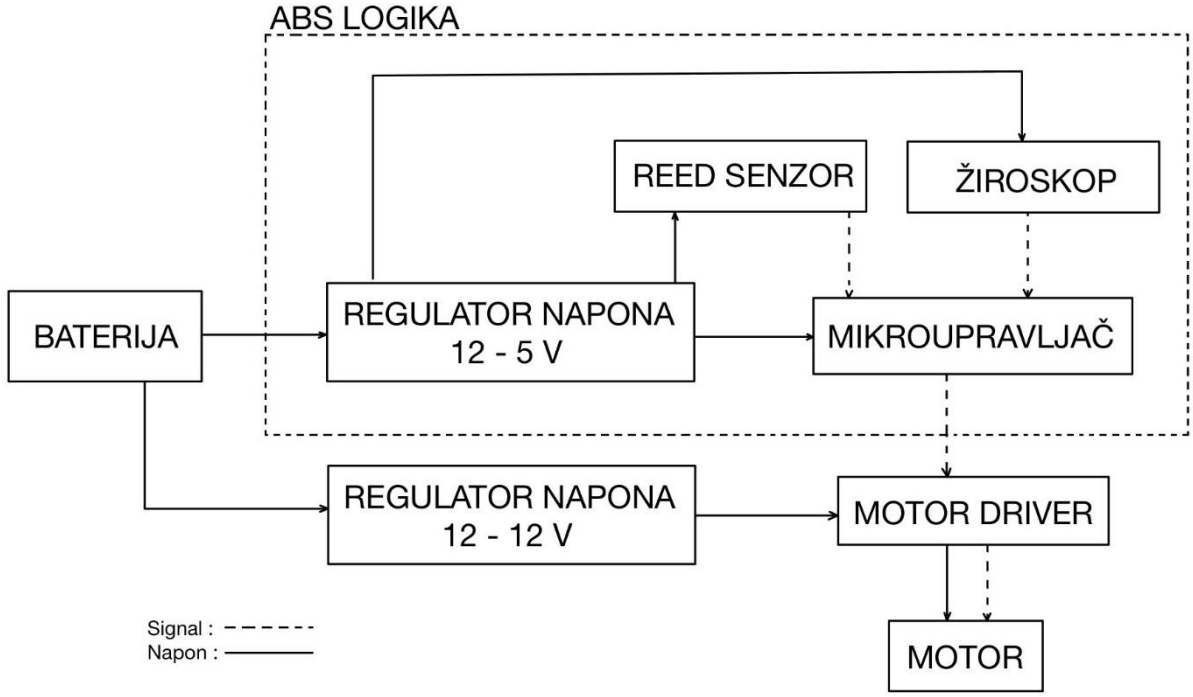


Sl. 2.3. Osvrt na sustav ABS-a, blokovski prikaz

Slika 2.3. prikazuje osvrt ABS sustava. Prikazuje korake rada ABS-a od programa koji se uključuje prekidačem pa sve do uključenja samog programa ABS-a ili do uključenja ručne kočnice.

2.2. Prijedlog sklopovskog rješenja

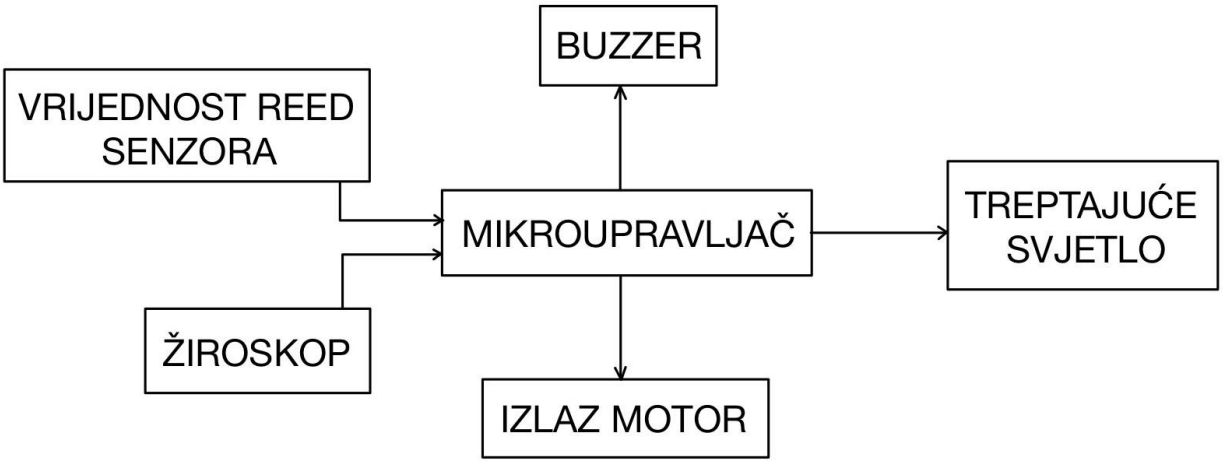
Blokovski prikaz sklopovskog prijedloga pokazuje prvi plan dijelova koji se koriste, njihov međusobni odnos i koncept sustava. Ideja je koristiti dva regulatora napona od kojih se jedan koristi za napajanje upravljača koračnog motora, a drugi za mikroupravljač, magnetski senzor i žiroskop. U kasnijim izvedbama je nadodan još jedan pretvarački modul, maknut žiroskop i zamijenjen s drugim magnetski senzorom radi jednostavnosti koda. Napajanje je pokazano punom crtom dok je signal isprekidana crta.



Sl. 2.4. Sklopovski prijedlog rješenja, blokovski prikaz

2.3. Prijedlog programskog rješenja

Programsko rješenje je pojednostavljeno radi čitljivosti i jednostavnosti izrade programa. U suštini podaci koje prima mikroupravljač su od magnetskog senzora i žiroskopa. Ti podaci se obrađuju u mikroupravljaču te ovisno o rezultatima događaju se određene funkcije. Pištalica, trepćuće svjetlo i motor reagiraju ovisno o signalima poslanim iz mikroupravljača. Proces je dosta linearan.



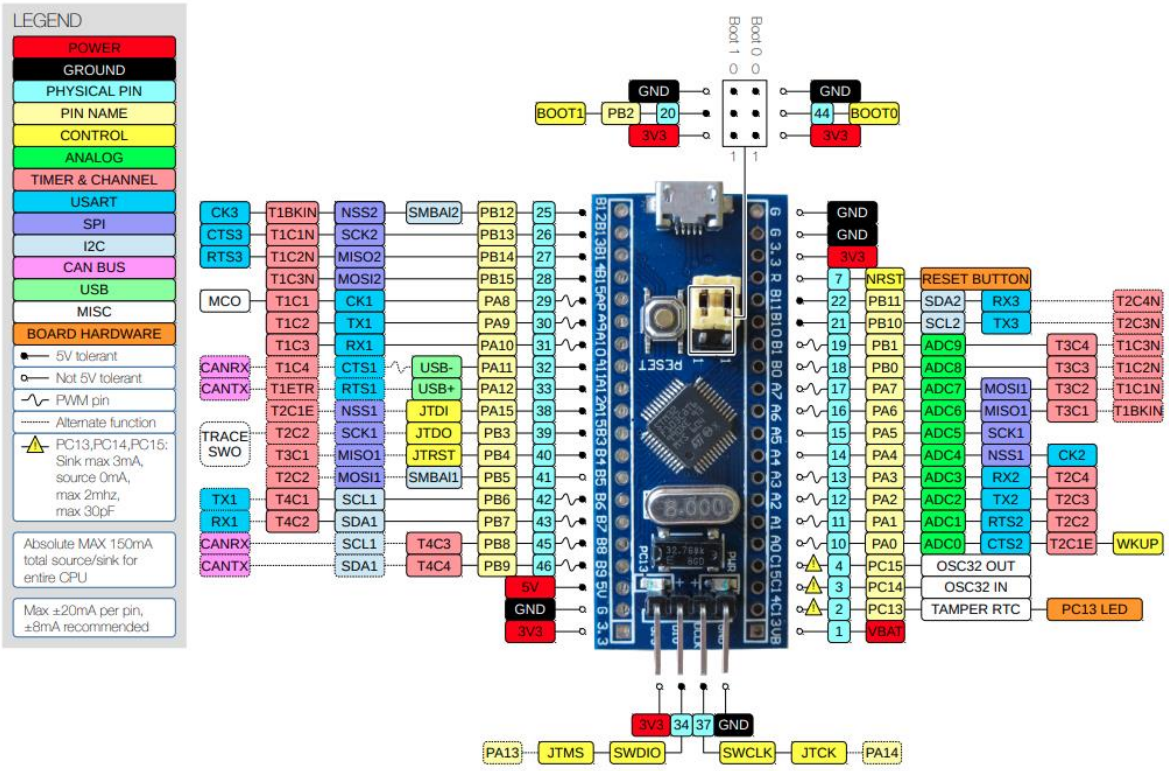
Sl. 2.5. Programski prijedlog rješenja, blokovski prikaz

3. REALIZACIJA ABS SUSTAVA KOČENJA ZA BICIKL

3.1. Korištene komponente i alati

STM32F103C8T6 mikroupravljač je razvojna pločica temeljena na ARM Cortex M3 mikroprocesoru, poznata po visokim performansama, maloj potrošnji energije i opsežnom skupu perifernih uređaja. STM32 se koristi za upravljanje komponenta i obradu nadolazećih podataka od senzora. Sadrži 10 analognih pinova i 37 digitalni I/O pinova. Za povezivanje s računalom koristi se ST-LINK V2.

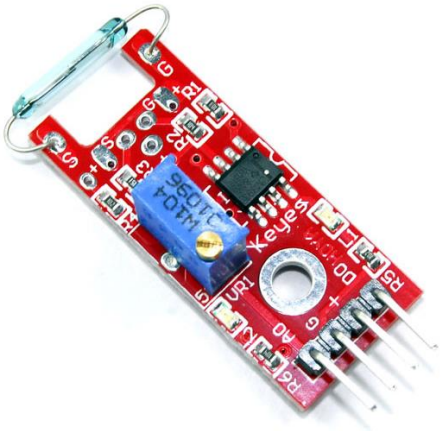
Koriste se dvije STM32 pločice zbog komplikacija u programu ABS-a no jedini problem je u konstrukcijskom djelu, a to je manjak prostor. Kućište u kojem treba stajati samo jedna STM32 pločica sada drži dvije. Zbog dobrog planiranja kućište se ne treba mijenjati već je dovoljnih dimenzija za obje pločice.



Sl. 3.1. STM32F103C8T6 Mikroupravljač (engl. STM32F103C8T6 *bluepill*) [2]

KY-025 magnetski senzor je mali električni prekidač koji se koristi za otkrivanje magnetskog polja. Magnetski prekidač se sastoji od fleksibilnih jezičaka izrađenih od magnetskog materijala koji su stavljeni u staklenu cijev. One se preklapaju, ali su odvojene malim razmakom. Primjena magnetskog polja koje generira magnet uzrokuje magnetiziranje oba

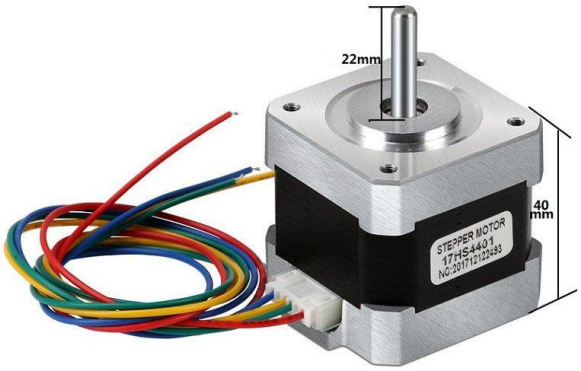
jezička te oni dolaze u kontakt. Magnetski senzor je montiran na ramu bicikla kod zadnjeg i prednjeg kotača, a magnet je postavljen na žbicu oba kotača ispred senzora. Svaki put kada magnet okine senzor on šalje signal. Pomoć tog signala se određuje brzina bicikla i blokada kotača.



