TEHNIČKA ŠKOLA RUĐERA BOŠKOVIĆA

GETALDIĆEVA 4, ZAGREB

ZAVRŠNI STRUČNI RAD:

Skateboard upravljan mikroupravljačem

MENTOR: dipl. ing. el. Zoran Dumančić

UČENIK: Gvido Maskalan

RAZRED: 4.H

Zagreb, travanj 2022.

Sadržaj

1. Uvod

Električna vozila svakim danom postaju sve učestalija u svakodnevnom životu. Potražnja za njima raste iz dana u dan i pitanje je samo kada će postati najpopularnijom vrstom prijevoznih sredstava. Glavni razlog ovoga je činjenica da je zagađivanje okoliša velik problem današnjice i potrebno ga je riješiti što prije, a jedan od najlakših načina za pridonijeti čišćoj Zemlji je korištenjem električnih prijevoznih sredstava. Još neki od razloga zašto su električna vozila sve popularnija su manje cijene transporta (jeftinije od vožnje automobilom, tramvajem, itd.) te izbjegavanje gužvi.

Iz istih ovih razloga sam odlučio napraviti električni skateboard te ga koristeći znanje iz područja mikroupravljača, ugradbenih računalnih sustava te elektrotehnike unaprijediti dodatnom funkcionalnosti kakvu nema mnogo takvih vozila. Cilj je bio napraviti električni skateboard koji bi se mogao povezati s mobitelom te preko mobilne aplikacije prikazivati zanimljive i korisne informacije poput brzine, prijeđenog puta, postotka baterije, itd. Aplikacijom bi se također moglo i upravljati skateboardom u slučaju da zaboravimo daljinski upravljač ili nam se isprazne baterije.

2. Teorijski dio

2.1. Električni skateboard

Električni skateboardi postoje već dugo. Prvi je bio proizveden 1997. godine, a napravio ga je Louise Finkle koji ga je 1999. godine i patentirao, no to je bilo pre rano za električna vozila s obzirom na to da su baterije bile loše [1]. Ideja je zaživjela tek pojavom boljih baterija, a prvi veliki proizvođač koji je sve započeo bio je Boosted Board, koji je kampanjom na Kickstarteru uspio prikupiti preko 500,000$. Ubrzo nakon pojavilo se mnogo proizvođača električnih skateboarda poput MeepoBoard-a, Maxfind-a, BajaBoard-a itd.

Slika 2.1.1. Boosted Board 2nd GEN DUAL+

Preuzeto s: https://www.voltaboards.nl/en/boosted-board-2nd-gen-dual.html

Električni skateboardi sastoje je od 3 glavne komponente: baterije, ESC-a (Electronic Speed Controller) i motora. Ove komponente detaljnije su objašnjenje u nastavku.

2.2. Baterija

Baterije su ključni dio svih električnih vozila, uključujući i skateboarde. One su spremnici energije koji se koriste za pokretanje svakog električnog vozila. U električnim skateboardima se koriste litij-ionske ili litij-polimerske baterije zbog mogućnosti da drže veliku količinu energije u jako malom volumenu (najefikasnije su).

Slika 2.2.1. 12V Lithium-Ion baterija

Preuzeto s: https://www.indiamart.com/proddetail/lithium-ion-battery-for-car-23422456562.html

Slika 2.2.2. Revox Pro 3S 5500mAh 30C Lithium Polymer baterija

Preuzeto s: http://www.aeroflyhobbies.com/lipo-battery-packs/181-revox-pro-3s-5500mah-30c-lithium-polymer-battery.html

Prilikom odabira baterije moramo obratiti pažnju ne samo na podatak o snazi baterije nego i ostalim specifikacijama poput napona, kapaciteta, life cycle-a, itd. Specifikacije baterija:

– napon – određuje maksimalnu brzinu skateboarda

– kapacitet – određuje domet

– life cycle – koliko pražnjenja baterija može proći prije nego počne gubiti efikasnist

– discharge current – koja je maksimalna struja koju možemo vući iz baterije (određuje akceleraciju)

2.3. ESC

ESC (Electronic Speed Controller) je uređaj koji se koristi za kontroliranje rotacijske brzine električnih motora. Oni se koriste u praktički svim uređajima koji imaju motore. Postoje dvije vrste ESC-a: brushed i brushless ESC-evi.

Brushed ESC-evi se koriste za motore s četkicama (brushed DC motori). Oni kao ulaz imaju bateriju te upravljač koji određuje brzinu motora, a kao izlaz imaju 2 žice, pozitivnu i negativnu, koje se spajaju na motor. Oni brzinu reguliraju promjenom vremenskog intervala puštanja i prekidanja toka struje kroz motor. [4]

Slika 2.3.1 Botbitz – 80a Brushed ESC

Preuzeto s: https://ranglebox.com/shop/product/botbitz-80a-brushed-esc/

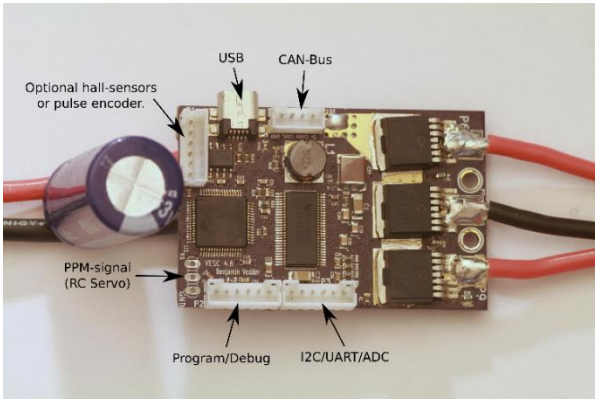
Brushless ESC-evi koriste se za BLDC motore (Brushless Direct Current motori). Oni kao ulaz imaju bateriju i upravljač isto kao i brushed ESC-evi, no za izlaz umjesto dvije žice imaju ih tri. BLDC motori rade slično kao i trofazni AC motor tako da im trebaju 3 žice (faze) za rad. Od tri žice u svakom trenutku je jedna spojena na napon, jedna na uzemljenje i jedna je odpojena. Na taj način se motor okreće, a brzina se regulira frekvencijom izmjene stanja na tim žicama. [4]

Slika 2.3.2. E-flite 45-Amp Linear Mode BEC Brushless ESC

Preuzeto s: https://www.amazon.com/EFL-45-Amp-Brushless-Esc/dp/B071Y3FRG5

2.3.1. VESC

VESC (Vedder's Electronic Speed Controller) je vrsta open-source brushless ESC-a. Prednost VESC-a je u tome što se mogu mijenjati mnoge postavke koje su na običnim ESC-evima nepromjenjive poput ulaznog napona, izlaznog napona, maksimalnog broja okretaja, maksimalne struje, itd. [6]



Slika 2.3.1.1. Originalni VESC

Preuzeto s: http://vedder.se/2015/01/vesc-open-source-esc/

VESC je kreirao Benjamin Vedder 2016. godine s ciljem kreiranja najboljeg ESC-a na tržištu. Glavni razlog uspjeha VESC-a je bio to što je u potpunosti programabilan pa se može koristiti za mnogo različitih namjena. Zbog toga je i postao toliko popularan u svijetu električnih skateboarda. Svaki skateboard je drugačiji, imaju drugačiju bateriju, različite motore i različite namjene. Kako bi radio kao što je namijenjen treba imati odgovarajući ESC, a skoro je nemoguće kupiti ESC s postavkama točno onakvima kakve nam trebaju. Tu nastupa VESC koji može poslužiti u širokom rasponu te se konfigurirati tako da maksimalno produlji životni vijek svih komponenti u sustavu.

S obzirom da je VESC open-source postoji puno proizvođača koji prodaju VESC-ove pod drugim imenima [6]. Jedan od njih je Flipsky čija je varijanta VESC-a FSESC. U ovom radu korišten je FSESC 4.12 koji radi na naponima od 8V do 60V (3S – 13S baterije) te ima mogućnost vođenja struje od 50A konstantno, te maksimalnu struju od 240A.

Slika 2.3.1.2. FSESC 4.12 50A baziran na VESC4.12

Preuzeto s: https://flipsky.net/products/torque-esc-vesc-%C2%AE-bldc-electronic-speed-controller

2.3.2. VESC Tool

VESC Tool je aplikacija koja omogućava potpunu konfiguraciju VESC-ova. [7] To je unaprijeđena verzija BLDC Tool-a kojeg je razvio Benjamin Vedder zajedno sa VESC-om. Cilj je bio kreirati aplikaciju koja bi bila dovoljna za u potpunosti iskonfigurirati VESC na jednostavan način, kako bi to mogao učiniti bilo tko.

Slika 2.3.2.1. VESC Tool

Preuzeto s: https://vesc-project.com/vesc_tool

VESC Tool je jako intuitivan alat kojim se može na lak način iskonfigurirati VESC korištenjem Wizarda (Čarobnjaka) za motore, upravljanje, itd. Naravno napredniji korisnici sve mogu ručno unositi onako kako žele, no i manje iskusni lako će se snaći u aplikaciji

Uz konfiguraciju VESC-a također nam omogućava i real-time praćenje svih parametara koji se mijenjaju kroz rad VESC-a. Podatci koje možemo pratiti su: struja koju VESC vuče iz baterije, struja koja se šalje motoru, duty cycle, temperatura motora, temperatura MOSFET-a, broj okretaja motora, pozicija rotora, napon baterije te količina energije potrošena i napunjena u Ah i Wh.

Još jedna zanimljiva opcija koju VESC Tool nudi je mobilna aplikacija koja preko Bluetooth-a omogućava skoro sve mogućnosti desktop verzije ali također omogućava nadgledanje podataka u stvarnom vremenu za vrijeme korištenja VESC-a u realnim uvjetima (npr. tijekom vožnje skateboarda). Aplikacija napravljena za ovaj završni rad za upravljanje skateboardom preko Bluetooth-a ima neke od funkcionalnosti koje nudi i mobilna aplikacija VESC Tool-a uz dodatak mogućnosti upravljanja skateboardom.

