# Sadržaj

[1 Uvod 1](#_Toc201835513)

[2 Hardverska implementacija 2](#_Toc201835514)

[2.1 Popis komponenti 2](#_Toc201835515)

[2.1.1 FireBeetle 2 ESP32-E 4](#_Toc201835516)

[2.1.2 Hobbywing Quicrun 16BL30 ESC 6](#_Toc201835517)

[2.1.3 Hobbywing Quicrun 2435 SL Brushless Motor G3 8](#_Toc201835518)

[2.1.4 Tower Pro SG90 standard servo motor 9](#_Toc201835519)

[2.1.5 OVONIC 7.4V 2200mAh 50C Li-Po baterije 11](#_Toc201835520)

[2.1.6 DFPlayer Pro 13](#_Toc201835521)

[2.1.7 WS2812B RGB LED diode 14](#_Toc201835522)

[2.1.8 Sony DualShock 4 16](#_Toc201835523)

[2.2 Logička shema fizičkih konekcija 17](#_Toc201835524)

[2.3 Modifikacija LEGO Technic modela 19](#_Toc201835525)

[3 Programski aspekt 22](#_Toc201835526)

[3.1 Korištene biblioteke 22](#_Toc201835527)

[3.2 Implementacija funkcija automobila 23](#_Toc201835528)

[3.3 Detaljan pregled implementiranih funkcija 24](#_Toc201835529)

[4 3D dizajn i ispis 30](#_Toc201835530)

[4.1 Pregled tehnika 3D ispisa 31](#_Toc201835531)

[4.2 Filamenti za 3D ispis 34](#_Toc201835532)

[4.3 Upoznavanje s Bambu Lab A1 Mini 3D printerom 36](#_Toc201835533)

[4.4 Odabir programa za 3D dizajn 38](#_Toc201835534)

[4.5 Proces dizajna 40](#_Toc201835535)

[4.6 Proces ispisa 41](#_Toc201835536)

[5 Zaključak 43](#_Toc201835537)

[6 Literatura 44](#_Toc201835538)

[8 Prilozi 47](#_Toc201835539)

# Popis oznaka i kratica

|  |  |  |
| --- | --- | --- |
| **Kratica** | **Puno značenje (engl.)** | **Puno značenje (hrv.)** |
| IDE | Integrated Development Environment | Integrirano razvojno okruženje |
| DC | Direct Current | Istosmjerna struja |
| LED | Light-Emitting Diode | Svjetleća dioda |
| ESC | Electronic Speed Controller | Elektronička kontrola brzine |
| IO | Input/Output | Ulaz/izlaz |
| UBEC | Universal Battery Elimination Circuit | Univerzalni krug za uklanjanje baterije |
| OTA | Over-The-Air | Bežična ažuriranja |
| SRAM | Static Random Access Memory | Statična radna memorija |
| BR/EDR | Basic Rate/Enchanced Data Rate | - |
| BLE | Bluetooth Low Energy |  |
| UART | Universal Asynchronous Receiver/Transmitter | Univerzalni asinkroni prijamnik-odašiljač |
| USB | Universal Serial Bus | Univerzalna serijska sabirnica |
| SPI | Serial Peripheral Interface | - |
| I2C | Inter-Integrated Circuit | - |
| I2S | Inter-Integrated Circuit Sound | - |
| IC | Integrated Circuit | Integrirani krug |
| Pb-Ac | Lead-Acid | Olovna baterija |
| Ni-Cd | Nickel-Cadmium | Nikal-kadmijeva baterija |
| Ni-MH | Nickel-Metal Hybride | Nikal-metal-hibridna baterija |
| Li-Ion | Lithium Ion | Litij-ionska baterija |
| Li-Po | Lithium Polymer | Litij-polimer baterija |
| DAC | Digital to Analogue Converter | Digitalno-analogni pretvornik |
| MP3 | MPEG Audio Layer 3 | - |
| WAV | Waveform Audio File Format | - |
| WMA | Windows Media Audio | - |
| FLAC | Free Lossless Audio Codec | - |
| AAC | Advanced Audio Coding | - |
| APE | Monkey's Audio | - |
| SMD | Surface Mounted Device | Uređaj za površinsku montažu |
| FDM | Fused Deposition Modeling | Izrada filamenta u slobodnom obliku |
| SLA | Stereolithography | Stereolitografija |
| UV | Ultraviolet | Ultraljubičasto zračenje |
| DLP | Digital Light Projection | Očvršćivanje s digitalno obrađenim svjetlosnim signalom |
| SLS | Selective Laser Sintering | Selektivno lasersko sinteriranje |
| DLSM | Direct Metal Laser Sintering | - |
| SLM | Selective Metal Sintering | - |
| EBM | Electron Beam Melting | Aditivna proizvodnja elektronskim snopom |
| SHS | Selective Heat Sintering | Selektivno toplinsko sinteriranje |
| 3DP | 3D Printing | Trodimenzionalni ispis |
| LOM | Laminated Object Manufacturing | Proizvodnja laminiranih objekata |
| UAM | Ultrasound Additive Manufacturing | Ultrazvučna aditivna proizvodnja |
| LENS | Laser Engineered Net Shaping | - |
| DMT | Direct Metal Tooling | - |
| BeAM | Be Additive Manufacturing | - |
| PLA | Poly-Lactic Acid | Polilaktična kiselina |
| PETG | Polyethylene Terephthalate Glycol-Modified | Polietilen tereftalat modificiran glikolom |
| ABS | Acrylo-nitrile Butadiene Styrene | Akrilonitril butadien stiren |
| ASA | Acrylo-nitrile Styrene Acrylate | Akrilonitril stiren akrilat |
| PVA | Polyvinyl Alcohol | Polivinil alkohol |
| PC | Polycarbonate | Polikarbonat |
| PPS | Polyphenylene Sulfide | Polifenilen sulfid |
| TPE | Thermoplastic Elastomers | Termoplastični elastomer |
| TPU | Thermoplastic Polyurethane | Termoplastični poliuretan |
| AMS | Automatic Material System | Automatski sustav materijala |
| CAD | Computer-Aided Design | Računalno potpomognuto projektiranje |