Sl. 3.2. KY-025 Magnetski Senzor (engl. KY-025 *reed sensor*) [3]

17HS4401 koračni motor se kreće u diskretnim koracima, koji rezultiraju preciznijom kontrolom položaja. Motor ima kut koraka 1,8º koraka ili 200 koraka/okretaj s momentom držanja od 40 N.cm.

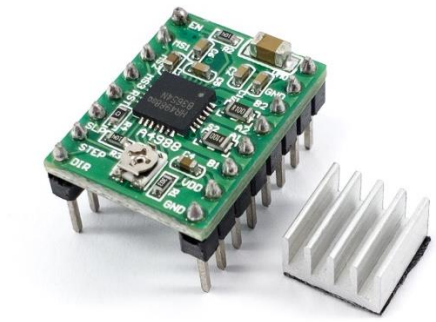
Motor posjeduje premalu snagu da sam savlada pritisak ruke na kočnicu. Zbog tog razloga je montiran na ramu u kombinaciji s polugom koja prenosi moment te tako svladava pritisak ruke.



Sl. 3.3. 17HS4401 Koračni Motor (engl. 17HS4401 *Stepper Motor*) [4]

A4988 upravljač koračnog motora se koristi zbog jednostavnosti upravljanja koračnim motorom. Za motor 17HS4401 su samo potrebna dva pina za kontrolu brzine i smjera. Slijed

impulsa određuje smjer vrtnje motora, frekvencija impulsa određuje brzinu, a broj impulsa koliko se motor okreće.

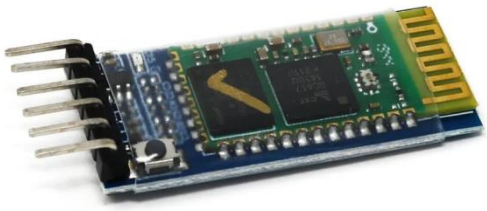


Sl. 3.4. A4988 Upravljač Koračnog Motora s Hladnjakom (engl. A4988 *Stepper Motor Driver Carier with Heat sink* [5]

LM2595HV DC/DC je napravljen za promjenu izlaznog napona u rasponu od 1,25 do 35 V. On je step-down pretvarač napona koji ima 8 pinova i izrađen je u paketu TO-263. Korištena su tri modula za regulaciju napon koji se dovodi drugim komponentama.

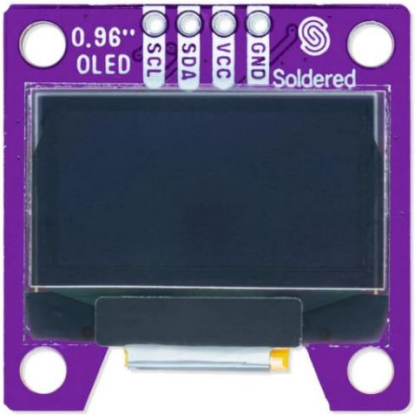
Sl. 3.5. LM25956S-ADJ DC/DC silazni Pretvarački Modul (engl. LM2596S-ADJ DC/DC *Step-down converter module*) [6]

HC-05 *Bluetooth* modul je napravljen za bežičnu komunikaciju. Omogućuje svim uređajima koji imaju serijski pristup komunikaciju putem *bluetooth*-a. Koristi se da mobitel može primati podatke o vremenu aktivnosti motora.



Sl. 3.6. Serijski *Bluetooth* HC-05 (engl. *Bluetooth Serial* HC-05) [7]

12C OLED Display služi za prikaz podataka. Ekran radi pri naponu od 3.3 – 5 volta. Dimenzije ekrana su 128 x 64 to jest 0.96 inča na kojem je prikazana trenutna brzina bicikla te ikone ABS-a i automobila u slučaju blokade kotača. Ikone se pojavljuju da signaliziraju uključen ABS sustav.



Sl. 3.7. 0.96 inč 12C OLED ekran (engl. 0.96 12C OLED *display*) [8]

ST-LINK V2 je standardni uređaj za razvoj softvera za STM32. Koristi se tijekom razvojne faze za učitavanje *firmwera* na STM32 i za otklanjanje pogrešaka.

Sl. 3.8. ST-LINK V2 Programer za Preuzimanje Simulatora (engl. ST-LINK V2 *Simulator Download Programmer*) [9]

Piezo element je uređaj koji daje zvučni signal. Uprojektnu signalizira uključen ABS.

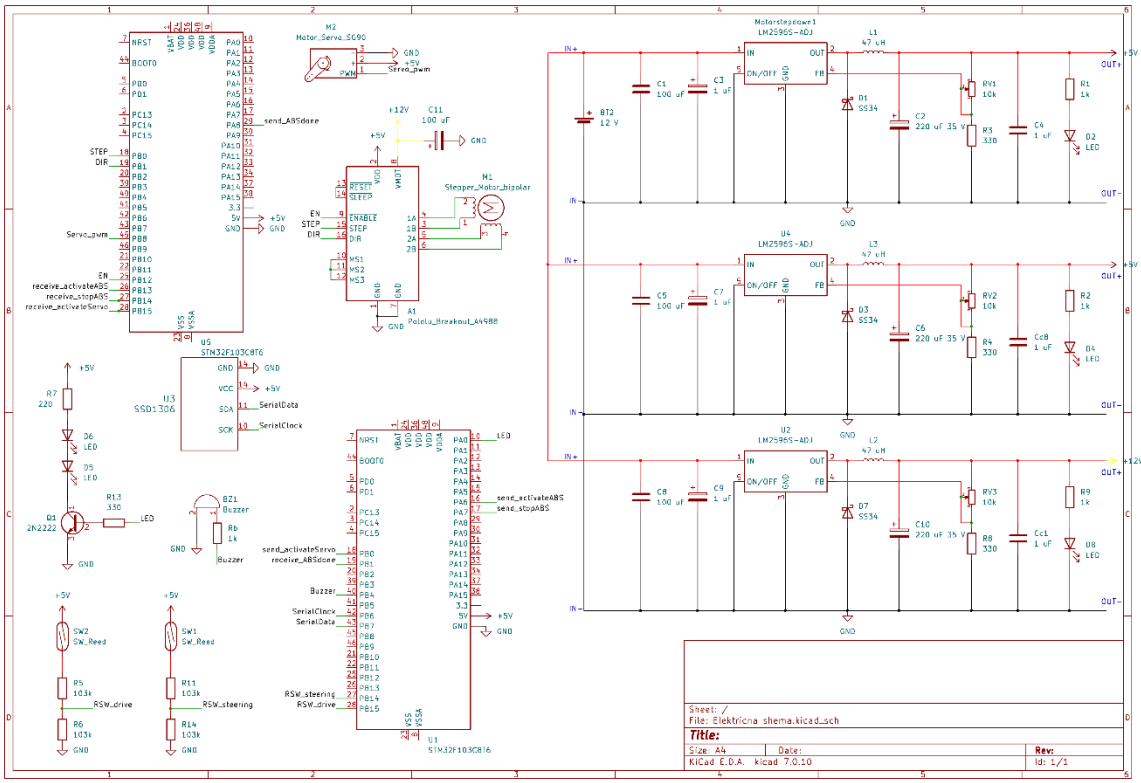
Sl. 3.9. Piezo element - Pištalica (engl. *Piezo element – Buzzer*) [10]

3.2. Realizacija konstrukcijskog i sklopovskog rješenja

Sustav ABS-a na biciklu se sastoji od mnogo komponenti od kojih je glavni mikroupravljač STM32F103C8T6 na razvojnoj pločici *bluepill.* Preko njega se upravlja sa svim ostalim komponentama koje su bitne za rad ABS sustava. Mikroupravljač je programiran u PlatformIO razvojnom okruženju.

Zbog komplikacija moraju se koristiti dva STM32 mikroupravljača koja koriste nadređeno/podređenu komunikaciju. Također se montiraju dvije odvojene bušene pločice.

Podređena pločica se sastoji od servomotora, koračnog motora i upravljača koračnog motora dok je na nadređenoj pločici povezano sve ostalo. Na podređen STM32 mikroupravljač je spojen servomotor koji s tipkalom upravlja ručnom kočnicom. Jedan pritisak na tipkalo uključuje ručnu kočnicu dok ju dva isključuju. Za upravljanje koračnim motorom koristi se upravljač koračnog motora koji ima 5 različitih rezolucija mikrokoraka do 1/16 koraka što je dovoljno precizno za zamišljen sustav ABS-a. Koračni motor sam nije dovoljan da svlada silu ruke na kočnicu, ali u kombinaciji s polugom može svladati silu ruke i tako otpustiti kočnicu. Za paljenje motora sam program mora prvo poslati signal da dolazi do blokade kotača. Za dolazak do tih podataka koriste se 2 magnetska senzora, jedan montiran na prednji, a drugi na stražnji dio osovine kod kotača. Signali koje šalju 2 magnetska senzora se obrađuju u nadređen mikroupravljač te se na 0.96 inč OLED ekranu prikazuje trenutna brzina i ako je došlo do blokade kotača. Sve to se šalje na podređenu pločicu koja upravlja koračnim motorom i servomotorom. Također na nadređenoj pločici se nalazi pištalica koji nam daje zvučnu naznaku da dolazi do blokade kotača.



Sl. 3.10. Električna shema ABS sustava nacrtana u programu KiCad

U prvoj iteraciji sheme na mikroupravljač je spojena MPU6050 jedinica koja se koristi kao žiroskop, ali nakon testiranja se jednostavniji pokazuje magnetski senzor koji se montira na okvir bicikla kod prednjeg i zadnjeg kotača. Kako bi mogao slati signal nasuprot senzora na žbicu se postavio mali magnet koji okidanjem senzora šalje signal na STM32 pločicu.

Također dolazi do nadodavanja drugog STM32 mikroupravljača. Dva mikroupravljača koriste nadređeno/podređenu komunikaciju. Komplikacije nastaju kada TimerInterrupt prekida normalan rad koračnog motora, a to se najlakše rješava dodatkom drugog STM32. Zbog toga su se ostale komponente moraju rasporediti na dvije bušene pločice. Kućište se ne mijenja, ali dodatkom komponenata je otežan postupak testiranja. Kućište se treba otvarati i pločice se moraju micati svaki puta kada se mijenja kod.

Najviše vremena zauzima lemljenje komponenti gdje je promijenjena prva pločica kada se miče žiroskop i mijenja početna shema, te dodatak drugog mikroupravljača koji zahtijeva još jednu bušenu pločicu.