2.4. BLDC Motori

BLDC motori su električni motori bez četkica koji za rad koriste istosmjernu struju. To im omogućavaju ESC-evi koji se spajaju na izvor istosmjerne struje i mijenjaju smjer struje kroz zavojnice u motoru (slično kao što radi i AC izvor) kako bi se motor okretao. Mogu biti izvedeni kao inrunner ili outrunner motori. [8]

Prednosti BLDC motora u odnosu na DC motore s četkicama su veća efikasnost jer nema gubljenja energije na trenje između statora i rotora zbog struganja četkica (85% - 90% efikasnosti u odnosu na 75% - 80%), dulji vijek trajanja (četkice se troše i često ih treba mijenjati) i tiši rad. [10]

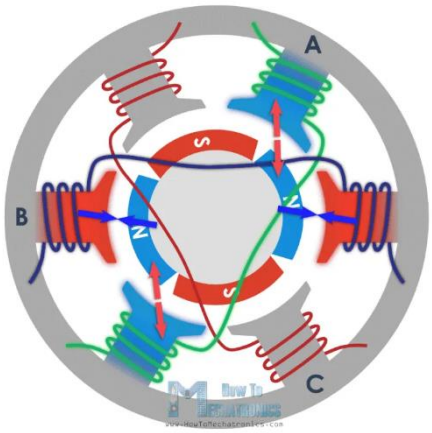
Svaki BLDC motor se sastoji od 2 glavna dijela: statora i rotora. Na rotoru se uvijek nalaze stalni magneti, a na statoru zavojnice. Princip rada je taj da puštanjem struje kroz zavojnicu kreiramo magnetsko polje koje može privlačiti ili odgurivati jedan od stalnih magneta na rotoru. Tako se paljenjem i gašenjem magnetskog polja jedne po jedne zavojnice može rotirati rotor. Kako bi povećali efikasnosti možemo mijenjanjem smjera struje kroz zavojnicu mijenjati da li polje privlači ili odgurava magnet (kada jedna zavojnica privlači magnet, druga ga odguruje). Također spajanjem dvije nasuprotne zavojnice možemo postići duplu snagu privlačenja ili odbijanja (dvije zavojnice privlače nasuprotne magnete u isto vrijeme). [9]

Slika 2.4.1. BLDC motor

Preuzeto s: https://www.tytorobotics.com/blogs/articles/how-brushless-motors-work

2.4.1. Način rada na primjeru motora s 2 pola

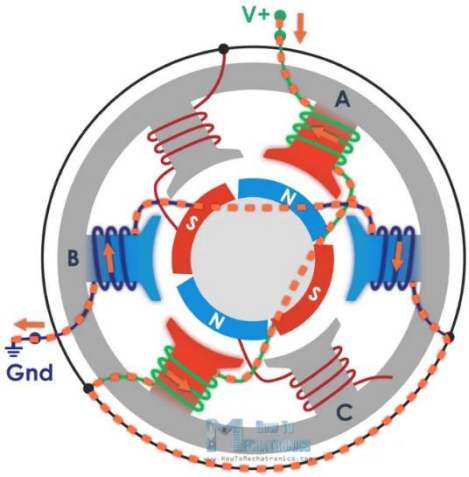
Za primjer možemo uzeti motor s 2 pola. U sredini, na rotoru, se nalaze četiri magneta, a na statoru šest zavojnica.



Slika 2.4.1.1. Princip rada BLDC motora

Preuzeto s: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

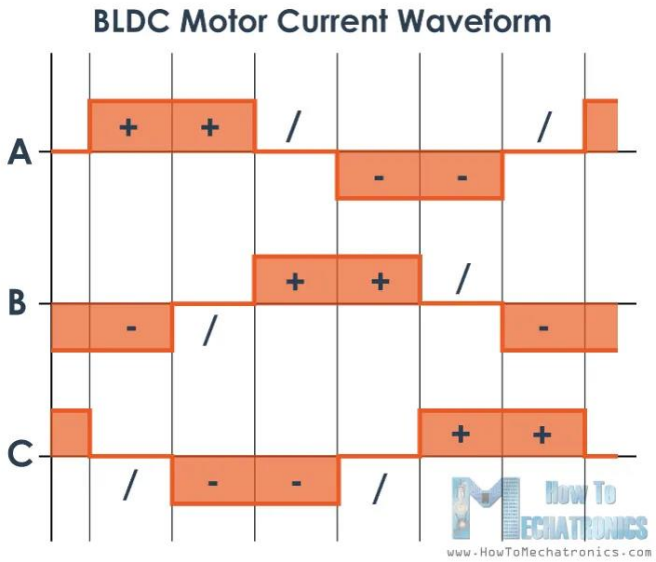
Kao što se vidi sa slike, zavojnice su spojene u grupama po dvije (nasuprotne) koje u isto vrijeme odrađuju istu funkciju (privlačenje ili odbijanje magneta). Tako u paru spojene su u Y konfiguraciju (jedna od dvije spojena je na zajedničku žicu, a druga na ESC). Takvim načinom spajanja možemo korištenjem jednog toka struje napajati 4 zavojnice. [9]



Slika 2.4.1.2. Tok struje u BLDC motoru

Preuzeto s: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

To možemo vidjeti i iz grafa koji prikazuje tok struje kroz žice spojene na ESC. U svakom trenutku je jedna žica spojene na napon s baterije, jedna na uzemljenje, i jedna nije spojena. Tako struja teče iz baterije kroz jedan par zavojnica u jednom smjeru induciranjem magnetskog polja jedne polarizacije, pa kroz zajedničku točku do drugog para zavojnica u suprotnom smjeru induciranjem magnetskog polja suprotne polarizacije u GND baterije. [9]

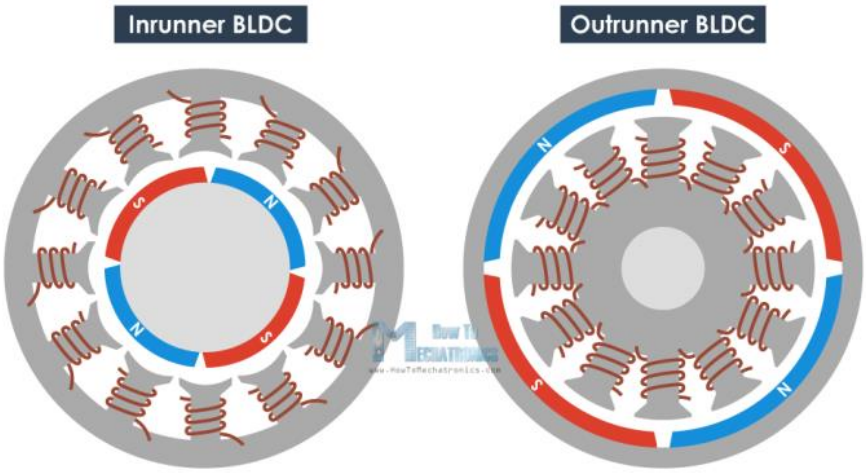


Slika 2.4.1.3. Graf struja na žicama motora

Preuzeto s: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/

2.4.2. Outrunner vs inrunner BLDC motori

Postoje dvije različite izvedbe BLDC motora: inrunner i outrunner motori. Inrunner motori su oni kojima se rotor nalazi u sredini dok su zavojnice na vanjskoj strani motora. Kod njih se okreće samo osovina. Outrunner motori su pak napravljeni tako da je stator u sredini motora, a rotor je cijelo kućište. O tome koji ćemo odabrati za neki projekt ovisi koja je namjena motora i kakve nam specifikacije trebaju (oba imaju određene prednosti).



Slika 2.4.2.1. Inrunner (lijevo) i outrunner (desno) motor

Preuzeto s: https://futurelab3d.com/how-brushless-motor-and-esc-work/

Najočiglednija razlika između inrunner i outrunner motora je u veličini. Inrunner motori su većinom manji u promjeru i duži od outrunner motora istih specifikacija.

Najvažnija razlika je u kV vrijednosti (rotacijska brzina po voltu). Inrunner motori većinom imaju veću kV vrijednost od outrunner motora zbog različitih veličina. Outrunner motori zbog većeg radijusa rotora imaju više magneta tako da ESC mora raditi puno brže da bi se motor okretao istom brzinom u usporedbi sa inrunner motorom. Također rotor outrunner motora mora prijeći puno veći put od rotora inrunner motora kako bi napravio jedan okretaj što utječe na kV vrijednost. Ovo je jedna od važnijih razlika zato što je kV vrijednost najvažnija specifikacija motora prilikom odabira motora jer određuje kojom će se maksimalnom brzinom moći okretati motor na određenom naponu. [11]

Još jedna važna razlika je u okretnom momentu koji je veći kod outrunner motora zbog većeg radiusa rotora što znači da je krak sile veći. Iz toga vidimo da je odnos kV vrijednosti i okretnog momenta BLDC motora obrnuto proporcionalan. [11]

Razlike se također mogu primijetiti i u efikasnosti motora, a do tih razlika dolazi zbog zagrijavanja. Što je temperatura motora veće efikasnost je manja tako da je po tome onaj motor koji ima bolje hlađenje efikasniji, a motori koji imaju bolje hlađenje su inrunner motori. To je zato što su zavojnice, koje su jedini dio motora koji se zagrijava, na vanjskoj strani motora tako

da sva toplina prelazi na kućište i brzo odlazi u zrak jer je površina velika, dok kod outrunner motora sva toplina ostaje zaglavljena unutar motora.

2.5. 2.4GHz upravljač

Koko bi VESC znao kojom brzinom treba okretati motor potrebno mu je slati nekakve ulazne podatke. Za to se većinom koriste 2.4GHz upravljači poput onih za RC autiće. Oni u daljinskom imaju potenciometar koji očitava koliko smo jako pritisnuli gas i to preko odašiljača šalju na prijemnik koji je spojen na VESC. Prijemnik na VESC šalje PWM (Pulse Width Modulation) signale iz kojih VESC iščitava kojom brzinom bi trebao okretati motor.

Slika 2.5.1. 2.4GHz RC upravljač

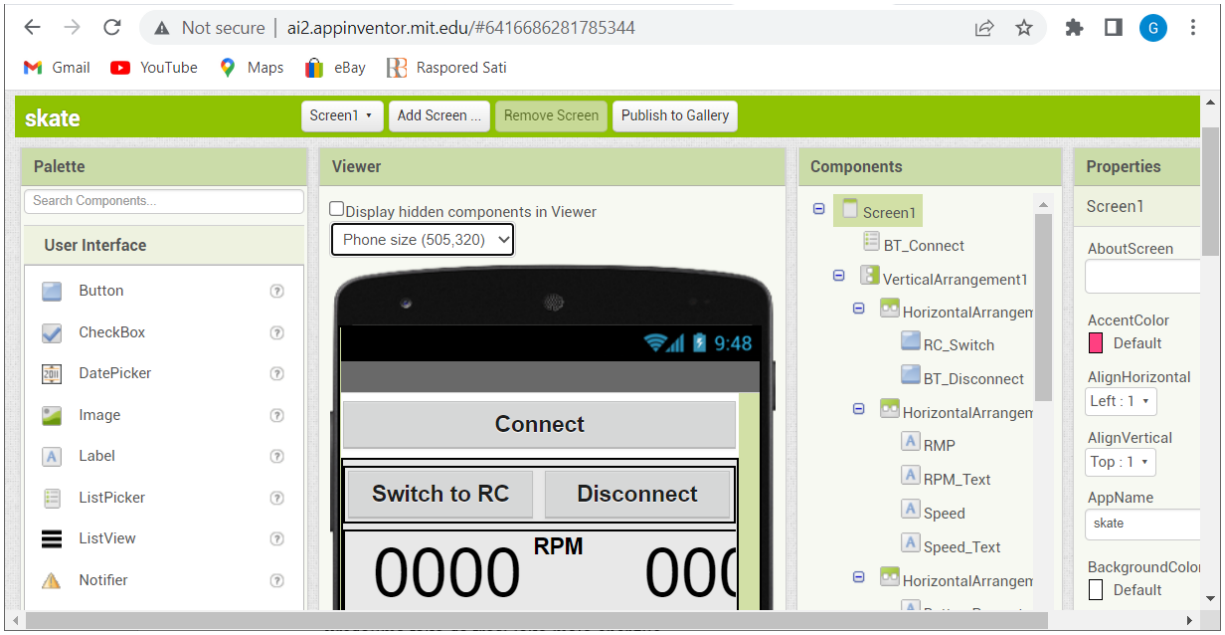
Preuzeto s:
https://hrsale.2021outletshops.ru/content?c=2.4%20ghz%20radio%20remote%20controller%20receiver%20transmitter%20for%20electric%20skateboard&id=7

2.6. MIT App Inventor

MIT App Inventor je program napravljen s ciljem da omogući svima, pa čak i djeci, da na jednostavan način mogu napraviti Android ili iOS aplikaciju. To je omogućeno korištenjem intuitivnog sučelja s drag-and-drop načinom rada i programiranjem blokovima koje je lako shvatljivo svim uzrastima.