# Popis slika

[Slika 1 - FireBeetle 2 ESP32-E mikroupravljač [1] 4](#_Toc201676339)

[Slika 2 - Hobbywing Quicrun 16BL30 ESC [3] 6](#_Toc201676340)

[Slika 3 - Hobbywing Quicrun 2435 G3 [5] 8](#_Toc201676341)

[Slika 4 - Tower Pro SG90 [autor] 10](#_Toc201676342)

[Slika 5 - Tablica čestih sastava baterija i njihove karakteristike [7] 12](#_Toc201676343)

[Slika 6 - DFPlayer Pro [9] 14](#_Toc201676344)

[Slika 7 - Dijagram WS2812B RGB LED diode [10] 16](#_Toc201676345)

[Slika 8 - Prijenos podataka između dioda [10] 17](#_Toc201676346)

[Slika 9 - DualShock 4 kontroler [11] 18](#_Toc201676347)

[Slika 10 - Logička shema fizičkih konekcija [autor] 20](#_Toc201676348)

[Slika 11 - Ford GT 2022 Lego Technic set [12] 21](#_Toc201676349)

[Slika 12 - Modificirani model [autor] 22](#_Toc201676350)

[Slika 13 - Legenda funkcija mapiranih na Sony DualShock 4 kontroleru [autor] 26](#_Toc201676351)

[Slika 14 - Način rada DLP i SLA tehnologija ispisa [17] 34](#_Toc201676352)

[Slika 15 - Pojednostavljen dijagram tehnologije SLS 3D ispisa [18] 35](#_Toc201676353)

[Slika 16 - Bambu Lab A1 Mini 3D printer [20] 36](#_Toc201676354)

[Slika 17 - Primjer filamenta u obliku niti na špuli [21] 37](#_Toc201676355)

[Slika 18 - Bambu Lab A1 Mini u neaktivnom stanju [autor] 39](#_Toc201676356)

[Slika 19 - Početno sučelje FreeCAD programa [autor] 41](#_Toc201676357)

[Slika 20 - Prazno FreeCAD sučelje spremno za dizajn modela [autor] 42](#_Toc201676358)

[Slika 21 - Adapteri za servo i DC motore [autor] 43](#_Toc201676359)

[Slika 22 - OrcaSlicer sučelje [autor] 44](#_Toc201676360)

[Slika 23 - Primjer sučelja spremnog za ispis [autor] 45](#_Toc201676361)