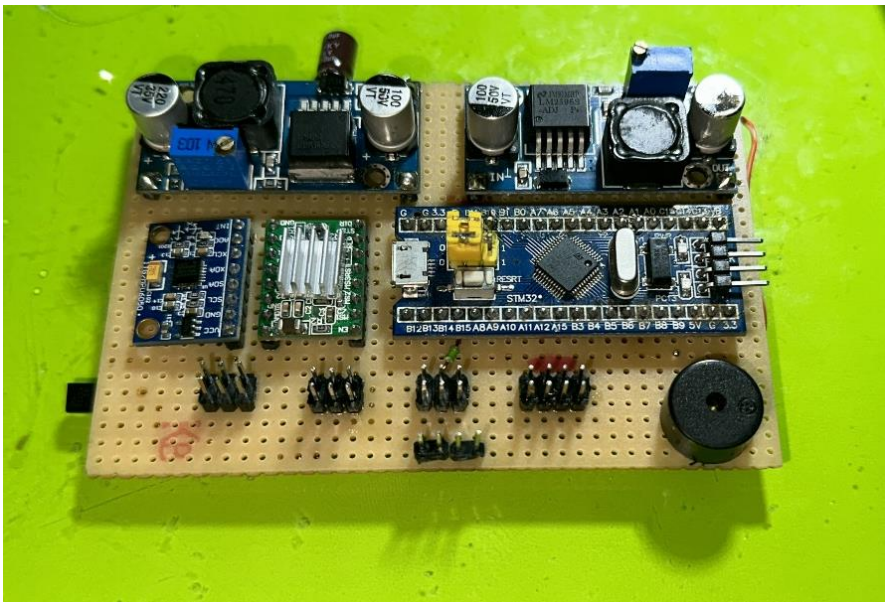
3.2.1. Koraci pri izradi ABS-a

Prvi korak je konstrukcija koja se montira na bicikl tako da ne smeta vozaču tijekom vožnje, a da ima dovoljno prostora da se tada jedna pločica može postaviti u kućište. Dizajn je vrlo jednostavan gdje se koračni motor s polugom, tipkalom i 0.96 inč OLED ekranom montira na upravljač bicikla. Komponente su žicama povezane s kutijom u kojoj stoje dva STM32 mikroupravljača i izvor napajanja koji su smješteni u okvir bicikla. Sama pločica je pričvršćena u kućište. Magnetski senzor se nalazi na zadnjem djelu okvira montiran nasuprot magneta koji se nalazi na žbici i svijetlo koje je ispod sjedala kao što je prikazano na Slici 3.4. Ova konstrukcija ostaje nepromijenjena. Sve je napravljeno za premještanje s bicikla na bicikl bez kozmetičke štete.

Sl. 3.11. Konstrukcija montirana na bicikl

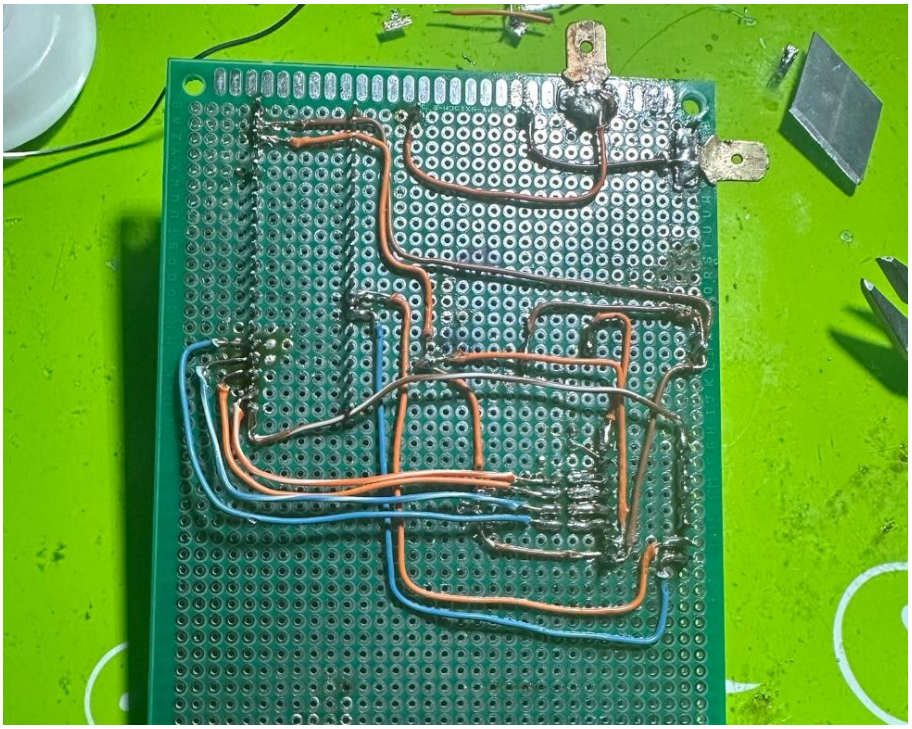
Nakon završetka konstrukcije započinje testiranje i spajanje komponenti na bušenu pločicu. Trenutačno sustav ABS-a tj. program ne funkcionira u potpunosti, ali se postupno testira i razvija kako se nadodaju komponente na bušenu pločicu.

Prva iteracija je ista kao prijedlog sklopovskog rješenja. Daljnje testiranje pločice sa slike 3.12 pokazuje da treba promijeniti komponente i da nije vrijeme da se pločica montira u kućište na biciklu.



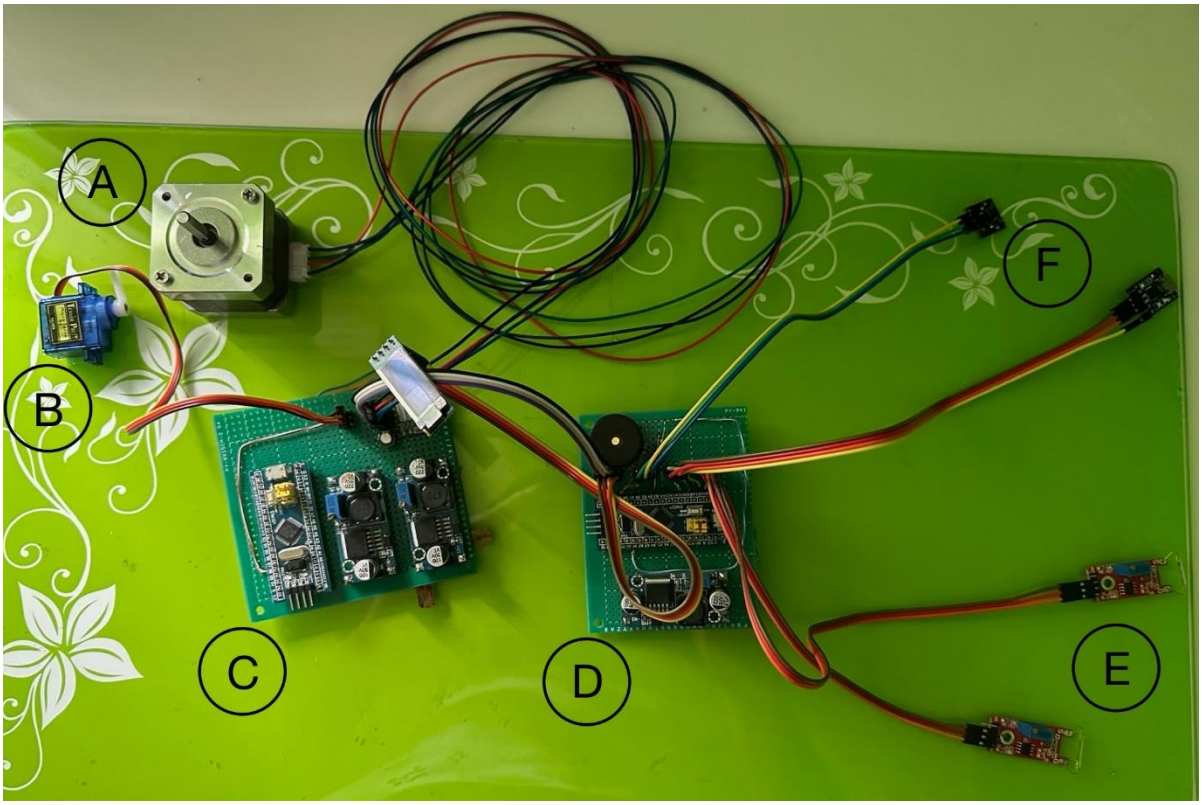
Sl. 3.12. Prva iteracije pločice prema prijedlogu sklopovskog rješenja

U ovom stadiju dolazi do problema te se nadodaje drugi STM32 i komponente se razdvajaju na dvije bušene pločice. Shema se također mijenja za olakšanje procesa lemljenja. Slika 3.13. prikazuje novu bušenu pločicu i napredak koji se postiže nakon prvog prototipa. Kontakti su preciznije zalemljeni, komponente su drugačije posložene, sama pločica je urednija. Bitno za spomenuti su žice koje se mjenjaju iz licnastiih u puni presjek.



Sl. 3.13. Proces lemljenja komponenata na bušenu pločicu

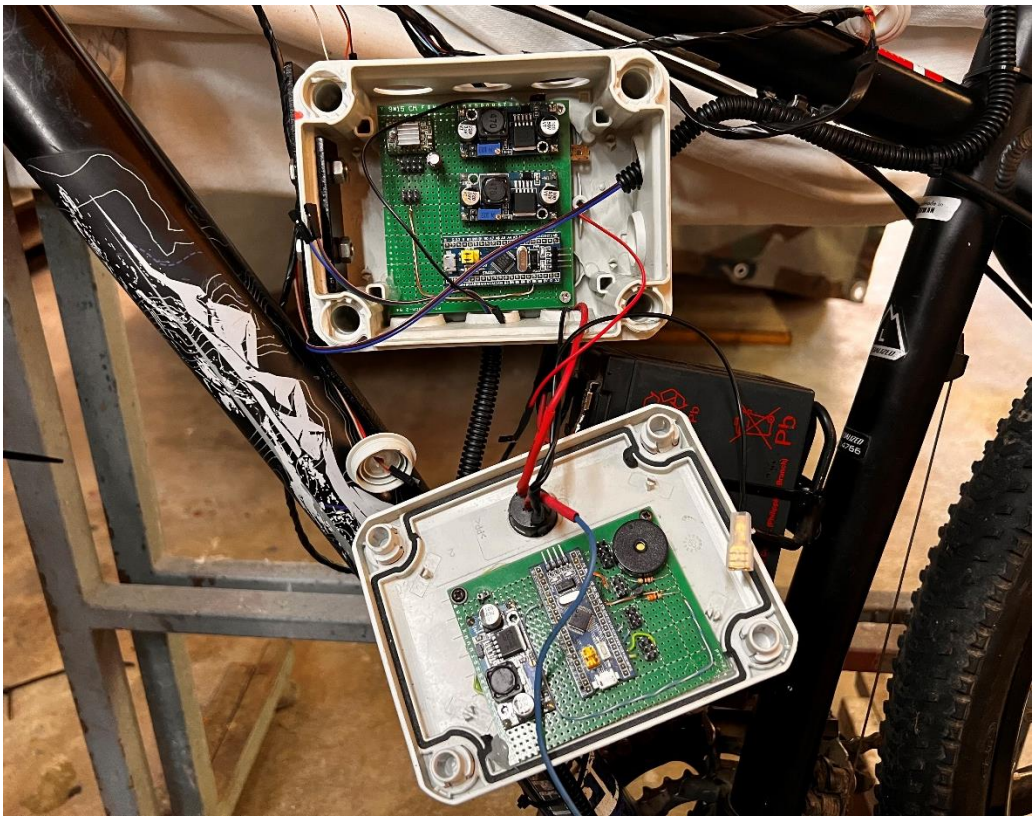
Sve komponente rade kada se sustav testira prije montiranja na bicikl, no tijekom procesa montiranja neke se spaljuju dok druge djelomično rade. STM32 podređena pločica radi kada je povezana na računalo putem ST-LINK-a, ali ne želi kada se spoji na napajanje koje je montirano na biciklu. Utvrđuje se da je greška u pločici te se jedan od pretvaračkih modula podešuje na manji napon. To utječe na servomotor koji je sada pod manjim naponom. Servomotor radi ispravno, ali manjak napona ga usporava i tijekom samog okretanja poskakuje.