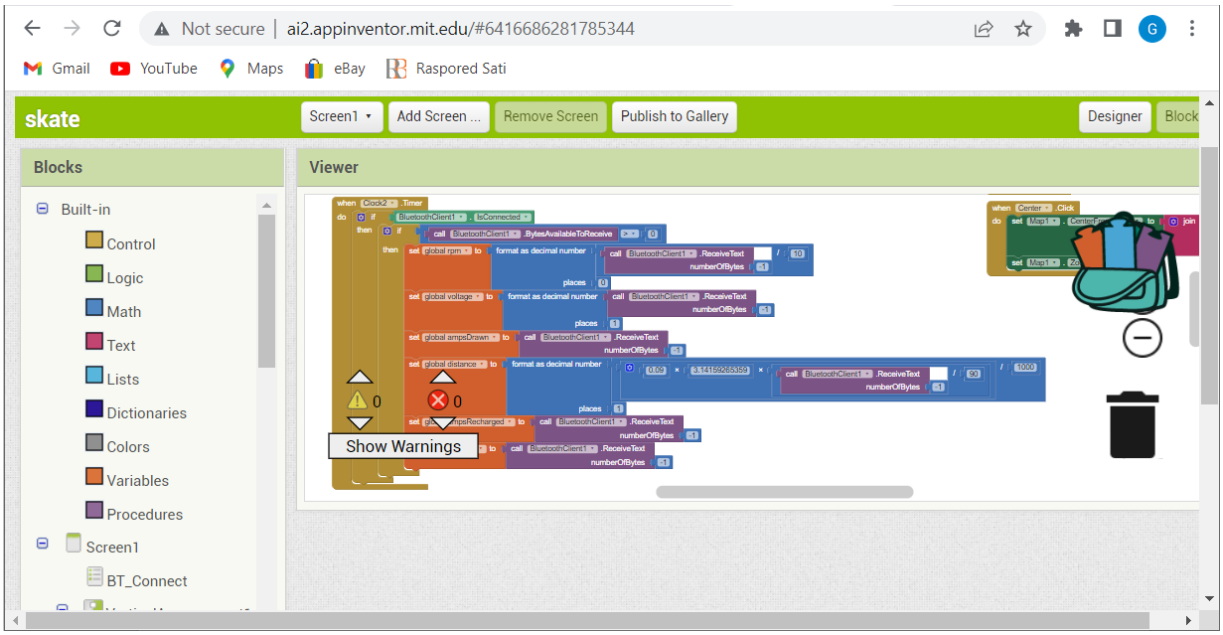
Kreirao ga je mali tim profesora i studenata s MIT CSAIL-a predvođen profesorom Hal Ablesonom. Oni također održavaju i web stranicu koja nam omogućava besplatno kreiranje aplikacija preko interneta, bez potrebe za instaliranjem ičega. [12]

Radno okruženje MIT App Inventora sastoji se od Designer moda i Blocks moda. Designer mod rada služi za izradu grafičkog sučelja aplikacije jednostavnim drag-and-drop načinom rada, dok u Blocks modu rada sve komponente korištene u grafičkom sučelju možemo programirati po želji.



Slika 2.6.1. Designer mode

Preuzeto s: http://ai2.appinventor.mit.edu/#6416686281785344



Slika 2.6.2. Blocks mode

Preuzeto s: http://ai2.appinventor.mit.edu/#6416686281785344

2.7. ESP32

ESP32 je mikroupravljač sa integriranim Wi-Fi-jem i Bluetooth-om namijenjen za korištenje u širokom području uporaba. Kreirala ga je kineska kompanija Espressif System kao nasljednika ESP8266, a proizvodi ih TSMC (Taiwan Semiconductor). [13]

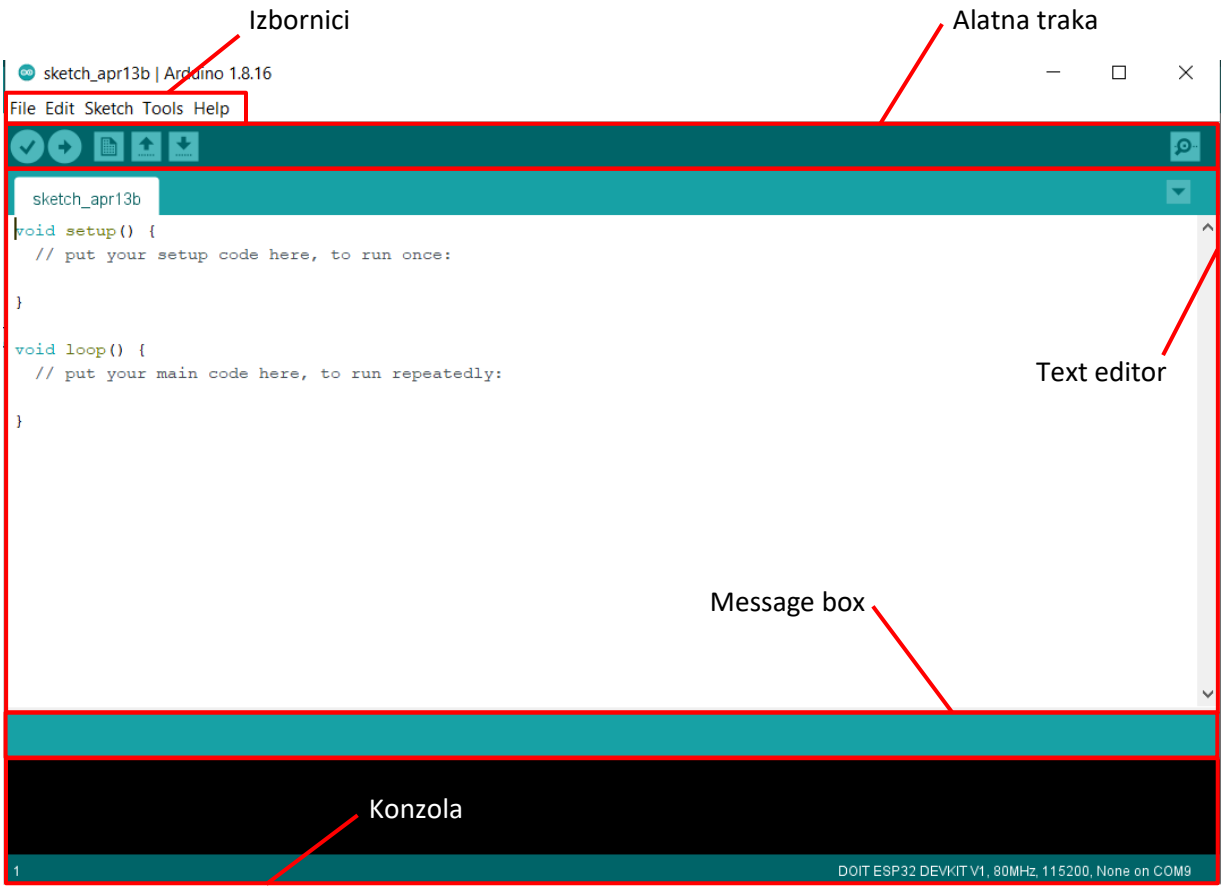
Proizvođači su pri samom razvoju obratili pažnju na to da bude koristan u puno različitih namjena, zato je prilagodljiv raznim uvjetima poput temperatura od -40°C do 125°C. Također se može prilagođivati vanjskim uvjetima zahvaljujući naprednim kalibracijskim sklopovima.

Još jedna velika prednost ESP32 je to što je namijenjen za uporabu u mobilnim uređajima tako da troši jako malo energije.

2.7.1. Arduino IDE

Arduino IDE (Integrated Development Environment) je okružje napravljeno kako bi pojednostavilo programiranje Arduina. To je okružje u kojemu lako možemo pisati i postavljati kod na bilo koju vrstu Arduina. [14]

Sastoji se od text editora za pisanje koda, područja za ispis poruka, tekstualne konzole, alatne trake i izbornika.



Slika 2.7.1.1. Arduino IDE

3. Tehničko-tehnološki dio

U ovom završnom radu uz izradu klasičnog električnog skateboarda fokus je bio na izradi ESP32 sustava koji zamjenjuje 2.4GHz upravljač kako bi se skateboardom moglo upravljati i s mobilnom aplikacijom preko Bluetooth-a. Još jedan razlog zašto je ESP32 u kombinaciji s mobilnom aplikacijom bolji od klasičnih upravljača je mogućnosti iščitavanja važnih i zanimljivih podataka sa VESC-a poput broja okretaja motora, napona baterije, potrošene struje, itd.

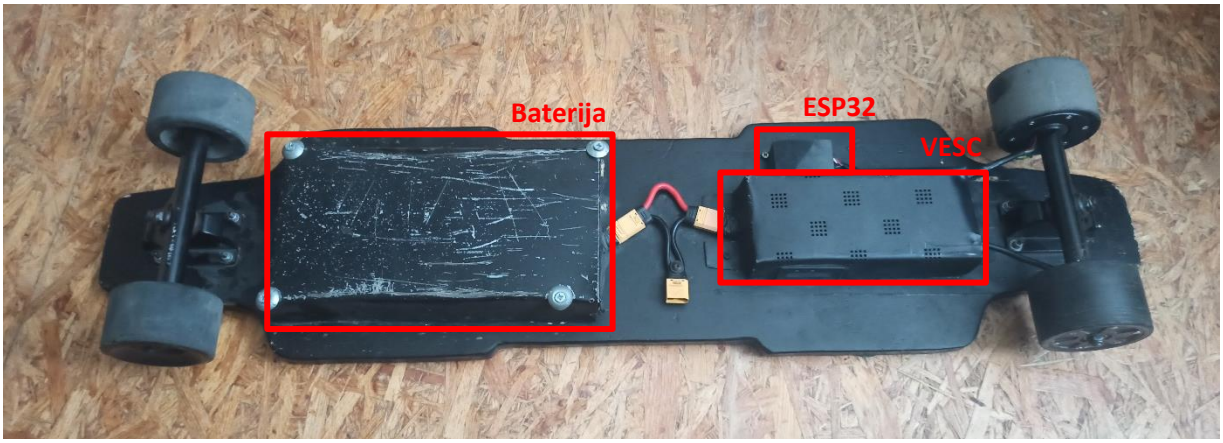
3.1. Izrada daske

Daska je napravljena izrezivanjem iz šperploče debljine 22mm. Nakon izrezivanja oblika iz šperploče obojena je u crnu boju radi ljepšeg izgleda, te je na gornju stranu nalijepljen griptape (vrsta naljepnice slične brusnom papiru koja sprječava klizanje nogu po daski i olakšava upravljanje).

Slika 3.1.1. Izrada daske

Nakon završetka same daske potrebno je učvrstiti osovine s kotačima i napraviti kućišta u kojima će se nalaziti sve komponente (baterija, VESC i ESP32). Kućišta za bateriju i VESC

su napravljena reciklažom poklopaca starih TV prijamnika i CD playera, dok je kućište za ESP32 3D printano.



Slika 3.1.2. Kučišta elektronike

3.2. Odabir baterije

Za izradu rada odabrana je litij-ionska baterija. Razlog toga je bio jednostavnost korištenja u odnosu na litij-polimer baterije koje su kompliciranije za punjenje, dok litij-ionske imaju ugrađen uređaj za balansiranje napona što pojednostavlja proces punjenja.

Baterija je konfiguracije 8S3P (tri paralele po 8 serijski spojenih baterija) što na kraju daje bateriju od 29.6V i 9Ah. Za izradu se koristi 24 18650 ćelije. Ćelije koje se koriste su Samsung Q30 18650 3000mAh 15A baterije. One su spojene u 8 grupa po 3 ćelije. U svakoj grupi ćelije su spojene paralelno te su grupe spojene serijski.