# Uvod

Vozila na daljinsko upravljanje često se opisuju kao igračke za djecu. No kao i u svakom drugom hobiju, postoje entuzijasti koji guraju svoje područje interesa naprijed. Kroz godine je nastala podjela između vozila na daljinsko upravljanje na jeftinije, manje kvalitetne igračke i skuplje, kvalitetnije komponente koje spadaju u hobi razred. Komponente hobi razreda se još dodatno dijele prema pogonu vozila koji može biti električni ili nitro. S razvojem tehnologija baterija je električni pogon postao puno češći, no oba su i dalje prisutna te imaju svoje prednosti i mane. Fokus ovog diplomskog rada je na izradi vozila na daljinsko upravljanje s električnim pogonom. Za izradu prototipa mora se odabrati niz fizičkih komponenti koje će obavljati funkcije vozila, napisati programski kod koji će upravljati komponentama i komunicirati s daljinskim upravljačem te koristiti tehnologiju 3D ispisa u svrhu brze izrade prototipa adaptera izlaznih osovina ugrađenih motora. Kao osnovna baza rada koristit će se LEGO Technic set dok će ostale komponente biti hobi razreda, javno dostupni proizvodi kompatibilni s Arduino platformom i ESP32 mikroupravljačem te po mjeri rađene komponente tehnologijom 3D ispisa. Diplomski rad je sastavljen od tri glavna dijela: hardverska implementacija, programska implementacija i 3D dizajn/ispis. Kroz poglavlje hardverske implementacije će se proći bitne značajke odabranih komponenti, njihove specifikacije i funkcije, pojasnit će se sve fizičke konekcije kroz logičku shemu i promjene napravljene na LEGO Technic modelu kako bi mogli smjestiti svi dijelovi. Nakon toga slijedi programski aspekt rada koji se primarno odnosi na proces programiranja u Arduino IDE (engl. Integrated Development Environment) okruženju. Opisati će se sve korištene biblioteke i pojasniti će se bitni dijelovi programskog koda odgovorni za upravljanje većinom funkcionalnosti poput kretanja unaprijed i unazad te skretanja. Zatim je na redu 3D dizajn i ispis gdje će se ukratko opisati tehnike 3D ispisa, upoznati s dostupnim 3D printerom te programima za dizajn i ispis. Krajnji cilj je kreirati funkcionalno vozilo na daljinsko upravljanje koje s daljinskim upravljačem komunicira putem Bluetooth veze. Ulogu prijamnika vozila preuzima varijanta ESP32 mikroupravljača koji će naredbe poslane s daljinskog upravljača protumačiti i proslijediti ispravnoj komponenti, a sve to zahtijeva pažljivo planiranu i izvedenu hardversku implementaciju, koja predstavlja početnu fazu projekta.

# Hardverska implementacija

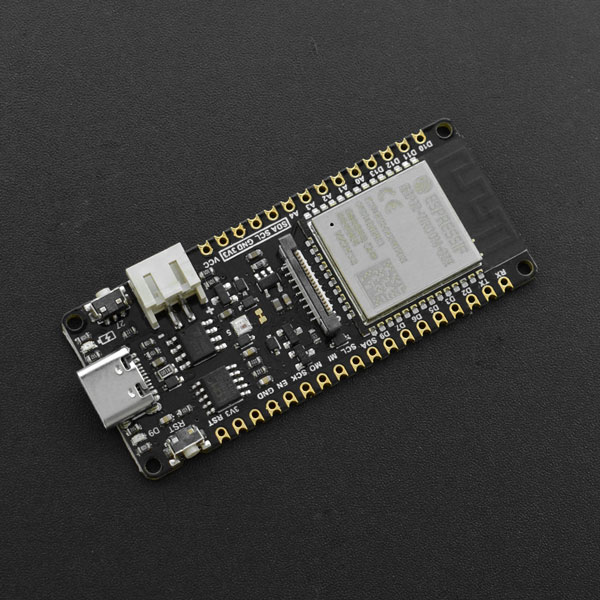
Po pitanju samih komponenti, hardverska implementacija je jednostavna. Potrebno je imati izvor napajanja s velikom brzinom pražnjenja, žice s dovoljnom propusnošću električne energije, pretvarače napona koji će pružati odgovarajući napon svim komponentama, servo motor koji će biti odgovoran za upravljanje prednjim kotačima, DC (engl. Direct Current) motor dovoljnog okretnog momenta za pokretanje stražnjih kotača, nekolicina RGB LED (engl. Light-Emitting Diode) dioda, par zvučnika i mikroupravljač koji će upravljati svim komponentama. U velikoj većini slučajeva, za izvor napajanja se obično koriste baterije, no postoje razlike između kemijskih sastava pojedinih baterija. Glavne karakteristike baterija bitne za ovaj diplomski rad je napon i broj ćelija jer to određuje ESC (engl. Electronic Speed Controller), DC motor, servo motor i potrebne pretvarače napona za napajanje ostalih komponenti. Kao platforma za ugradnju svih komponenti odabran je Lego Technic model, kojem je, prije same montaže, potrebno napraviti nekoliko izmjena kako bi se sve komponente pravilno smjestile u sam model. U nastavku će biti definirane sve komponente.