Sl. 3.14. A – Koračni motor; B – Servomotor; C – Podređena pločica; D – Nadređena pločica; E – Magnetski senzori; F – UART bridge i Tipkalo

Ovo je krajnje rješenje, sve komponente su zalemljene i sve što ostaje je montiranje u kućište i testiranje. Iz slike 3.15. se vidi da je kućište dovoljno veliko za dvije bušene pločice, ali da ima manjak mjesta za spajanje ST-LINK. Na ovom aspektu se može poraditi, ali promjene u kodu su minimalne pa nema potrebe za većim kućištem.

Sve komponente su pričvršćene i osigurane s vrućim ljepilom, silikonom i plastičnim vezicama (engl. *zip tie*). Vruće ljepilo se koristi za pričvršćivanje i izoliranje žica koje se ne mogu s izolir trakom.



Sl. 3.15. Prikaz gotovog sklopa ABS-a montiranog u kućište bicikla

Slika 3.16. prikazuje napredak konstrukcije na upravljaču bicikla. Poluga je skraćena da ne smeta vozaču tijekom vožnje. Koračni motor je pričvršćen vijcima za metalnu pločicu koja ga drži za upravljač. Uzica ga povezuje s polugom koju privlači kada se aktivira ABS. 0.96 inč OLED ekran je pričvršćen vrućim ljepilom jednako kao servomotor.

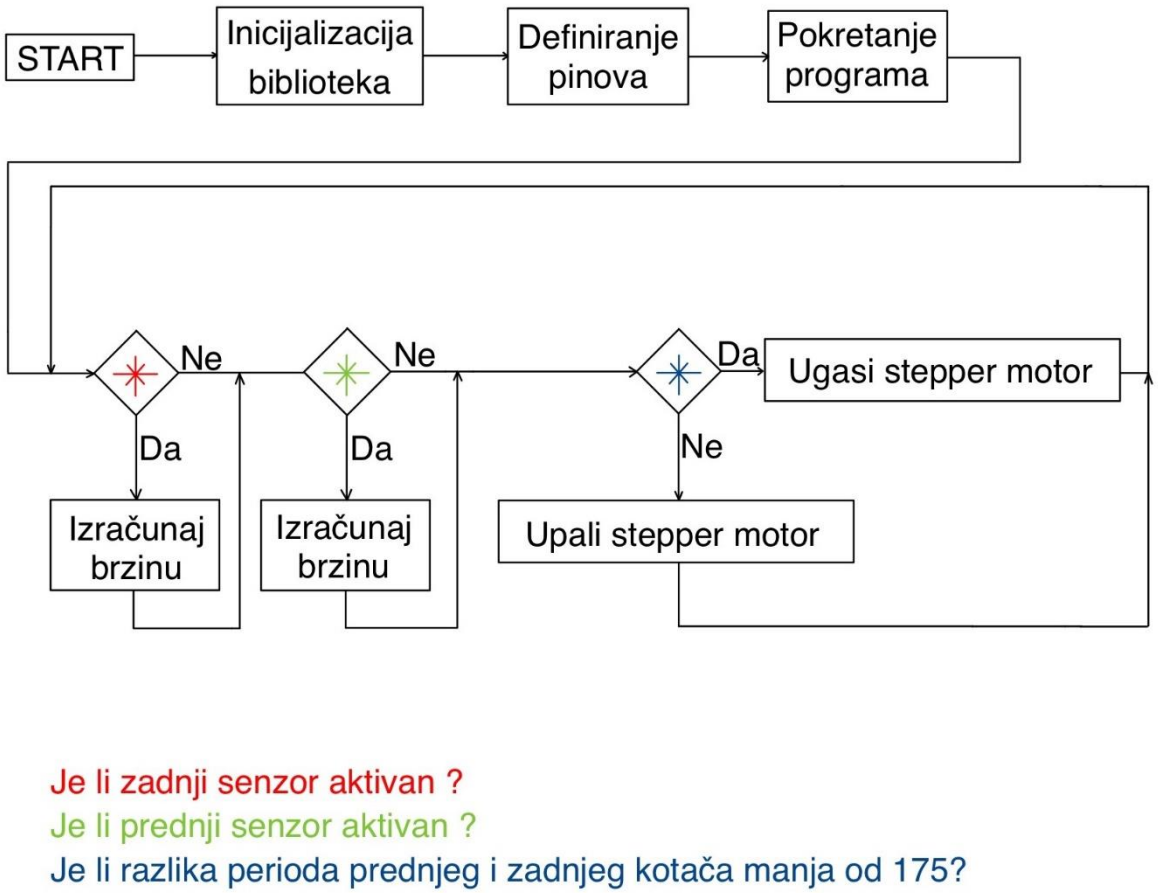
Sl. 3.16. Komponente pričvršćene na upravljač bicikla

3.3. Realizacija programskog rješenja

Za realizaciju ABS sustava koristi se program VSC (engl. *Visual Studio Code*) s ekstenzijom PlatformIO. Sheme su nacrtane u progamu KiCad, a ikone koje se pojavljuju na 0,96 inč OLDE ekranu u programu Photopea.

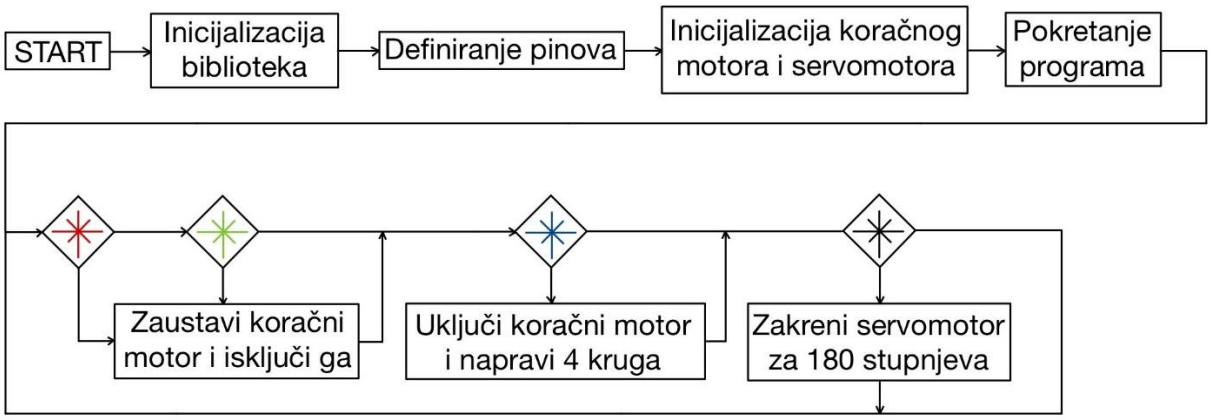
Postoje nadređena i podređena STM32 pločica. Nadređena pločica sadrži svu logiku ABS programa i većina komponenata je upravljana istom. Podređena STM32 pločica sadrži logiku za servomotor i koračni motor.

Prvi korak je inicijalizacija biblioteka, zatim treba definirati varijable i fizičke pinove. Kada se pokrene program on se okreće u petlji koja konstantno provjerava vrijednosti i prema njima određuje je li potrebno uključivati ABS kao što se vidi na slici 3.17. Istovremeno komunicira s podređenom pločicom od koje dobiva podatke za servomotor i koračni motor.



Sl. 3.17. Pojednostavljena logika nadređenog programa

Program koji se stavlja na podređenu pločicu započinje kao i program na nadređenoj pločici. Fokus mu je komunikacija s nadređenom pločicom, gdje on prvenstveno služi kao pasivan modul koji prati upute. Kao što je prikazano na slici 3.18. prvi korak je inicijalizacija biblioteka, a zatim definiranje svih potrebnih varijabli koje uključuju zastave (eng. *flag*) i korištene fizičke pinove. Zatim se započinje program koji funkcionira kao petlja tj. konstantno provjerava podatke i određuje što radi s njima. Bitno za spomenuti je korištenje istog vremena za zastave stop i start ABS. Korištenje drugačijeg vremena ponekad rezultira u preklapanju.



Je li koračni motor dugo neaktivan?
sendStopABS = 1
sendActiveABS = 1
sendActiveServo = 1

Sl. 3.18. Logika podređenog programa

4. TESTIRANJE I REZULTATI

4.1. Metodologija testiranja

Pri dizajniranju ABS sustava za bicikl, uzima se u obzir smještaj napajanja i komponenti u ograničenom prostoru. Manji prostor zahtjeva preciznije smještanje komponenti te zbog tog manjeg prostora su komponente zalemljene na šo manju pločicu. Samo napajanje se nalazi na mjestu stalka za boce. Nalazi se na mjestu koje je najpraktičnije za povezivanje s ostalim komponentama. Upravljač koračnog motora treba dobiti napajanje od 12 V, a za ostale komponente se napon regulira na 5 V. Greška u namještanju napona je rezultira spaljenim komponentama.

Za početak testiranja, hardver se treba spojiti i treba se dodati najnoviji firmver na STM32 mikroupravljač. Testiranje se odvija u dva dijela:

- Testiranje konstrukcije
- Testiranja s uključenim i isključenim ABS-om

Testiranje konstrukcije je brz proces koji se sastoji od provjeravanja stabilnosti elemenata koji se montiraju na okvir bicikla. Elementi se čvrsto zatežu vezicama i vrućim ljepilom da ne dođe do raspada.

Testiranje je duži proces koji zahtjeva da se bicikl odvede na prostor koji ima asfaltirani put i pješčanu stazu. Bicikl se testira s uključenim i isključenim ABS-om te se uspoređuju rezultati. Da rezultati budu precizniji koristi se metar s kojim se mjeri udaljenost od trenutka kada se povuče ručica kočnice do potpunog stajanja bicikla. Kredom se povlači crta koja zabilježava početnu crtu gdje vozač povlači ručicu kočnice do crte koja označava mjesto gdje bicikl staje.

4.2. Rezultati testiranja

4.2.1. Testiranje konstrukcije

Probna vožnja služi za utvrđivanje čvrstoće i pouzdanosti konstrukcije. Zatvoren poklopac je pričvršten za kućište kao što se vidi na slici 4.1. Jedina primjedba konstrukciji je HC-05 *Bluetooth* modul koji se mora ostaviti s vanjske strane zbog manjka prostora.

Slika 4.1. Konstrukcija kućišta

4.2.2. Testiranja s uključenim i isključenim ABS-om

Konstrukciju i sustav ABS-a je potrebno testirati na raznolikim podlogama. Rezultati tih testiranja su uspoređeni i vidi se da postoje poboljšanje kada je ABS uključen. Za preciznost rezultata brzina je podjednaka na svim testiranjima i iznosi 15 km/h. Podloga prilikom testiranja se ne mijenja sve dok se ne obavi 20 testova.