Slika 3.2.1. Izrada baterije

Na bateriju je dodan i BMS (Battery Management System) koji brine o tome da napon svih ćelija ostane isti, odnosno da se sve troše podjednako i tako osigurava da se niti jedna ne potroši ispod dozvoljenog napona ili napuni preko maksimalnog.

3.3. Odabir motora

Kao pogon odabrani us outrunner BLDC motori koji se nalaze unutar kotača. Razlog odabira BLDC motora je taj što su efikasniji i tiši od običnih brushed DC motora, a razlog odabira outrunner motora unutar kotača je taj što su puni jednostavniji za korištenje (nije potrebno raditi nosače motora, remenice ili zupčanike, koristiti lanac, itd.).

Korišteni motor je outrunner BLDC motor od 600W. Ima kV vrijednost 70kV (70 RPM/V). Može raditi na naponima od 6S do 10S (22.2V – 37V), ali spojen je na 8S bateriju (29.6V). Maksimalna brzina je 2500RPM, što u kombinaciji s kotačem promjera 90mm daje maksimalnu brzinu od oko 40km/h na 37V. Zbog baterije manjeg napona i tereta vozača skateboard na kraju može ići brzinom od oko 30km/h.

Slika 3.3.1. Motor

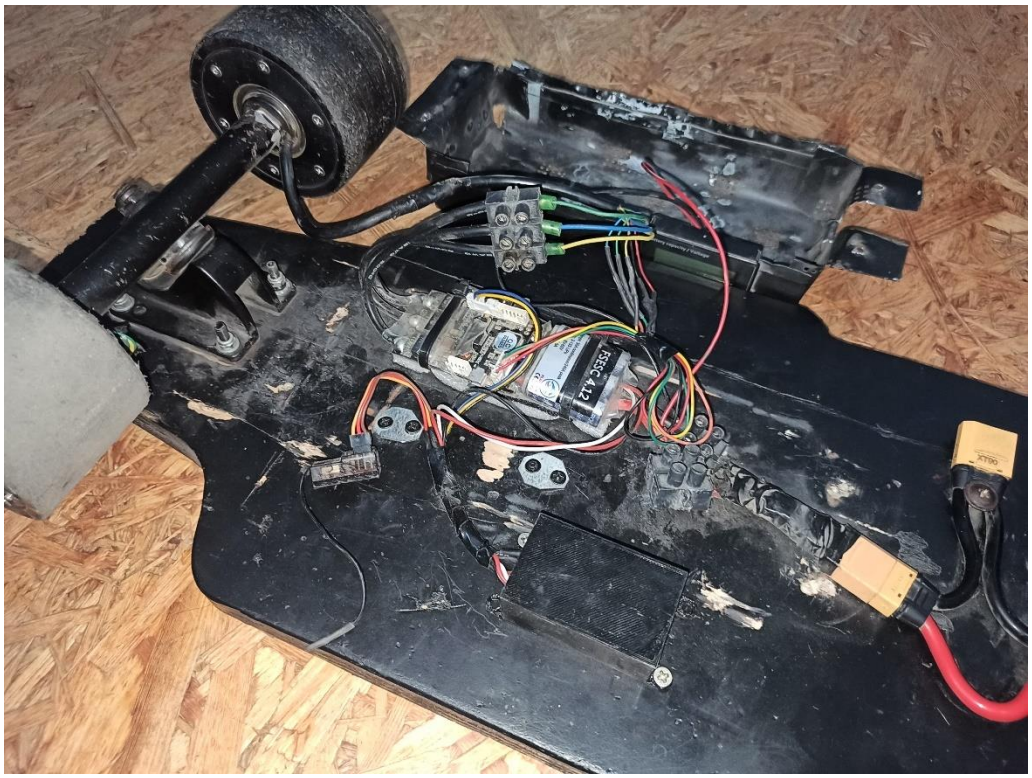
3.4. Spajanje VESC-a

VESC je glavna komponenta električnog skateboarda koja povezuje sve komponente i bez koje ništa ne bi radilo. On se spaja direktno na bateriju i sva struja do motora ide kroz njega. Osim motora na njega se spaja i upravljač (u ovom slučaju ESP32) preko žica za PWM signale. On također ima i CAN-bus, port za UART komunikaciju, port za hall senzore i SWD i mini-USB portove koji služe za programiranje. U ovom radu se ne koriste SWD i CAN portovi, ali ostali su u upotrebi.

Mini-USB port korišten je za programiranje VESC-a tako da odgovara specifikacijama baterije i motora. Također se može koristiti za nadgledanje parametara poput struje, napona, itd. u stvarnom vremenu.

Na port za hall senzore spojeni su hall senzori iz motora koji služe za praćenje pozicije motora u svakom trenutku i pomoću kojih VESC zna točno gdje je rotor i kojom se brzinom okreće kako bi njime mogao efikasnije upravljati.

Također se koristi i port za UART komunikaciju. Na njega je spojen ESP32 i on nam omogućava povlačenje podataka sa VESC-a kako bi se mogli slati na mobilnu aplikaciju i prikazivati korisniku.



Slika 3.4.1. VESC

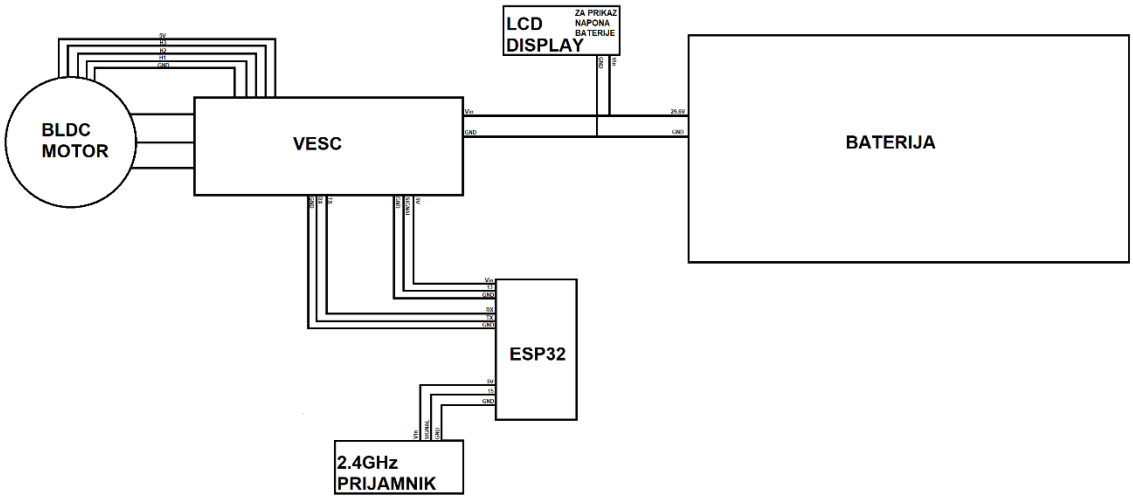
3.5. ESP32

ESP32 odabran je umjesto ostalih mikroupravljača poput Arduino UNO-a zbog svoje veličine, brzine, jednostavnosti i male potrošnje. Glavni faktor koji je prevagnuo u odabiru bio je ugrađen Bluetooth modul, za razliku od Arduino UNO-a na koji se on dodaje naknadno te zauzima više mjesta, dok je i sam Arduino UNO veći.

3.5.1. Spoj

ESP32 korišten je kao posrednik podataka između mobile aplikacije i VESC-a. On je Bluetooth-om povezan na mobitel, a na VESC je spojen na dva načina. Signalnom žicom koja služi za slanje informacije o brzini kojom bi se motori trebali okretati i UART komunikacijom koja služi za dobavljanje svih ostalih informacija sa VESC-a.

Na ESP32 također je spojen i prijemnik RC upravljača kako bi u slučaju da ne želimo ili ne možemo koristiti mobitel mogli skateboardom upravljati uz pomoć RC upravljača. ESP32 signale s upravljača prima i samo ih prosljeđuje na VESC.



Slika 3.5.1.1. Blok shema

3.6. Mobilna aplikacija

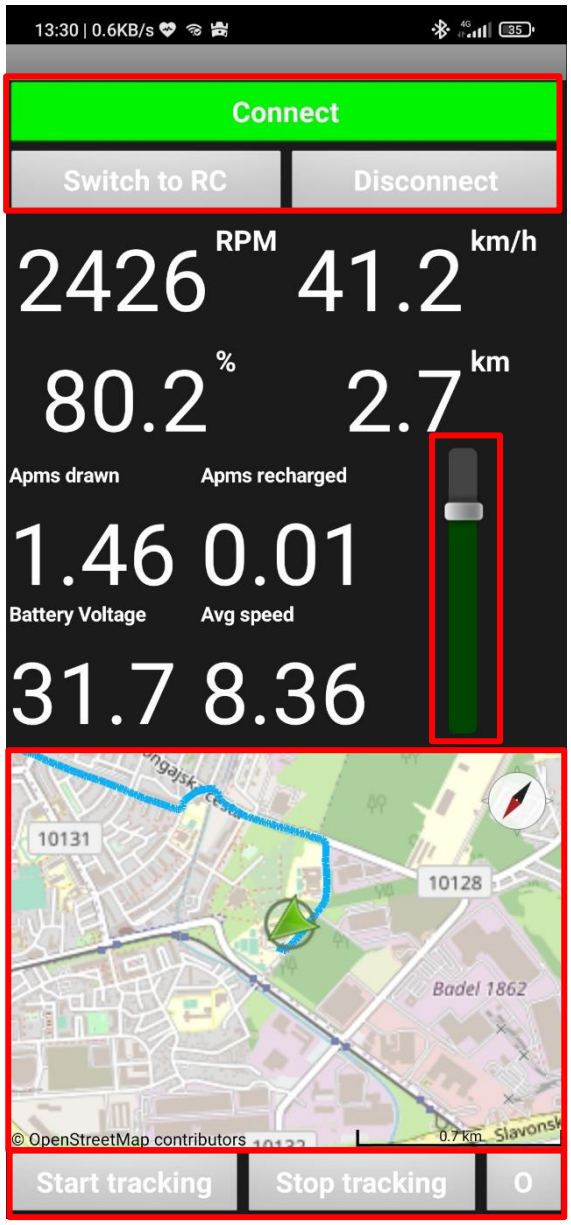
Mobilna aplikacija omogućava nam spajanje na ESP32 Bluetooth-om, upravljanje skateboardom te iščitavanje raznih podataka sa VESC-a. Za izradu aplikacije korišten je MIT App Inventor.

3.6.1. Izgled aplikacije

Aplikacija je napravljena tako da je jednostavna za korištenje i lako razumljiva. Na vrhu svi gumbi potrebni za početak rada (spajanje, prebacivanje načina kontroliranja i odspajanje). Na sredini ekrana nalaze se sve korisne informacije dobivene sa VESC-a koje bi mogle zanimati korsinika. Klizač se također nalazi na mijestu ekrana gdje je lako doći do njega prstom. Karta

se nalazi na dnu ekrana jer je u vožnji manje bitna i namijenjena primarno za pregled puta po završetku vožnje.