Preko formula se računa vrijeme koje je potrebno da bicikl stane:

$$s = \frac{1}{2} * a * t^2 \tag{4-1}$$

gdje su:

s – prijeđen put [m];

a – ubrzanje bicikla [m/s^2];

t – proteklo vrijeme [s]

$$v = v_0 - a * t \tag{4-2}$$

gdje su:

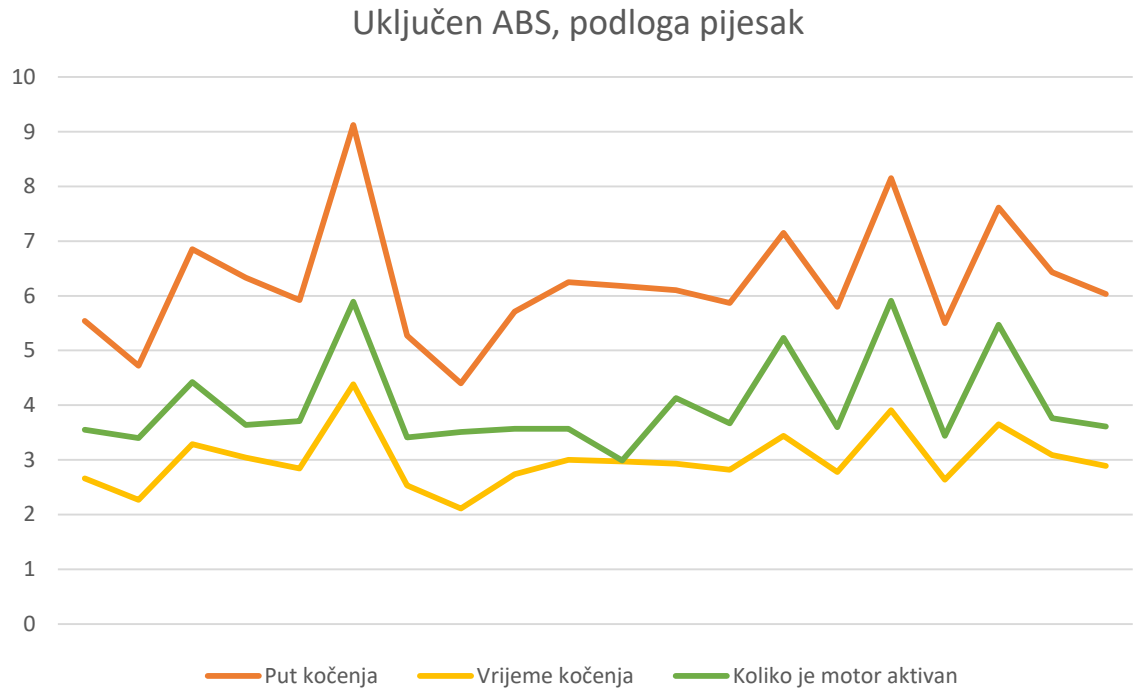
v – krajnja brzina [m/s];

v_0 – brzina na početku [m/s]

Tab. 4.1. ABS program testiran na podlozi pijesak

Broj mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Put kočenja [m]	5,54	4,72	6,85	6,33	5,92	9,12	5,27	4,40	5,71	6,25
Vrijeme kočenja [s]	2,66	2,27	3,29	3,04	2,84	4,38	2,53	2,11	2,74	3,00
Koliko je motor aktivan [s]	3,55	3,40	4,42	3,64	3,71	5,89	3,41	3,51	3,57	3,57

Broj mjerenja	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
Put kočenja [m]	6,18	6,10	5,87	7,15	5,80	8,15	5,50	7,61	6,43	6,03
Vrijeme kočenja [s]	2,97	2,93	2,82	3,44	2,78	3,91	2,64	3,65	3,09	2,89
Koliko je motor aktivan [s]	2,99	4,13	3,67	5,23	3,60	5,91	3,44	5,47	3,76	3,61

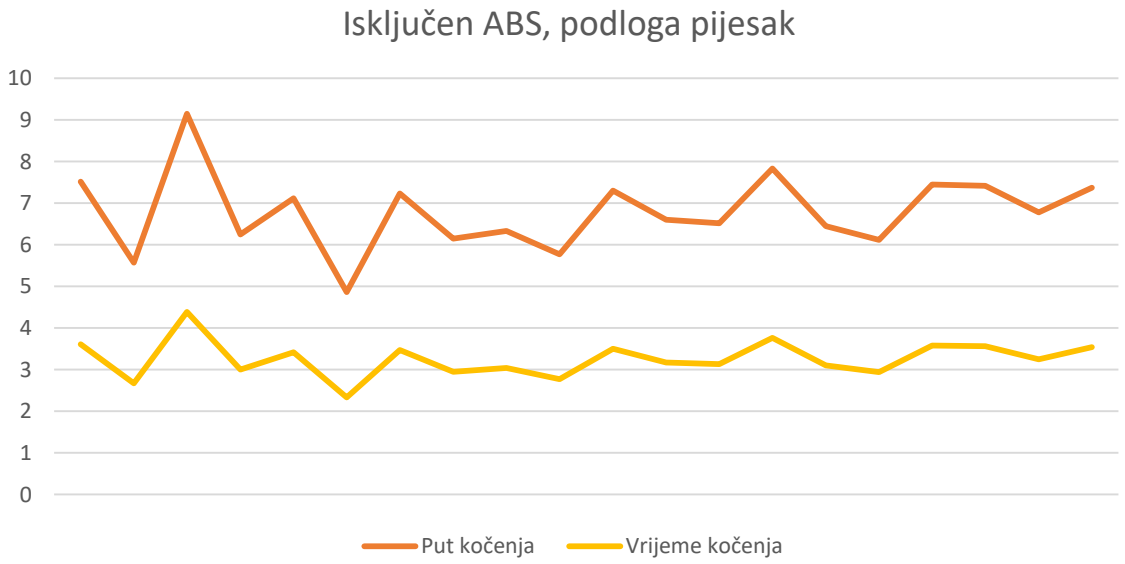


Sl. 4.2. Tablica 4.1. prikazana grafički

Tab. 4.2. Testiranje na podlozi pijesak bez ABS-a

Broj mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Put kočenja [m]	7,52	5,57	9,15	6,25	7,12	4,86	7,23	6,15	6,33	5,77
Vrijeme kočenja [s]	3,61	2,67	4,39	3,00	3,42	2,33	3,47	2,95	3,04	2,77

Broj mjerenja	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
Put kočenja [m]	7,30	6,60	6,52	7,83	6,45	6,12	7,45	7,42	6,78	7,37
Vrijeme kočenja [s]	3,50	3,17	3,13	3,76	3,10	2,94	3,58	3,56	3,25	3,54



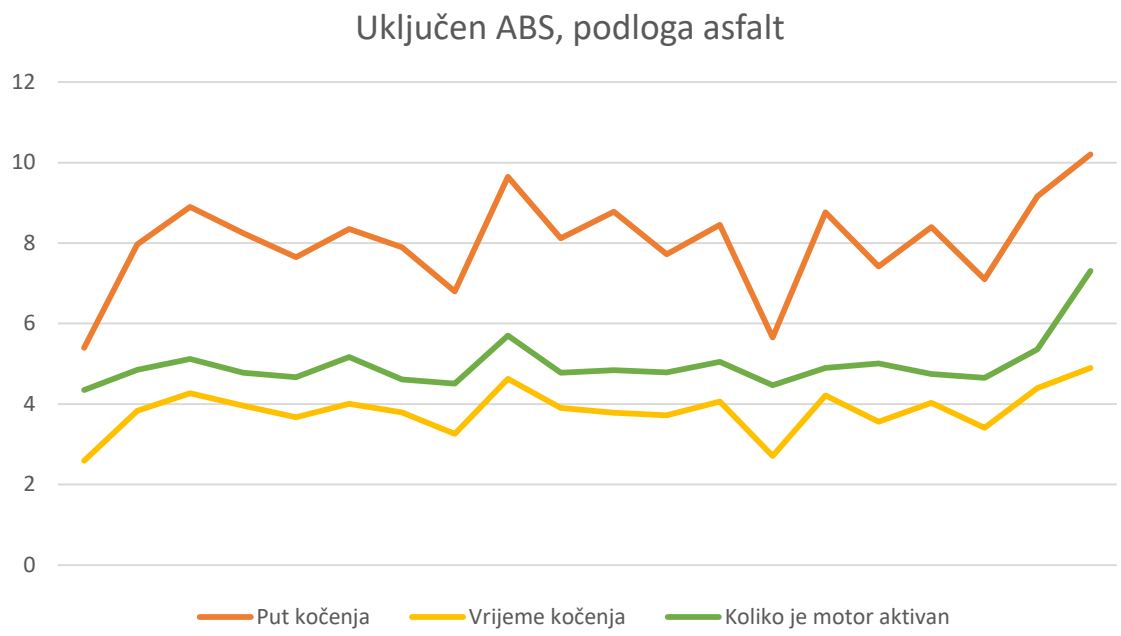
Sl. 4.3. Tablica 4.2. prikazana grafički

Testiranje na prvoj podlozi pokazuje da ABS pomaže tijekom kočenja. Vrijeme potrebno da bicikl stane se smanjuje jednako kao i put kočenja. Ako se usporedi najduži put kočenja iz tablice 4.1 i 4.2 može se primijetiti da je prosječni put kočenja manji kad je uključen ABS. Ovaj rezultat je očekivan i može potvrditi da sustav radi ispravno.

Tab. 4.3. ABS program testiran na podlozi asfalt

Broj mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Put kočenja [m]	5,40	7,97	8,90	8,25	7,65	8,35	7,90	6,80	9,65	8,12
Vrijeme kočenja [s]	2,59	3,83	4,27	3,96	3,67	4,01	3,79	3,26	4,63	3,90
Koliko je motor aktivan [s]	4,35	4,85	5,12	4,78	4,67	5,17	4,61	4,51	5,70	4,78

Broj mjerenja	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
Put kočenja [m]	8,78	7,72	8,45	5,65	8,76	7,42	8,40	7,10	9,16	10,2
Vrijeme kočenja [s]	3,78	3,72	4,06	2,71	4,21	3,56	4,03	3,41	4,40	4,90
Koliko je motor aktivan [s]	4,84	4,79	5,05	4,47	4,90	5,01	4,75	4,65	5,36	7,31

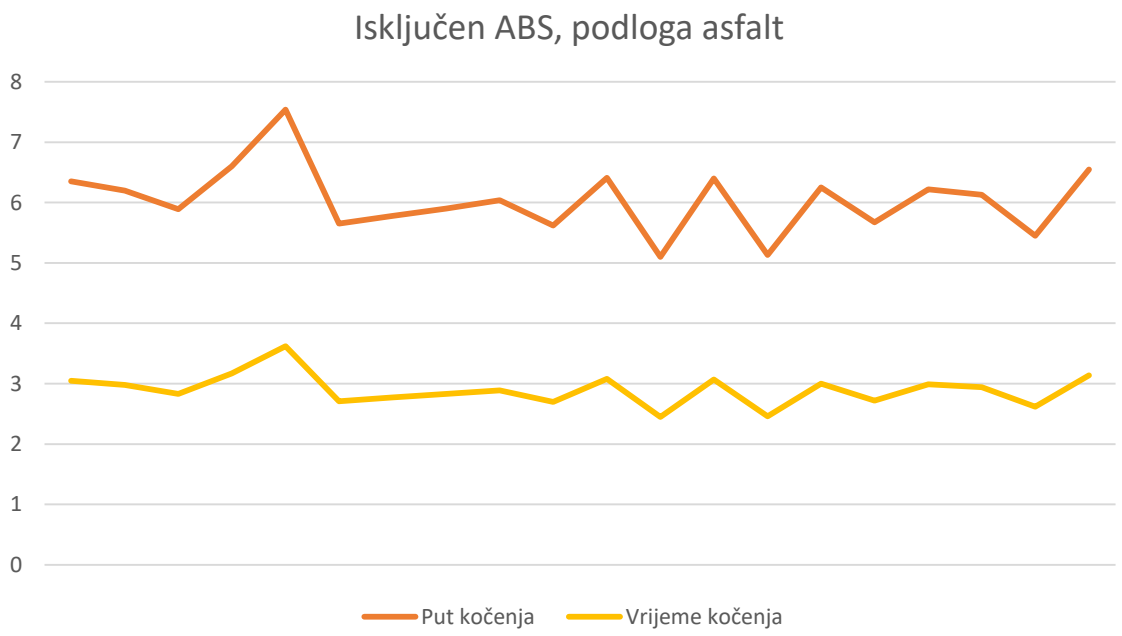


Sl. 4.4. Tablica 4.3. prikazana grafički

Tab. 4.4. Testiranje na podlozi asfalt bez ABS-a

Broj mjerenja	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Put kočenja [m]	6,35	6,20	5,89	6,60	7,54	5,65	5,78	5,90	6,02,	5,62
Vrijeme kočenja [s]	3,05	2,98	2,83	3,17	3,62	2,71	2,77	2,83	2,89	2,70

Broj mjerenja	11.	12.	13.	14.	15.	16.	17.	18.	19.	20.
Put kočenja [m]	6,41	5,10	6,40	5,13	6,25	5,67	6,22	6,13	5,45	6,55
Vrijeme kočenja [s]	3,08	2,45	3,07	2,46	3,00	2,72	2,99	2,94	2,62	3,14



Sl. 4.5. Tablica 4.4. prikazana grafički

Iz rezultata je vidljivo da sustav ABS-a funkcionira, ali nije jednako efikasan kao i na pijesku. Za bolje usporedive rezultate računa se srednja vrijednost koja daje bolji uvid u rezultate.

$$s_{sr} = \frac{1}{n}\sum_{i=1}^{n} si \qquad (4\text{-}3)$$

gdje su:

s_{sr} – srednja vrijednost prijeđenog puta [m];

n – broj mjerenja;

si – zbroj n mjerenja [m]

Uključen ABS, podloga pijesak: $s_{sr} = 6{,}25\ m$

Isključen ABS, podloga pijesak: $s_{sr} = 7{,}13\ m$

Uključen ABS, podloga asfalt: $s_{sr} = 7{,}97\ m$

Isključen ABS, podloga asfalt: $s_{sr} = 6{,}04\ m$

5. ZAKLJUČAK

Cilj je napraviti konstrukciju ABS-a koja se može upotrebljavati na dnevnoj bazi i koja je pouzdana. Za ovaj pothvat treba potrošiti vrijeme na planiranje i proučavanje. U ovoj fazi je važno istražiti postojeće sustave, dostupne komponente, mogućnosti napajanja i upravljanja motora. Predznanje stečeno na fakultetu ubrzava proces dizajna i programiranja. Greške su nastale kao rezultat manjka informacija i neznanja komponenata.

Nastavak problema tijekom realizacije konstrukcije pokazuje kako pokušaji štednje na ključnim komponentama mogu dovesti do dodatnih komplikacija u radu. Kupnja skupljih komponenata tj. jačeg koračnog motora znači da poluga nije potrebna. Tijekom montiranja na bicikl dolazi do grešaka zbog kojih su neke komponente ostaju oštećene, a druge se moraju zamijeniti. Unatoč tome konstantno otvaranje kućišta zbog prebacivanja podataka na STM32 mikroupravljač postaje izazov. Ti izazovi ističu važnost precizne i pažljive montaže.

Tijekom testiranja ABS sustava na biciklu, uspjeli su se postići vidljivi rezultati. Izmjene napravljene tijekom razvoja imaju značajan utjecaj na te rezultate. Funkcionalnost programa i fizičkih komponenti je prikazana rezultatom testiranja ABS sustava. Vještine i znanja stečena na radu projekta služe za budućnost.

LITERATURA

[1] YouTube, DIY Anti-lock Braking System [online], dostupno na: https://www.youtube.com/watch?v=iiRpGHiKHJ0&t=265s , pristuo 3.6.2024.

[2] Makershop.de, STM32F103C8T6 Mikroupravljač [online], dostupno na: https://www.makershop.de/plattformen/mikrokontroller/stm32f103c8t6-board/ , pristup 3.6.2024.

[3] AliExpress, KY-025 Magnetski Senzor [online], dostupno na: https://www.aliexpress.com/item/32951193666.html , pristup 3.6.2024.

[4] AliExpress, 17HS4401 Koračni Motor [online], dostupno na: https://www.aliexpress.com/item/1005003210970428.html , pristup 3.6.2024.

[5] AliExpress, A4988 Upravljač Koračnog Motora s Hladnjakom [online], dostupno na: https://www.aliexpress.com/item/1005003765818545.html , pristup 3.6.2024.

[6] AliExpress, LM2596S-ADJ DC/DC korak-dolje Pretvarački Modul [online], dostupno na: https://www.aliexpress.com/item/1005004536447339.html , pristup 3.6.2024.

[7] Makershop.de, Serijski *Bluetooth* HC-05 [online], dostupno na: https://www.makershop.de/module/kommunikation-module/hc-05-bluetooth/ , pristup 3.6.2024.

[8] Soldered, 0.96 inč 12C OLED ekran [online], dostupno na: https://soldered.com/hr/proizvod/ekran-oled-i2c-bijeli-0-96-ssd1306/ , pristup 3.6.2024.

[9] AliExpress, ST-LINK V2 Programer za Preuzimanje Simulatora [online], dostupno na: https://www.aliexpress.com/item/32792513237.html , pristup 3.6.2024.

[10] ElectronIC Center, Piezo element – Pištalica [online], dostupno na: https://electronic-center.hr/piezo-element-22mm.html , pristup 3.6.2024.

[11] LM2596 Simple Switcher Istosmjerni pretvarač, Texas Instruments, 2024., dostupno na: https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1709180842472&ref_url=https%3A%2F%2Fwww.ti.com%2Fproduct%2FLM2596 , pristup 3.6.2024.

[12] STM32F103C8T6 Tehnička specifikacija [online], STMicroelectronics, 2007., dostupno na: https://www.alldatasheet.com/datasheet-pdf/pdf/201596/STMICROELECTRONICS/STM32F103C8T6.html , pristup: 3.6.2024.

[13] RM0008 Referentni priručnik [online], STMicroelectronics, 2021., dostupno na: https://www.st.com/resource/en/reference_manual/rm0008-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf , pristup: 3.6.2024.

[14] 17HS4401 Koračni motor Tehnička specifikacija [online], MotionKing, dostupno na: https://pdf1.alldatasheet.com/datasheet-pdf/view/1131976/MOTIONKING/17HS4401.html ,pristup 306.2024.

[15] Servomotor SG90 Tehnička specifikacija [online] , dostupno na: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf , pristup 3.6.2024.

[16] DatasheetCatalog.com, Dioda Tehnička specifikacija [online], dostupno na: https://soldered.com/productdata/2015/02/Soldered_mXrurvs_datasheet.pdf , pristup 3.6.2024.

[17] How To Mechatronics, Stepper Motor and Arduino – The Ultimate Guide [online], 2022., dostupno na: https://www.youtube.com/watch?v=7spK_BkMJys , pristup 3.6.2024.

[18] upir, Drive in Style: Arduino Gear Indicator (full tutorial) [online], ožujak 2024., dostupno na: https://www.youtube.com/watch?v=QixtxaAda18 , pristup 3.6.2024.

SAŽETAK

Podjela na segmente kao što su planiranje i istraživanje sve do izrade konstrukcije ABS-a je prikazano. Cijeli proces izrade od samog planiranja sklopovskog rješenja koje se je razvijalo kako su se dodavale nove komponente. Sklopovsko rješenje je u suštini ostalo nepromijenjeno. Konstrukcija je usko vezana uz njega te se nije mogla započeti dok se prvi prototip nije napravio. Taj prototip je pokazao greške i moguća unaprjeđenja koja se mogu napraviti kako bi krajnje rješenje bolje funkcioniralo. Ta unaprjeđenja se vide u urednosti i preciznosti tijekom lemljenja, promjenom licnastih žica u one punog presjeka i promjena senzora kako bi se pojednostavio program. Završni korak je testiranje koje je pokazalo predviđene rezultate. Bicikl je bolje funkcionirao s ABS-om na klizavim površinama dok je na suhoj podlozi kao asfalt produlji put kočenja.

Ključne riječi: ABS, konstrukcija, bicikl

SUMMARY

Title: Abs breaking system for bicycle

The division into segments such as planning and research up to the creation of ABS structure is shown. The entire production process, from the very planning of the circuit simulation, which was developed as new components were added. The assembly solution is essentially unchanged. The construction is closely related to it and could not be started until the first prototype was made. That prototype showed bugs and possible improvements that could be made to make the final solution work better. These improvements can be seen in the neatness and precision during soldering, changing the individual wires to those of the full section, and changing the sensors to simplify the program. The final step is the testing that shows the predicted results. The bike worked better with ABS on slippery surfaces, while on dry surfaces like asphalt, the braking distance was longer.

Keywords: ABS, construction, bicycle

ŽIVOTOPIS

Marin Valenčak je rođen 28.09.2001. godine u Osijeku. Živi u Osijeku i pohađa Fakultet elektrotehnike, računarstva i informacijskih tehnologija. Završio je Osnovnu Školu Retfala i nakon toga je upisala Elektrotehničku i prometnu školu Osijek, smjer Tehničar za mehatroniku. Tijekom fakultetskog obrazovanja, student je radio poslove u Bel-Tel d.o.o.-u i PROBE d.o.o. obavljajući poslove u struci.