Važne/zanimljive informacije povučene sa VESC-a



Gumbi za povezivanje sa ESP32 i za promjenu načina kontroliranja skateboarda

Klizač za upravljanje skateboardom

Karta s mogućnosti ucrtavanja prijeđenog puta

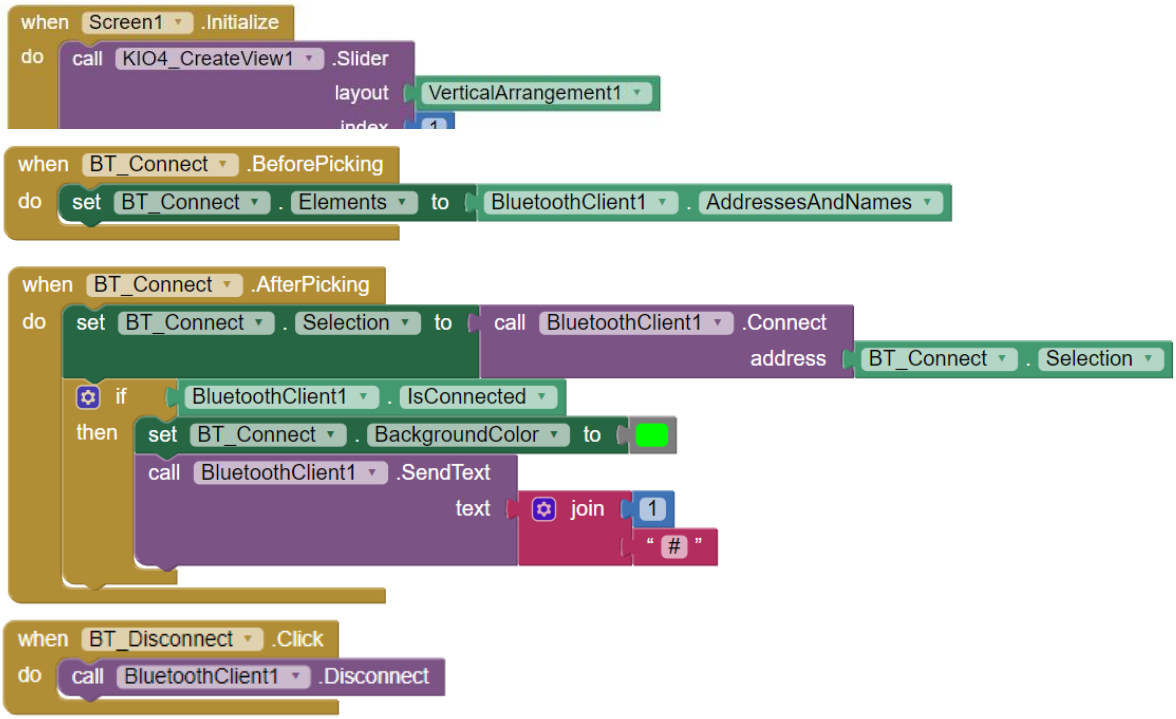
Gumbi za upravljanje kartom

Slika 3.6.2.1.Mobilna aplikacija

4. Kod

4.1. Kod mobilne aplikacije

Kod aplikacije pisan je u MIT App Inventor-u korištenjem metode programiranja blokovima. U nastavku su objašnjenja koda.

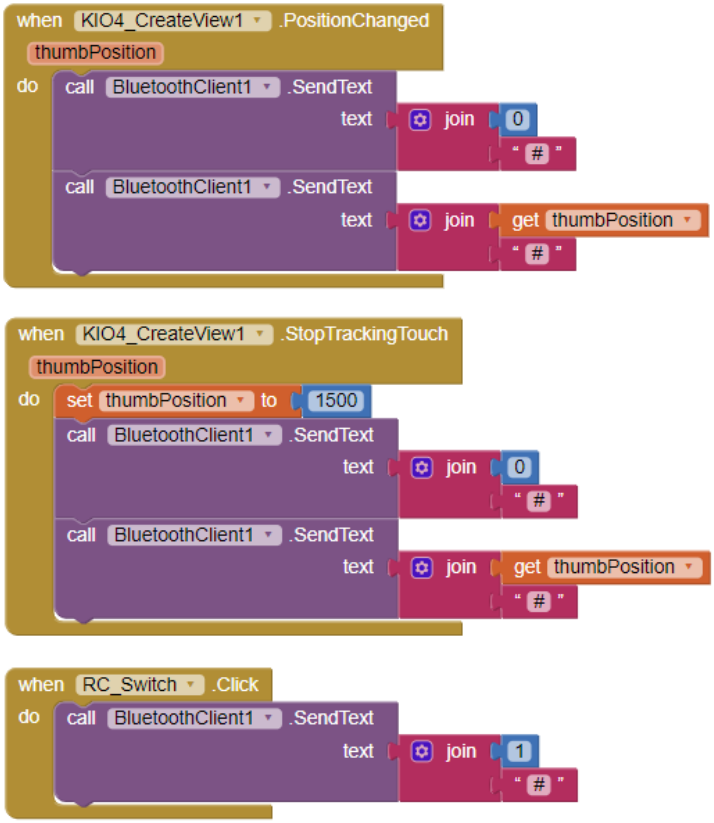


Ovaj kod izvršava se samim paljenjem aplikacije, odnosno inicijalizacijom ekrana što se događa odmah po paljenju aplikacije. Služi za kreiranje klizača kojim upravljamo skateboardom (vrijednosti klizača šalju se Bluetooth-om na ESP32 koji ih obrađuje i šalje na VESC).

Prvi blok koda služi za dodavanje svih pronađenih Bluetooth uređaja na popis s kojega možemo birati na koji uređaj ćemo se spojiti.

Drugi blok koda izvršava se nakon odabira uređaja s popisa. Njegovim izvršavanjem se pokušavamo spojiti na uređaj i ukoliko se uspješno povežemo mijenja pozadinsku boju gumba u zeleno kako bi nam prikazao uspješno povezivanje te šalje poruku na ESP32 kako bi znao da se povezao i kako bi način kontroliranja prebacio na aplikaciju.

Treći blok koda se izvršava pritiskom na Disconnect tipku u aplikaciji i odspaja nas od ESP32.

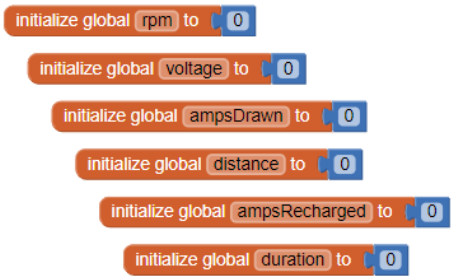


Ovaj dio koda služi za kontroliranje skateboarda.

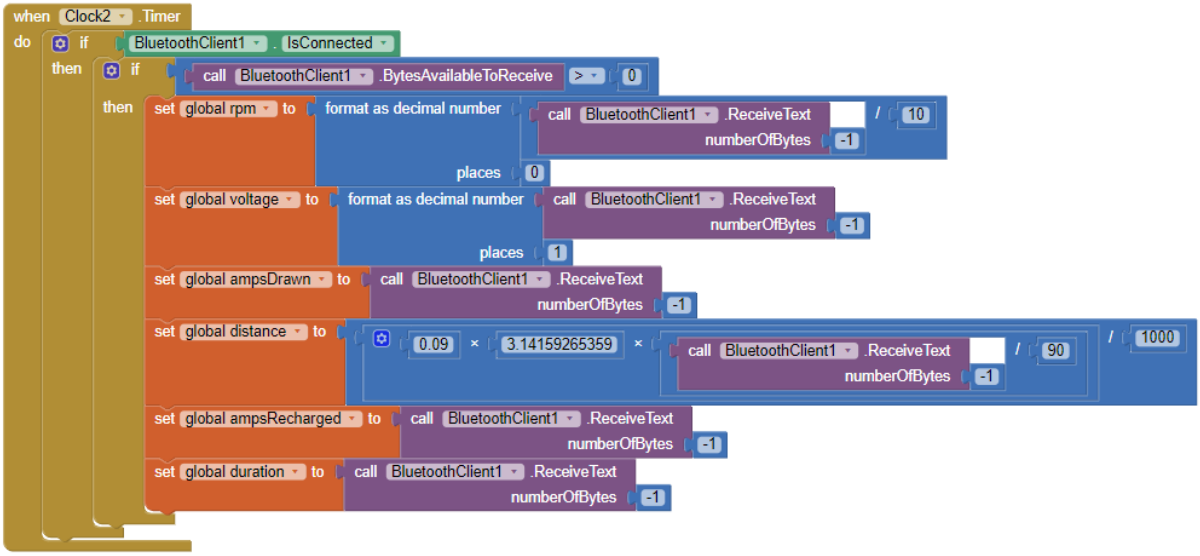
Prvi blok se izvršava kada dođe do pomicanja klizača. Tada se na ESP32 šalje poruka koja mu govori što slati na VESC kako bi se motor okretao onom brzinom kojom mi želimo.

Drugi blok se izvršava kada maknemo prst s klizača i on šalje vrijednost od 1500 na ESP32, što je srednja vrijednost (ona na kojom skateboard niti ne ubrzava niti ne koči), radi sigurnosnih razloga.

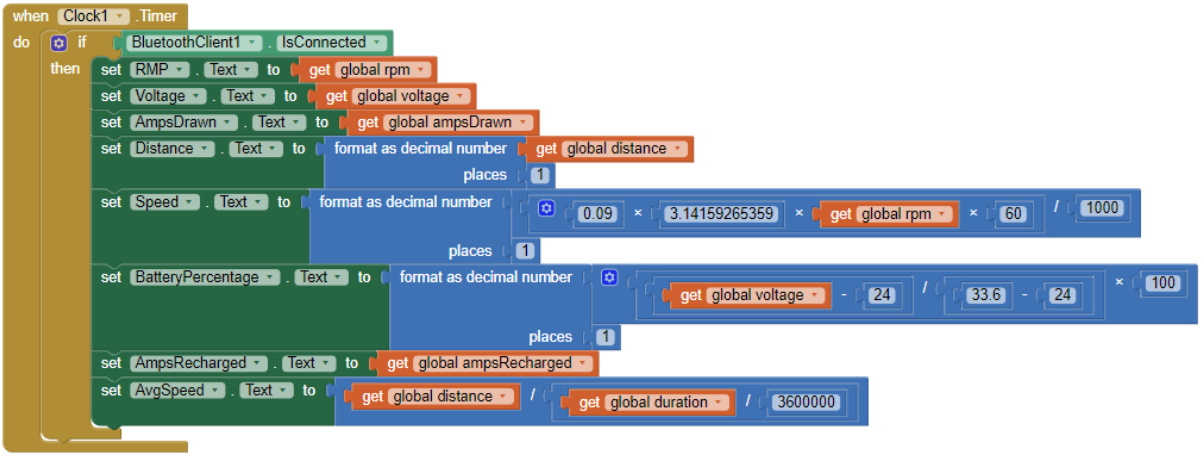
Treći dio koda služi za mijenjanje moda kontroliranja između aplikacije i RC upravljača klikom na gumb “Switch to RC”.