PRILOZI

P.1. Nadređeni kod

```
#include <Arduino.h>
#include <U8g2lib.h>
#include <stdlib.h>
#include "A4988.h"
#include <SoftwareSerial.h>
#include "TimerInterrupt_Generic.h"
#include "images.h"

// definiranje pinova kao imena zbog lakšeg korištenja dalje u kodu

#define LED             PA1     // zadnje svjetlo
#define Buzzer          PB4     // pištalica
#define r               0.315   // polumjer
#define RSW_drive       PB15    // senzor na zadnjem kotaču
#define RSW_steering    PB14    // senzor na prednjem kotaču

// ova četiri pina koriste se za nadređenu-podređenu komunikaciju
// imaju odgovarajuće pinove na podređenoj strani
#define send_activateABS    PA6 // send_activateABS --- receive_activateABS
#define send_stopABS        PA7 // send_stopABS --- receive_stopABS
#define send_activateServo  PB0 // send_activateServo --- receive_activateServo
#define receive_ABSDone     PB1 // receive_ABSDone --- send_ABSDone

#define Pushbutton      PB9 // tipkalo

#define TIMER_INTERVAL_MS               1
#define DEBOUNCING_INTERVAL_MS          100
#define DEBOUNCING_INTERVAL_MS_STEPPER  2000

#define freq 5000


// Inicijalizacija STM32 TIM1
STM32Timer ITimer(TIM1);
//Inicijalizacija ekrana
```

```
U8G2_SSD1306_128X64_NONAME_F_HW_I2C OLED_display(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);

SoftwareSerial BluetoothSerial(PA10, PA9); //RX and TX pinovi za Bluetooth Serial


volatile long rotationTime_drive = 0;
double Speed_drive_wheel      = 0.00;
double avgSpeed_driveWheel    = 0.00;

volatile long rotationTime_steering = 0;
double Speed_steering_wheel      = 0.00;
double avgSpeed_steeringWheel    = 0.00;

float avgSpeed = 0.0;

volatile int debounceCounter_drive, debounceCounter_steering,
debounceCounter_pushbutton, pushbuttonCounter, pushbuttonWait, servoCounter = 0,
ABS_currentState;
volatile long ABS_blinkCounter = 0, debounceCounter_stepper, ABS_active = 0,
printABStime;
volatile bool state, state_test, flag_driveWheel,flag_steeringWheel,
flag_checkABS, flag_activateABS, flag_blinkABS;
volatile bool flag_pushbutton, flag_activateLED, flag_activateBuzzer, ABS_state,
servoState, flag_countABS, flag_countABSdone, flag_printABStime;
long deltaRotationTime = 0, ABS_counterStop = 0, ABS_counterStart = 0;


void TimerHandler()
{

    //Provjera je li senzor na zadnjem kotaču aktivan, debounceCounter_drive
koristi se kako ne bi dobili zbunjujući signal
    if ( !digitalRead(RSW_drive) && (debounceCounter_drive >=
DEBOUNCING_INTERVAL_MS / TIMER_INTERVAL_MS ) )
    {
        flag_driveWheel = true;
        flag_checkABS = true;
```

```
    }
    else
    {
        debounceCounter_drive++;
    }
    rotationTime_drive++; // Povećaj vrijeme rotacije zadnjeg kotača
    // Provjera je li senzor na prednjem kotaču aktivan, debounceCounter_drive
koristi se kako ne bi dobili zbunjujući signal
    if ( digitalRead(RSW_steering) && (debounceCounter_steering >=
DEBOUNCING_INTERVAL_MS / TIMER_INTERVAL_MS ) )
    {
        flag_steeringWheel = true;
        flag_checkABS = true;
    }
    else
    {
        debounceCounter_steering++;
    }
    rotationTime_steering++; // // Povećaj vrijeme rotacije prednjeg kotača
    // Provjera je li pritisnuto tipkalo
    if (!digitalRead(Pushbutton) && (debounceCounter_pushbutton >= 200 /
TIMER_INTERVAL_MS ) )
    {
        flag_pushbutton = true;

    }
    else
    {
        debounceCounter_pushbutton++;
    }
    /* Između svakog pritiska tipkala postoji vrijeme čekanja
    kako bi program pravilno odlučio koju naredbu izvršiti */
    if (pushbuttonCounter > 0 && pushbuttonCounter < 3)
    {
        pushbuttonWait++;
    }
    /* Koristi se "lažni" analogni jer na strani roba koriste se
    interrupt pinovi koji reagiraju na rastući brid signala,
```

```
    pa se ovime nakon određenog vremena pripremi za sljedeću iteraciju
    ovo vrijedi za servoState i ABS_currentState varijable*/
    if(servoState)
    {
        servoCounter++;
    }
    if (ABS_currentState == 1)
    {
        ABS_counterStop++;
    }
    else if (ABS_currentState == 2)
    {
        ABS_counterStart++;
    }
    // ABS_active služi za brojanje vremena od kad se motor uključio do trenutka
kada se isključi.
     if(flag_countABS)
    {
        ABS_active++;
    }
    // flag_countABSdone postane logička jedinica kada ima rastući brid na
interrupt pinu
    /* Ova funkcija sprema ABS_active u varijablu za printanje,
    kako bi se ona pripremila za sljedeću iteraciju. */
    if (flag_countABSdone)
    {
        flag_countABSdone = false;
        flag_countABS = false;
        printABStime = ABS_active;
        ABS_active = 0;
        flag_printABStime = true;
    }
    // ABS_blinkCounter služi za treperenje ikonica na ekranu i LED svjetla
    ABS_blinkCounter++;
}

void ABSdoneHandler() // interrput funkcija koja dobije signal kada se motor
ugasio
```

```
{
    flag_countABSdone = true;
}
void driveWheelHandler() // funkcija za računanje brzine zadnjeg kotača
{
    Speed_drive_wheel = (double) ( (2.0 * r * 3.14159*1000) / (
    rotationTime_drive*TIMER_INTERVAL_MS ) );
    avgSpeed_driveWheel = ( 2.0 * avgSpeed_driveWheel + Speed_drive_wheel) / 3;

    rotationTime_drive = 0;
    debounceCounter_drive = 0;
}

void steeringWheelHandler() // funkcija za računanje brzine prednjeg kotača
{
    Speed_steering_wheel = (double) ( (2.0 * r * 3.14159*1000) / (
    rotationTime_steering*TIMER_INTERVAL_MS ) );
    avgSpeed_steeringWheel = ( 2.0 * avgSpeed_steeringWheel +
    Speed_steering_wheel) / 3;

    rotationTime_steering = 0;
    debounceCounter_steering = 0;
}
void stopped() // funkcija koja resetira sve varijable kada je bicikli
zaustavljen
{
    avgSpeed_driveWheel = 0;
    rotationTime_drive = 0;
    avgSpeed_steeringWheel = 0;
    rotationTime_steering = 0;
}
// funkcija za iscrtavanje varijabli na ekran
// prima dvije varijable tipa float koje se pretvaraju u string
// onda se pomoću printf naredbe ispisuju na ekran
// setCursor služi za mijenjanje pozicije na koju ispisujemo
void displayHandler(float x, float y)
{
    avgSpeed = (avgSpeed + x + y)/3;
```

```
    char avgBikeSpeed[7];
    char unit[5] = "km/h";
    char speedWheelD[7];
    char speedWheelS[7];
    dtostrf(x, 7, 2, speedWheelD);
    dtostrf(y, 7, 2, speedWheelS);
    dtostrf(avgSpeed, 7, 2, avgBikeSpeed);
    OLED_display.setCursor(0,12);
    OLED_display.printf(speedWheelD);
    OLED_display.setCursor(0,24);
    OLED_display.printf(speedWheelS);
    OLED_display.setCursor(0,36);
    OLED_display.printf(avgBikeSpeed);
    OLED_display.setCursor(20,48);
    OLED_display.printf(unit);
    OLED_display.sendBuffer();
}

/* funkcija za ispisivanje ikonica na ekran
kada je kotač na biciklu blokiran */
void displayABSHandler() {
    if (!flag_blinkABS)
    {
        OLED_display.drawXBMP(128-32, 0, 32, 32, ABS_32x32); // draw the ABS logo
        OLED_display.drawXBMP(128-32*2-4, 0, 32, 32, black_bg); // draw the ABS
logo
        OLED_display.sendBuffer(); // transfer internal memory to the display
    }
    else
    {
        OLED_display.drawXBMP(128-32*2-4, 0, 32, 32, Auto_32x32);
        OLED_display.drawXBMP(128-32, 0, 32, 32, black_bg);
        OLED_display.sendBuffer(); // transfer internal memory to the display
    }


}
```

```
// funkcija za crtanje ikonice kada je ABS sustav ugašen
void displayABSoffHandler()
{
    OLED_display.drawXBMP(128-32, 0, 32, 32, ABS_off); // draw the ABS off logo
    OLED_display.sendBuffer(); // transfer internal memory to the display

}

// funkcija za kontrolu načina svijetljenja crvenih LED-ica
// ima dva dijela, prvi kada ABS ne radi onda LED svijetli konstantno
// u drugom dijelu ABS se uključio i LED treperi
// koristi se BJT za upravljanje LED i LED pin je spojen na bazu
// to znači da što je veća struja na bazi to će LED jače svijetliti
// mi to kontroliramo pomoću analogWrite funkcije
void blinkLED()
{
    // flag_activateLED koristi se da bi program "znao"
    // kada LED treba treperiti, kada konstantno svijetliti
    if(flag_activateLED)
    {
        flag_activateLED = false;
        if(!flag_blinkABS)
        {
            analogWrite(LED, 0);
        }
        else if(flag_blinkABS)
        {
            analogWrite(LED, 255);
        }
    }
    else
    {
        analogWrite(LED, 50);
    }

}

// funkcija za kontroliranje piezoelektričnog elementa
```

```
// koristi se tone() funkcija iz Arduino.h biblioteke
// ona prima pin, frekvenciju i vrijeme koliko će dugo pištati
// pištalica ima svoju zastavicu da se može odmah prekinuti
// kada kotač više nije blokiran
void Buzzer_Buzz()
{
    if(flag_activateBuzzer)
    {
        flag_activateBuzzer = false;
        tone(Buzzer, freq ,1000/4);
    }
    else
    {
        noTone(Buzzer);
    }

}

// funkcija za povećavanje vrijednosti pushbuttonCounter
// jer tipkalo nam ima više od jedne funkcije (pali/gasi servo motor i pali/gasi
ABS sustav)
void pushbuttonHandler()
{
    pushbuttonCounter++;
    pushbuttonWait = 0;
    debounceCounter_pushbutton = 0;
}
// funkcija koja odlučuje što će program napraviti
// ovisno o tome koliko puta smo stisnuli tipkalo
// korištena je switch case funkcija pošto s njome
// ljepše izgleda nego sa if funkcijama, no radilo bi i sa njima
void pushbuttonDecisionHandler()
{
    switch(pushbuttonCounter)
        {
            case 1:
            pushbuttonCounter = 0;
            if(avgSpeed_driveWheel == 0 && avgSpeed_steeringWheel == 0)
```

```
            {
                servoState = true;
                digitalWrite(send_activateServo, HIGH);
                // kod za paljenje servo motora
                // servo motor može se upaliti jedino kada bicikl stoji,
                // zato je unutar if funkcije
            }
                break;
            case 2:
            pushbuttonCounter = 0;
                ABS_state = !ABS_state;
                // kod za paljenje i gašenje ABS-a
                break;
            default:
                pushbuttonCounter = 0;
                // dio koda ako stisnemo previše puta tipkalo, da ga vrati na 0
                break;
        }
}
void setup()
{
    // Pokreće BluetoothSerial
    BluetoothSerial.begin(9600);
    // Podešavanje senzora i tipkala
    // Senzor na zadnjeg kotaču šalje logičku nulu kada ima signala, isto kao i
tipkalo
    // Senzor na prednjem kotaču šalje logičku jedinicu kada ima signala
    pinMode(RSW_drive, INPUT_PULLUP);
    pinMode(RSW_steering, INPUT);
    pinMode(Pushbutton, INPUT_PULLUP);

    pinMode(LED, OUTPUT);
    pinMode(Buzzer, OUTPUT);
    pinMode(send_activateABS, OUTPUT);
    pinMode(send_stopABS, OUTPUT);
    pinMode(send_activateServo, OUTPUT);
    // Konfiguracija receive_ABSDone kao interrupt pin-a
```

```
    // To se radi kako bi imali što točniju informaciju koliko dugo je motor
radio.
    pinMode(receive_ABSDone, INPUT_PULLDOWN);
    attachInterrupt(digitalPinToInterrupt(receive_ABSDone), ABSdoneHandler,
RISING);

    // Pokretanje ekrana, te postavljanje fonta
    // setPowerSave(0) gasi štednju ekrana što znači da nam je ekran cijelo
vrijeme upaljen
    OLED_display.begin();
    OLED_display.enableUTF8Print();
    OLED_display.setPowerSave(0);
    OLED_display.setFont(u8g2_font_7x14B_tf);
    // clear() očisti ekran
    OLED_display.clear();
    delay(5000);

    // Postavljanje bitnih varijabli na 0
    debounceCounter_drive = 0;
    debounceCounter_steering = 0;
    debounceCounter_stepper = 0;
    debounceCounter_pushbutton = 0;
    pushbuttonCounter = 0;
    ABS_currentState = 0;
    flag_blinkABS = false;
    ABS_state = false;
    state = false;
    // Vezanje timer interrupta sa ISR funkcijom
    // attachInterruptInterval prima vrijeme u mikrosekundama
    ITimer.attachInterruptInterval(TIMER_INTERVAL_MS * 1000, TimerHandler);
}

void loop()
{
    // inline if funkcija koja služi za treperenje LED i ikonica na ekranu
      ABS_blinkCounter >= (500/TIMER_INTERVAL_MS) ? flag_blinkABS =
      !flag_blinkABS, ABS_blinkCounter = 0 : flag_blinkABS = flag_blinkABS;
    // u sljedeće dvije if funkcije koriste se zastave
```

```
// koje kada su 1, pozivaju funkcije za računanje brzine
if(flag_driveWheel){
    flag_driveWheel = false;
    driveWheelHandler();
}
if(flag_steeringWheel){

    flag_steeringWheel = false;
    steeringWheelHandler();
}

// ovdje su dvije if funkcije kako bi bilo lakše za čitati kod
// ako nema signala na senzorima i prošlo je dovoljno vremena
// program to shvaća kao da bicikli stoji pa poziva stopped() funkciju
if(!flag_driveWheel && !flag_steeringWheel)
{
    if (rotationTime_drive >= 5000 && rotationTime_steering >= 5000)
    {
        stopped();
    }

}
// ABS_state služi za paljenje i gašenje ABS-a, po zadanome on je aktivan
// kada se ugasi ABS onda se iscrtava ikonica za ugašeni ABS
if(!ABS_state)
{
    // flag_checkABS bude 1 svaki puta kada ima signala na ijednom od senzora
    // on provjerava je li zadnji kotač blokiran
    if(flag_checkABS)
    {
        flag_checkABS = false;

        // infline if služi kao absolute funkcija u Math.h biblioteci
        deltaRotationTime = (rotationTime_drive - rotationTime_steering) >=
        0 ? (rotationTime_drive - rotationTime_steering) :
        (rotationTime_drive - rotationTime_steering)*(-1);
        // deltaRotationTime je razlika u periodama prednjeg i zadnjeg kotača
```

```
            // ako je razlika prevelika, kotač je blokirao i program
flag_activateABS postavlja na log. 1
            if(deltaRotationTime <= 175)
            {
                flag_activateABS = false;
                ABS_currentState = 1;
                flag_activateBuzzer = false;
                Buzzer_Buzz();
                // digitalWrite se koristi jer je koračni motor na drugoj STM32
pločici
                // send_stopABS pin je spojen na odgovarajući pin na podređenoj
pločici
                // i on podređenoj pločici govori da zaustavlja koračni motor
istog trena
                digitalWrite(send_stopABS, HIGH);
            }
            else
            {

                flag_activateABS = true;

            }
        }
        // ako je flag_activateABS == 1, kotač je blokiran i treba upaliti
koračni motor
        // displayHandler() se ovdje nalazi kako bi tijekom
        // iscrtavanja ikonica na ekran brzina  još uvijek se iscrtavala

        if(flag_activateABS)
        {
            flag_activateABS = false;
            displayHandler(3.6*avgSpeed_driveWheel, 3.6*avgSpeed_steeringWheel);
            displayABSHandler();
            flag_activateLED = true;
            blinkLED();
            flag_activateBuzzer = true;
            Buzzer_Buzz();
            ABS_currentState = 2;
```

```
            // send_activateABS pin je spojen na odgovarajući pin na podređenoj
ploči
            // on podređenoj ploči kaže da upali koračni motor
            digitalWrite(send_activateABS, HIGH);
            flag_countABS = true;
        }
    }
    else if(ABS_state)
    {
        displayABSoffHandler();
    }
    // *1*, *2* i *3* if funkcije služe za postavljanje odgovarajućih pinova na
logičku nulu
    // jer na podređenoj ploči ti odgovarajući pinovi su interrupt-i
    // pa se ovime pripremamo za sljedeću iteraciju
    if(ABS_currentState == 1 && ABS_counterStop >= 10) // *1*
    {
        ABS_currentState = 0;
        ABS_counterStop = 0;
        BluetoothSerial.println("Stop stepper");
        digitalWrite(send_stopABS, LOW);
    }
    if(ABS_currentState == 2 && ABS_counterStart >= 10) // *2*
    {
        ABS_currentState = 0;
        ABS_counterStart = 0;
        BluetoothSerial.println("Send stepper");
        digitalWrite(send_activateABS, LOW);
    }
    if(servoState && servoCounter >= 1) // *3*
    {
        servoState = false;
        servoCounter = 0;
        digitalWrite(send_activateServo, LOW);
    }

    // poziva funkciju pushbuttonHandler()
    if (flag_pushbutton)
```

```
    {
        flag_pushbutton = false;
        pushbuttonHandler();
    }

    // nakon određenog vremena bez signala na tipkalu, program postavlja
pushbuttonCounter na 0
    if(debounceCounter_pushbutton >= 5000)
    {
        pushbuttonCounter = 0;
    }
    // nakon svakog stiska na tipkalo, resetira se pushbuttonWait na 0,
    // kako bi korisnik imao vremena stisnuti tipkalo 2 ili više puta
    // ta varijabla je potrebna jer bez nje program bi uvijek izvršavao
    // slučaj 1, a to je aktivacija servo motora
    if(pushbuttonWait >= 5*100/TIMER_INTERVAL_MS)
    {
        pushbuttonWait = 0;
        pushbuttonDecisionHandler();
    }
    // blinkLED() se poziva kako bi LED svijetlio cijelo vrijeme
    blinkLED();
    // displayHandler() se poziva kako bi se ispisivale brzine oba kotača
    displayHandler(3.6*avgSpeed_driveWheel, 3.6*avgSpeed_steeringWheel);
    // flag_printABStime služi da se vrijeme aktivnosti motora ispiše samo kada
je potrebno,
    // a ne cijelo vrijeme jer bi to usporilo program još više.
    if (flag_printABStime)
    {
        flag_printABStime = false;
        BluetoothSerial.print("Stepper counter: ");
        BluetoothSerial.println(printABStime);
    }



}
```