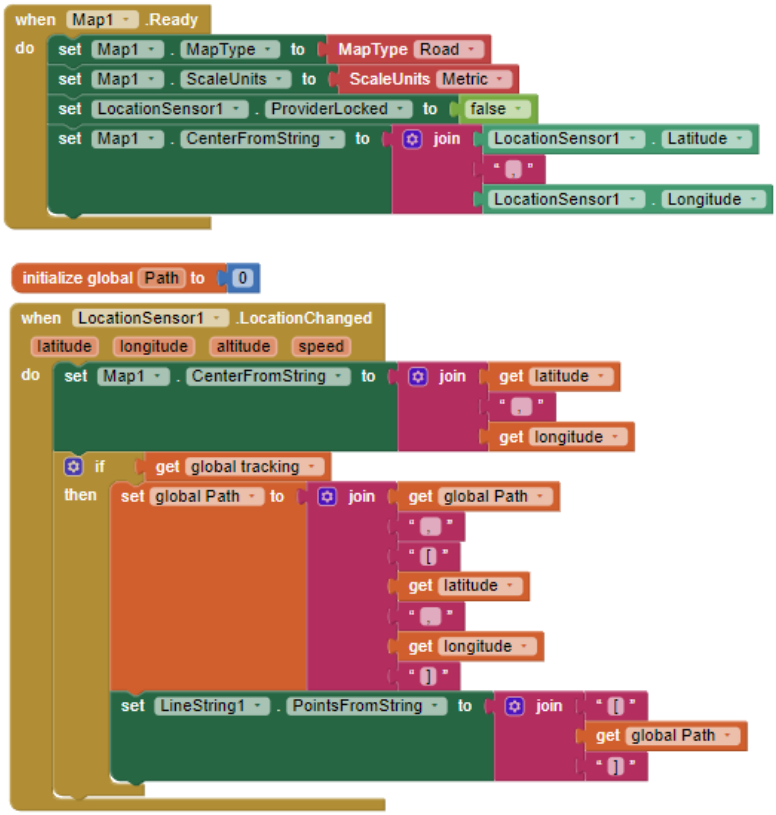
Inicijalizacija varijabli koje se primaju sa ESP32.



Ovaj dio koda izvršava se svakih 10 milisekundi. Njime se, ukoliko je mobitel povezan sa ESP32, učitavaju vrijednosti koje šalje ESP32. Te vrijednosti ESP32 dobije iz VESC-a i prosljeđuje aplikaciji kako bi se mogle izračunati informacije prikazane u aplikaciji.



Ovaj kod izvršava se svakih 100 milisekundi i njime se računaju i ispisuju vrijednosti prikazane u aplikaciji: broj okretaja, napon baterije, potrošena i napunjena energija, pređena udaljenost, brzina, postotak baterije I prosječna brzina.

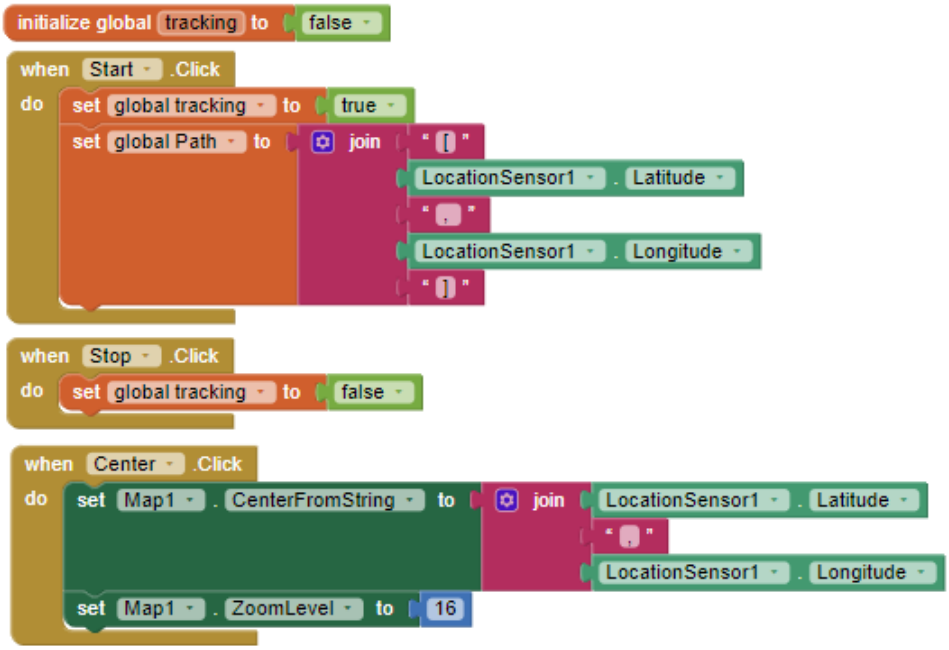


Ovaj dio koda vezan je uz funkciju karte i praćenja lokacije.

Prvi blok koda izvršava se kada se inicijalizira karta, što je neposredno nakon pokretanja aplikacije. Njime se postavljaju parametri karte poput tipa karte i mjerne jedinice. Također automatski centrira kartu na našoj lokaciji ako ju odmah pronađe.

Nakon toga inicijaliziramo varijablu Path koja služi za crtanje putanje kretanja.

Treći dio se izvršava prilikom promjene lokacije i on na varijablu Path dodaje trenutnu lokaciju te ucrtava liniju od prošle točke do sadašnje, što kreira putanju našeg kretanja.



Ovim djelom koda upravljamo kartom, odnosno praćenjem putanje.

Prvi dio koda kreira varijablu tracking bool tipa, koja nam govori da li je ucrtavanje putanje uključeno ili ne.

Drugi dio koda pokreće ucrtavanje putanje i u varijablu Path sprema našu trenutnu lokaciju kada kliknemo na “Start tracking” gumb.

Treći blok koda izvršava se klikom na “Stop tracking” i prekida ucrtavanje putanje na kartu.

Zadnji dio koda koristi se za centriranje karte na našu lokaciju.

4.2. Kod ESP32

Kod za ESP32 pisan je u Arduino IDE-u instalacijom ESP32 pločica te korištenjem BluetoothSerial, ESP32Servo i VescUart biblioteka.

BluetoothSerial biblioteka je namijenjena za uspostavu i korištenje serijske komunikacije preko Bluetooth-a koristeći ESP32.

ESP32Servo biblioteka je ista kao i Servo.h biblioteka, samo što radi na ESP32. Ona se koristi za slanje PWM signala na VESC jer su signali koje VESC prima isti kao i oni koji se

šalju na servo motore. Korištenjem writeMicroseconds() naredbe možemo odrediti koja je širina signala koji se šalje i tako slati one signale koje VESC prepoznaje kao ubrzavanje ili kočenje.

VescUart je open-source biblioteka napravljena od strane korisnika VESC-a kako bi omogućila jednostavan način pristupanja podatcima koji se nalaze u VESC-u UART komunikacijom.

```
#define Throttle 0
#define RC_switch 1

#include<BluetoothSerial.h>
#include<ESP32Servo.h>
#include<VescUart.h>

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

Servo esc;
BluetoothSerial SerialBT;
VescUart UART;

int cmd = 0;
int pos = 1500;
int interval = 100;
int previousMillis = 0;
bool useRC = true;
int RC = 15;
int RC_pos = 1500;
int duration = 0;

int a = 0;
int b = 0;
int c = 0;
bool x = true;

void setup()
{
    esc.attach(13);

    pinMode(RC, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(RC), RC_read, CHANGE);

    Serial.begin(115200);
    Serial2.begin(115200);
    while (!Serial) {; }
    UART.setSerialPort(&Serial2);

    SerialBT.begin();
```

```
    Serial.setTimeout(10);
    Serial2.setTimeout(10);
    SerialBT.setTimeout(10);
}

void loop()
{
    SerialBT.register_callback(callback);

    if (useRC)
    {
        interrupts();
        esc.writeMicroseconds(RC_pos);
    }
```

Ovaj kod izvršava se ako je ESP32 podešen tako da prima podatke s RC upravljača. On omogućava prekide koji se koriste za čitanje podataka s prijamnika upravljača te na VESC šalje očitane vrijednosti

```
    if (SerialBT.available() > 0)
    {
        cmd = SerialBT.readStringUntil('#').toInt();
        if (cmd == Throttle)
        {
            pos = SerialBT.readStringUntil('#').toInt();
            if (!useRC)
            {
                noInterrupts();
                esc.writeMicroseconds(pos);
            }
        }
        if (cmd == RC_switch)
        {
            useRC = !useRC;
        }
        SerialBT.flush();
    }
```

Ovaj kod služi za čitanje komandi primljenih preko Bluetooth-a (s mobilne aplikacije). Čitanjem prvog zaprimljenog znaka prepoznaje da li se radi o upravljanju skateboardom ili o komandi za mijenjanje kontrole između aplikacije i RC upravljača. Ukoliko se radi o brzini provjerava se da li je trenutni način kontroliranja preko aplikacije, i ako je vrijednost se šalje na VESC.

```
    if (millis() - previousMillis >= interval)
    {
        duration += 100;

        if (UART.getVescValues())
        {
            SerialBT.println(UART.data.rpm);
            SerialBT.println(UART.data.inpVoltage);
            SerialBT.println(UART.data.ampHours);
            SerialBT.println(UART.data.tachometerAbs);
            SerialBT.println(UART.data.ampHoursCharged);
            SerialBT.println(duration);
        }
        else
        {
            Serial.println("Failed to get data!");
        }
        previousMillis = millis();
    }
}
```

Ovaj kod služi za slanje podataka primljenih sa VESC u mobilnu aplikaciju.

```
void RC_read()
{
    if (digitalRead(RC) == HIGH) a = micros();
    else b = micros();
    if (b > a) RC_pos = constrain
        (map(b - a, 1180, 1920, 1000, 2000), 1000, 2000);
}
```

Funkcija RC_read izvršava se svakom promjenom stanja na signalnoj žici prijamnika RC upravljača. Ona mjeri širinu signala u mikrosekundama i to šalje na VESC (služi za čitanje PWM signala)

```
void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t* param)
{
    if (event == ESP_SPP_CLOSE_EVT ){
        useRC = true;
    }
}
```

Ova se funkcija izvršava spajanjem uređaja preko Bluetootha i služi za mijenjanje moda kontroliranja u mobilnu aplikaciju.

5. Zaključak

Cilj ovog rada bio je izrada električnog skateboarda te ugrađivanje mogućnosti povezivanja s mobitelom u njega. Sam skateboard izrađen je kao i svaki drugi eklektični skateboard. Prvo je napravljena daska na koju su kasnije stavljene sve komponente (baterija, VESC, motori) koje su međusobno povezane. Dodavanje mogućnosti spajanja mobitelom odrađeno je korištenjem ESP32 mikroupravljača kao posrednika između mobilne aplikacije i VESC-a. On u sebi ima ugrađen Bluetooth modul koji omogućava komunikaciju s mobilnom aplikacijom, a UART-om je povezan na VESC i tako može povlačiti informacije s njega.

Ovaj rad mogao bi se unaprijediti izradom nove biblioteke koja bi zamijenila VescUart. Glavni problem sa VescUart bibliotekom je što omogućuje pristup samo nekim informacijama sa VESC-a dok su mogućnosti puno veće od toga. Time bi omogućili prikaz svih informacija koje možemo vidjeti koristeći i VESC Tool što je puno opširnije i korisnije.

Još jedan način kojim bi se mogao unaprijediti ovaj završni rad je poboljšavanjem samih komponenti skateboarda što bi omogućilo puno veći domet i brzinu skateboarda te sposobnost vožnje po svim terenima, odnosno ne bi bio ograničen samo na cestu što bi uvelike poboljšalo njegovu korisnost.

6. Literatura

1. The History of Skateboarding & The Evolution of The Electric Skateboard. 14.5.2019. URL: https://transportationevolved.com/history-of-skateboarding-electric-skateboard/ (pristupljeno: 29.3.2022.)

2. Electric Skateboard Batteries – All you need to know. 12.3.2020. URL: https://e-skateboarder.com/batteries/ (pristupljeno: 31.3.2022.)

3. What is Electronic Speed Control (ESC) & Its Working. URL: https://www.elprocus.com/electronic-speed-control-esc-working-applications/ (pristupljeno: 1.4.2022.)

4. What is an Electronic Speed Controller and how does it differ from brushed to brushless motors?. URL: https://www.modelflight.com.au/blog/electronic-speed-controllers (pristupljeno: 1.4.2022.)

5. Are ESC required for all motors? 20.9.2017. URL: https://www.rcgroups.com/forums/showthread.php?2960235-Are-ESC-required-for-all-motors (pristupljeno: 1.4.2022.)

6. Understanding ESC & VESC in Electric Skateboard: How to choose? 22.11.2019. URL: https://www.electricskateboardhq.com/understanding-electronic-speed-controller-esc-and-vesc/ (pristupljeno: 1.4.2022.)

7. What's VESC? How to use it ? What's difference between VESC and ESC? 5.5.2020. URL: https://spintend.com/blogs/news/whats-vesc-how-to-use-it-whats-difference-between-vesc-and-normal-esc (pristupljeno: 1.4.2022.)

8. VESC Tool. URL: https://vesc-project.com/vesc_tool (pristupljeno: 8.4.2022.)

9. All About Brushless DC Motors. URL: https://www.thomasnet.com/articles/machinery-tools-supplies/brushless-dc-motors/ (pristupljeno: 1.4.2022.)

10. How Brushless Motor and ESC Work. URL: https://howtomechatronics.com/how-it-works/how-brushless-motor-and-esc-work/ (pristupljeno: 1.4.2022.)

11. What is the Difference Between Brushed and Brushless DC Motors. URL: https://www.parvalux.com/what-is-the-difference-between-brushed-and-brushless-dc-motors (pristupljeno: 8.4.2022.)

12. Brushless Inrunner vs Outrunner motor? 2.8.2018. URL: https://www.radiocontrolinfo.com/brushless-inrunner-vs-outrunner-motor/ (pristupljeno: 8.4.2022.)

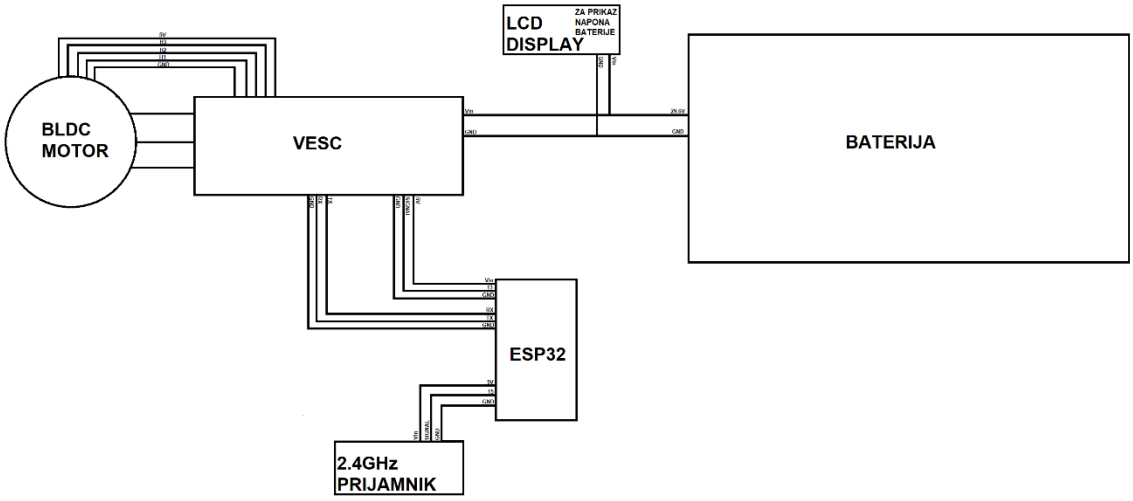
13. About Us. URL: https://appinventor.mit.edu/about-us (pristupljeno: 14.4.2022.)

14. ESP32. URL: https://www.espressif.com/en/products/socs/esp32 (pristupljeno: 14.4.2022.)

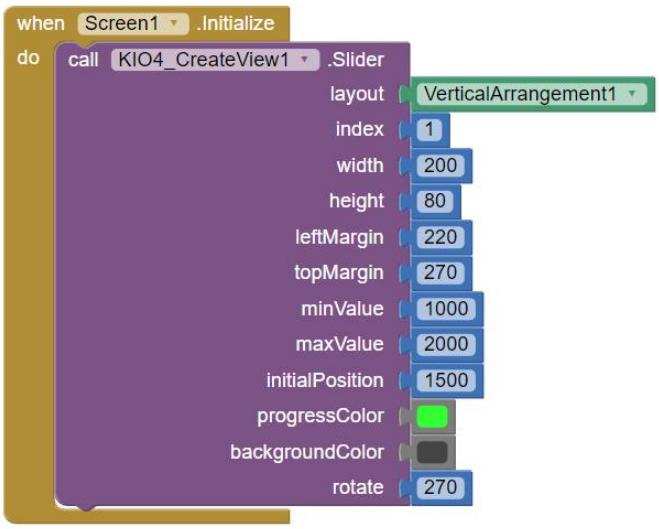
15. Arduino IDE. URL: https://www.arduino.cc/en/software (pristupljeno: 15.4.2022.)