P.2. Podređeni kod

```
#include <Arduino.h>
#include "ServoEasing.hpp"
#include "Servo.h"
#include <stdlib.h>
#include "A4988.h"


// definiranje pinova kao imena zbog lakšeg korištenja dalje u kodu

#define MOTOR_STEPS     200
#define stepper_RPM     100
#define MICROSTEPS      1

#define DIR             PB0
#define STEP            PB1
#define EN              PB12

#define MS1             PA5
#define MS2             PA4
#define MS3             PA3

#define Servo_pwm       PB8

// ova četiri pina koriste se za nadređenu-podređenu komunikaciju
// imaju odgovarajuće pinove na nadređenoj strani
#define receive_activateABS     PB13 // receive_activateABS --- send_activateABS
#define receive_stopABS         PB14 // receive_stopABS --- send_stopABS
#define receive_activateServo   PB15 // receive_activateServo ---
send_activateServo
#define send_ABSDone            PA8 // send_ABSDone --- receive_ABSDone

//Inicijalizacija upravljača koračnog motora
A4988 stepper(MOTOR_STEPS, DIR, STEP, EN, MS1, MS2, MS3);
//Inicijalizacija servo motora
ServoEasing myServo;

volatile bool flag_startABS, flag_stopABS, flag_servoHandler;
```

```
unsigned long time, debounceServo = 0;

// interrupt funkcija koja postavlja zastavu za pokretanje servo motora
void ServoHandler()
{
    flag_servoHandler = true;
}

// interrupt funkcija koja postavlja zastavu za pokretanje koračnog motora
void startABS()
{
    flag_startABS = true;
}

// interrupt funkcija koja postavlja zastavu za zaustavljanje koračnog motora
void stopABS()
{
    flag_stopABS = true;
    flag_startABS = false;
}

// funkcija koja pali i pokreće koračni motor
// koračni motor se okreće 4 kruga i onda staje i gasi se
// startMove() se koristi jer ne blokira ostali dio programa
// dok koračni motor okrene 4 kruga naspram move() funkcije
void startStepper()
{
    stepper.enable();
    stepper.startMove(4 * MOTOR_STEPS * MICROSTEPS);
}

// funkcija koja zaustavlja koračni motor
 void stopStepper()
 {

    stepper.stop();
    stepper.disable();
    // šalje se signal da se koračni motor ugasio na nadređenu stranu
```

```
    // to služi za brojanje vremena aktivnosti koračnog motora
    digitalWrite(send_ABSDone,HIGH);
    delay(1);
    digitalWrite(send_ABSDone,LOW);

 }
// funkcija za okretanje servo motora u jednu stranu pa u drugu stranu
// radi tako da pomoćui read() funkcije očitava se pozicija servo motora
// u stupnjevima, i ovisno u kojoj se poziciji nalazi,
// okretati će se natrag odnosno naprijed
// kao i sa koračnim motorom koristi se startEaseTo() jer ne blokira izvršavanje
programa
void servoLogic()
{
    if(myServo.read() >= 175)
    {
        myServo.startEaseTo(0);
    }
    if(myServo.read() <= 5)
    {
        myServo.startEaseTo(180);
    }

}
void setup()
{
    // postavljanje 3 ulazna pina kao interrupte i 1 kao izlaz
    pinMode(receive_activateABS, INPUT);
    attachInterrupt(digitalPinToInterrupt(receive_activateABS), startABS,
RISING);
    pinMode(receive_stopABS, INPUT_PULLDOWN);
    attachInterrupt(digitalPinToInterrupt(receive_stopABS), stopABS, RISING);
    pinMode(receive_activateServo, INPUT_PULLDOWN);
    attachInterrupt(digitalPinToInterrupt(receive_activateServo), ServoHandler,
RISING);
    pinMode(send_ABSDone, OUTPUT);

    // pokretanje servo motora i postavljanje logičkog stanja na EN pinu
```

```
    // stepper se ne pali
    // begin služi kako bi biblioteka znala kojom brzinom da se okreće i ako
postoje mikrokoraci
    stepper.begin(stepper_RPM, MICROSTEPS);
    stepper.setEnableActiveState(LOW);

    // postavljanje zastavica na početno stanje
    flag_startABS = false;
    flag_stopABS = false;

    // povezivanje servo motora sa njegovim PWM pinom
    myServo.attach(Servo_pwm);
    // postavljanje brzine zakreta servo motora
    myServo.setSpeed(40);
    // zakret servo motora na početnu poziciju
    myServo.easeTo(0);
    delay(5000);
}

void loop()
{
    // ako je zastavica flag_startABS u logičkoj jedinici i prošlo je dovoljno
vremena,
    // pozovi funkciju za pokretanje koračnog motora
    if(flag_startABS && (millis() - time) >= 200)
    {
        flag_startABS = false;

        time = millis();
        startStepper();
    }
    // nextAction() funkcija mora biti pozvana u kodu kako bi motor ispravno
funkcionirao
    // vezana je sa startMove() funkcijom
    if (stepper.nextAction() > 100)
    {

    }
```

```
    // ako je zastavica flag_stopABS u logičkoj jedinici ili prošlo je dovoljno
vremena,
    // pozovi funkciju za zaustavljanje koračnog motora
     if(flag_stopABS || (millis() - time) >= 3000)
    {
        flag_stopABS = false;
        time = millis();
        stopStepper();
    }

    // ako je zastavica flag_servoHandler u logičkoj jedinici i prošlo je
dovoljno vremena,
    // pozovi funkciju za zakret servo motora
    if (flag_servoHandler && (millis() - debounceServo) >= 500)
    {
        flag_servoHandler = false;
        debounceServo = millis();
        servoLogic();
    }




}
```