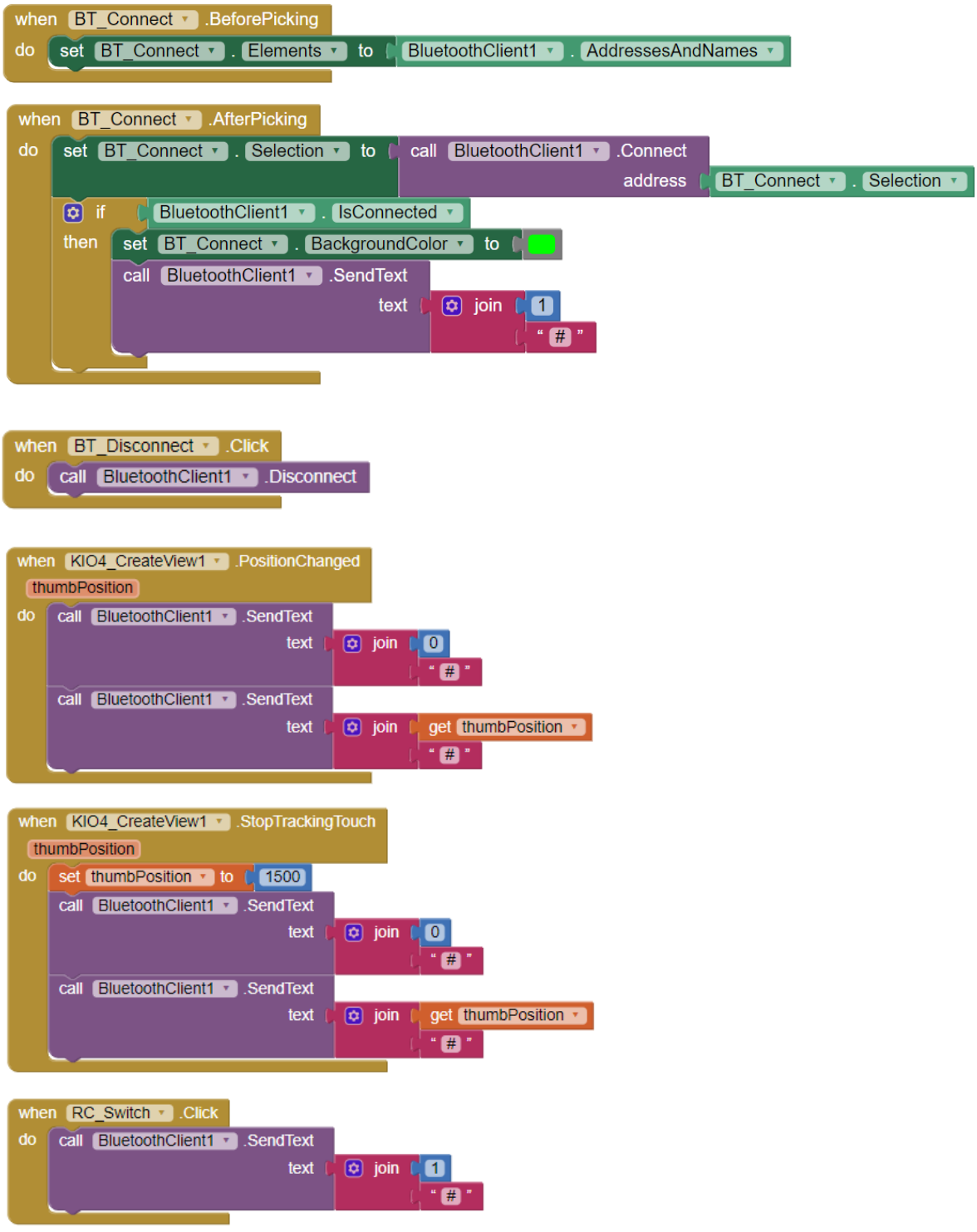
7. Prilozi

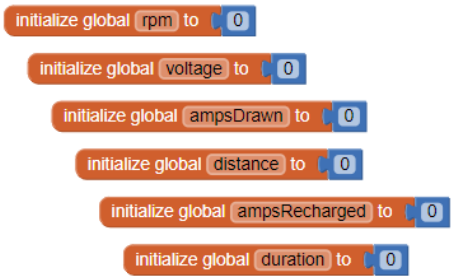
7.1. Shema

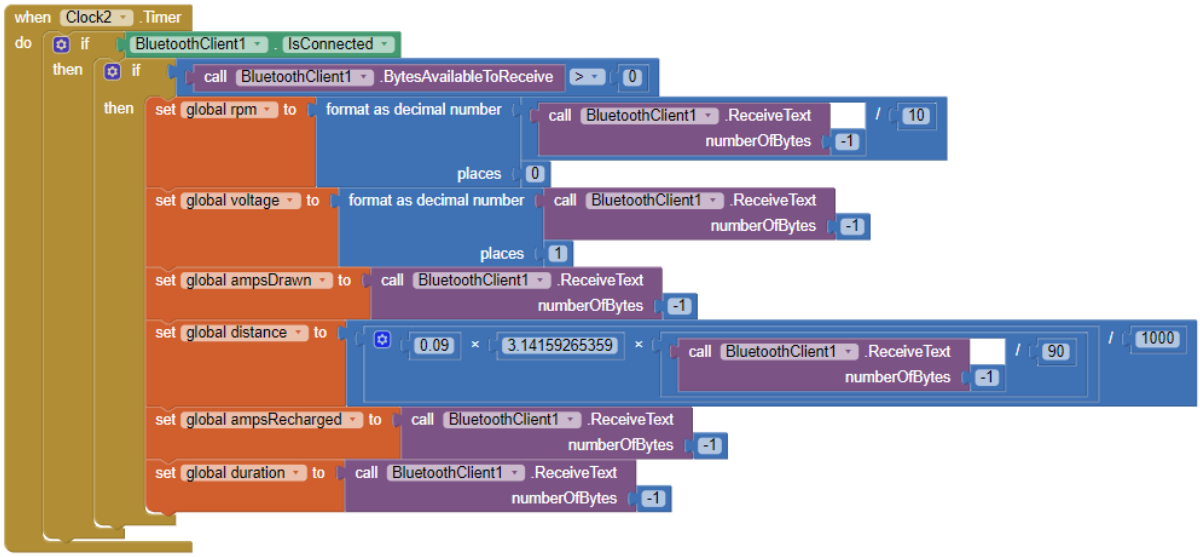


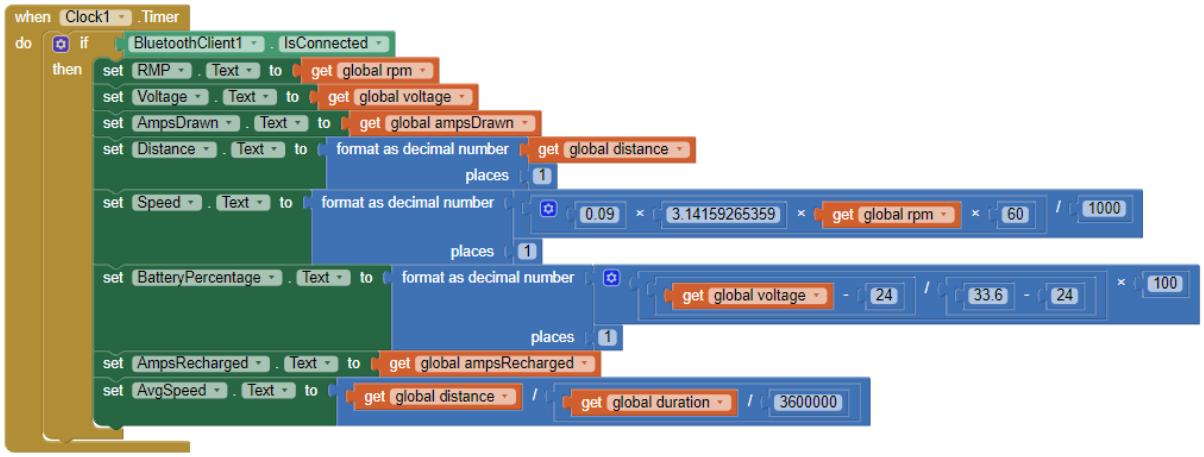
7.2. Kod mobilne aplikacije

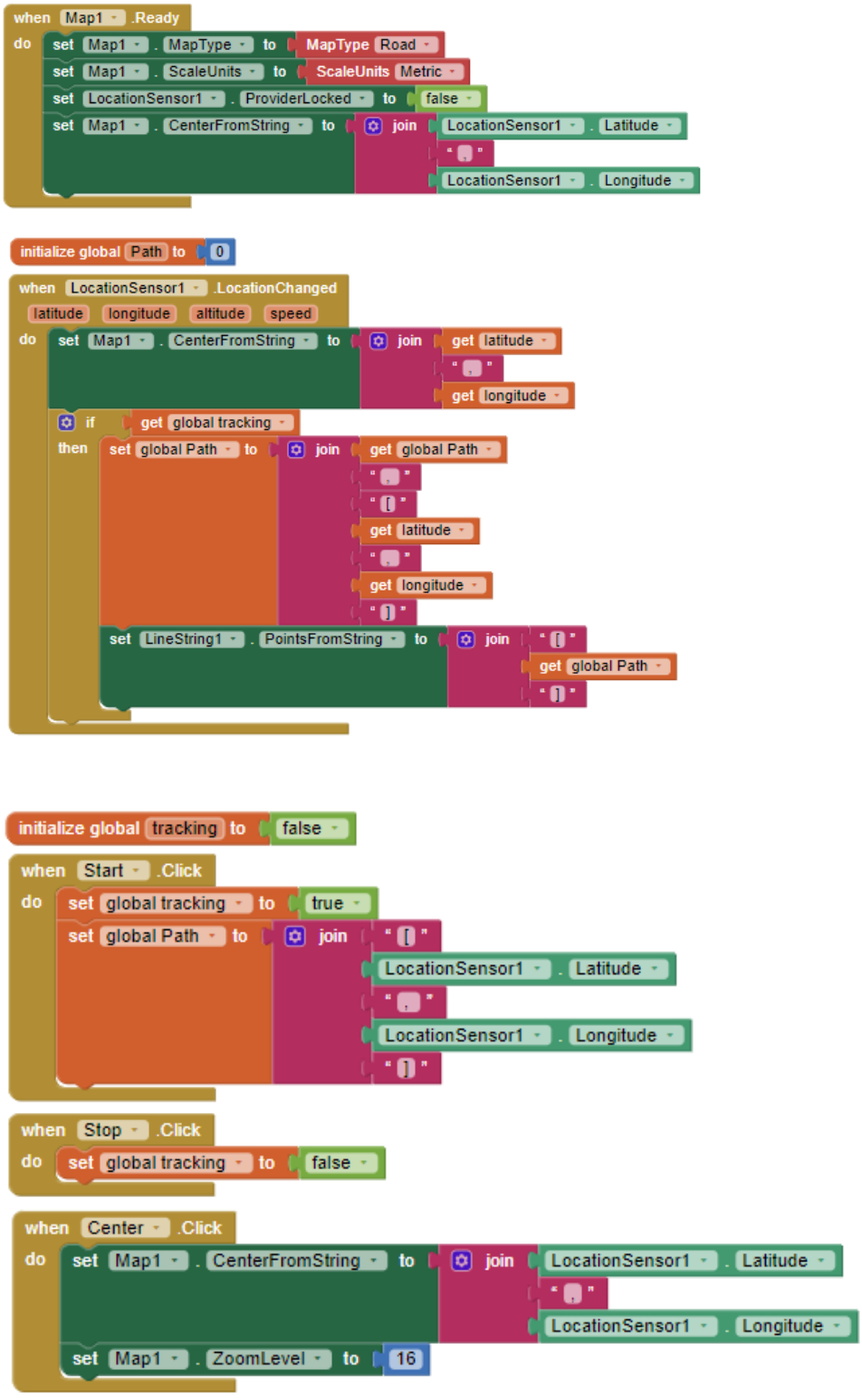



when BT_Connect .BeforePicking
do set BT_Connect . Elements to BluetoothClient1 . AddressesAndNames
when BT_Connect .AfterPicking
do set BT_Connect . Selection to call BluetoothClient1 .Connect
address BT_Connect . Selection
if BluetoothClient1 . IsConnected
then set BT_Connect . BackgroundColor to
call BluetoothClient1 .SendText
text join 1
"#"
when BT_Disconnect .Click
do call BluetoothClient1 .Disconnect
when KIO4_CreateView1 .PositionChanged
thumbPosition
do call BluetoothClient1 .SendText
text join 0
"#"
call BluetoothClient1 .SendText
text join get thumbPosition
"#"
when KIO4_CreateView1 .StopTrackingTouch
thumbPosition
do set thumbPosition to 1500
call BluetoothClient1 .SendText
text join 0
"#"
call BluetoothClient1 .SendText
text join get thumbPosition
"#"
when RC_Switch .Click
do call BluetoothClient1 .SendText
text join 1
"#"


initialize global rpm to 0
initialize global voltage to 0
initialize global ampsDrawn to 0
initialize global distance to 0
initialize global ampsRecharged to 0
initialize global duration to 0


when Clock2 .Timer
do if BluetoothClient1 . IsConnected
then if call BluetoothClient1 .BytesAvailableToReceive > 0
then set global rpm to format as decimal number call BluetoothClient1 .ReceiveText numberOfBytes -1 / 10
places 0
set global voltage to format as decimal number call BluetoothClient1 .ReceiveText numberOfBytes -1
places 1
set global ampsDrawn to call BluetoothClient1 .ReceiveText numberOfBytes -1
set global distance to 0.09 × 3.14159265359 × call BluetoothClient1 .ReceiveText numberOfBytes -1 / 90 / 1000
set global ampsRecharged to call BluetoothClient1 .ReceiveText numberOfBytes -1
set global duration to call BluetoothClient1 .ReceiveText numberOfBytes -1


when Clock1 .Timer
do if BluetoothClient1 . IsConnected
then set RMP . Text to get global rpm
set Voltage . Text to get global voltage
set AmpsDrawn . Text to get global ampsDrawn
set Distance . Text to format as decimal number get global distance
places 1
set Speed . Text to format as decimal number 0.09 × 3.14159265359 × get global rpm × 60 / 1000
places 1
set BatteryPercentage . Text to format as decimal number get global voltage - 24 / 33.6 - 24 × 100
places 1
set AmpsRecharged . Text to get global ampsRecharged
set AvgSpeed . Text to get global distance / get global duration / 3600000


when Map1 .Ready
do set Map1 . MapType to MapType Road
set Map1 . ScaleUnits to ScaleUnits Metric
set LocationSensor1 . ProviderLocked to false
set Map1 . CenterFromString to join LocationSensor1 . Latitude
" , "
LocationSensor1 . Longitude
initialize global Path to 0
when LocationSensor1 .LocationChanged
latitude longitude altitude speed
do set Map1 . CenterFromString to join get latitude
" , "
get longitude
if get global tracking
then set global Path to join get global Path
" , "
" ["
get latitude
" , "
get longitude
"] "
set LineString1 . PointsFromString to join " ["
get global Path
"] "
initialize global tracking to false
when Start .Click
do set global tracking to true
set global Path to join " ["
LocationSensor1 . Latitude
" , "
LocationSensor1 . Longitude
"] "
when Stop .Click
do set global tracking to false
when Center .Click
do set Map1 . CenterFromString to join LocationSensor1 . Latitude
" , "
LocationSensor1 . Longitude
set Map1 . ZoomLevel to 16

7.3. ESP32 kod

```
#define Throttle 0
#define RC_switch 1

#include<BluetoothSerial.h>
#include<ESP32Servo.h>
#include<VescUart.h>

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

Servo esc;
BluetoothSerial SerialBT;
VescUart UART;

int cmd = 0;
int pos = 1500;
int interval = 100;
int previousMillis = 0;
bool useRC = true;
int RC = 15;
int RC_pos = 1500;
int duration = 0;

int a = 0;
int b = 0;
int c = 0;
bool x = true;

void setup()
{
    esc.attach(13);

    pinMode(RC, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(RC), RC_read, CHANGE);

    Serial.begin(115200);
    Serial2.begin(115200);
    while (!Serial) {; }
    UART.setSerialPort(&Serial2);

    SerialBT.begin();
    Serial.setTimeout(10);
    Serial2.setTimeout(10);
    SerialBT.setTimeout(10);
}

void loop()
{
    SerialBT.register_callback(callback);

    if (useRC)
    {
        interrupts();
```

```
        esc.writeMicroseconds(RC_pos);
    }


    if (SerialBT.available() > 0)
    {
        cmd = SerialBT.readStringUntil('#').toInt();
        if (cmd == Throttle)
        {
            pos = SerialBT.readStringUntil('#').toInt();
            if (!useRC)
            {
                noInterrupts();
                esc.writeMicroseconds(pos);
            }
        }
        if (cmd == RC_switch)
        {
            useRC = !useRC;
        }
        SerialBT.flush();
    }

    if (millis() - previousMillis >= interval)
    {
        duration += 100;

        if (UART.getVescValues())
        {
            SerialBT.println(UART.data.rpm);
            SerialBT.println(UART.data.inpVoltage);
            SerialBT.println(UART.data.ampHours);
            SerialBT.println(UART.data.tachometerAbs);
            SerialBT.println(UART.data.ampHoursCharged);
            SerialBT.println(duration);
        }
        else
        {
            Serial.println("Failed to get data!");
        }
        previousMillis = millis();
    }
}


void RC_read()
{
    if (digitalRead(RC) == HIGH) a = micros();
    else b = micros();
    if (b > a) RC_pos = constrain
        (map(b - a, 1180, 1920, 1000, 2000), 1000, 2000);
}


void callback(esp_spp_cb_event_t event, esp_spp_cb_param_t* param)
```

```
{
    if (event == ESP_SPP_CLOSE_EVT ){
        useRC = true;
    }
}
```