## Popis komponenti

Popis komponenti nije dugačak, no važno je osigurati kompatibilnost. Najbitnija karakteristika svih odabranih komponenti je mogućnost međusobne komunikacije. Kako bi se uopće krenulo s odabirom komponenti, potrebno je definirati bazu diplomskog rada, tj. model automobila u koji će se ugrađivati komponente. Za tu svrhu bit će korišten LEGO Technic Ford GT 2022 set #42154. U slučaju da službeni model nema dovoljno mjesta, postoje neslužbene, alternativne konfiguracije dostupne na internetu. Za ulogu prijamnika i mozga rada odabran je FireBeetle 2 ESP32-E zbog ugrađenog modula za Bluetooth i kompaktnog dizajna te podrške za IO (engl. Input/Output) shield u svrhu lakšeg napajanja. Upravljanje prednjim kotačima je prepušteno Tower Pro SG90 servo motoru dok je za stražnji pogon zadužen Hobbywing Quicrun 2435 SL Brushless Motor G3. DC motor treba ESC kako bi funkcionirao i zbog toga je odabran Hobbywing Quicrun 16BL30. Za sve zvukove motora su odgovorni Fermion DFPlayer Pro MP3 modul i dva 8 Ohm 3W zvučnika. Prednja i stražnja svjetla su emulirana pomoću četiri WS2812B RBG LED diode. Napajanje se sastoji od dvije Ovonic 7.4V 2200mAh Li-Po baterije koje rade paralelno, silazni pretvarač napona na 5V 3A zadužen za mikroupravljač i komponente. ESC će pružati izvor napajanja servo motoru od 6V i 1A. Sony DualShock 4 V2 bit će korišten za upravljanje svime putem Bluetooth veze. Ostale komponente koje će biti korištene su paralelni XT60 adapter za baterije i 14AWG žice proizvođača Team Corally koje podržavaju do 200A. Sve specifikacije i razlozi odabira biti će navedeni za svaku pojedinačnu komponentu te potencijalne alternative ako postoje.

### FireBeetle 2 ESP32-E

Cijeli diplomski rad se oslanja na ESP32-E mikroupravljaču koji preuzima funkciju prijamnika u automobilu na daljinsko upravljanje. U ovom slučaju, riječ je o posebnoj varijanti od proizvođača DFRobot. Za razliku od Arduino Nano ESP32 varijante, FireBeetle 2 ima manje flash memorije, podržava manji raspon ulaznog napona i ima stariju verziju procesora. Ako se u obzir uzme razlika u cijeni i mogućnost istovremene narudžbe drugih komponenti, ovi nedostatci nisu značajni. Na slici 1 prikazan je FireBeetle 2 ESP32-E mikroupravljač.

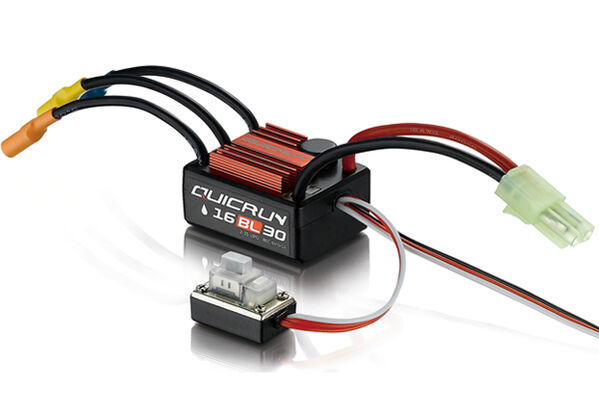


Slika - FireBeetle 2 ESP32-E mikroupravljač [1]

Glavne karakteristike zbog kojih je odabran su ugrađena podrška za Wi-Fi i Bluetooth, dvojezgreni procesor te kompaktan dizajn uz dovoljan broj I/O (engl. Input/Output) pinova. Kao što je prije spomenuto, Tensilica TX6 je dvojezgreni procesor koji je mozak ovog mikroupravljača. Radi na frekvenciji od 240MHz što je značajno više od mikroupravljača poput Arduino Uno čija je frekvencija rada 16MHz. Jedna jezgra zadužena je isključivo za rad s bežičnim vezama i komunikaciju dok je druga jezgra namijenjena za izvršavanje programskog koda koji se spremi u flash memoriju. Flash memorija ima 4MB i ovisno o shemi particije, različito je raspoređena. Prema osnovnim postavkama, 1.2MB namijenjeno je za programski kod, 1.5MB za datotečni sustav i preostalo je za OTA (engl. Over-The-Air) prenošenje programskog koda. Prva iteracija završenog programskog koda bez ikakve optimizacije stane na 1.2MB prostora uz nešto prostora koje ostane neiskorišteno. Ulogu radne memorije preuzima 520KB SRAM (engl. Static Random Access Memory). Ovdje će biti spremljeni svi privremeni podaci poput varijabli i njihovih vrijednosti. Podržani Wi-Fi standardi su 802.11 b/g/n/d/e/i/k/r, a maksimalna brzina prijenosa je do 150Mbps. Raspon frekvencija rada je od 2.4GHz do 2.5GHz, rad na 5GHz nije podržan iako podržava IEEE standard 802.11n. Po pitanju Bluetooth-a, podržana je v4.2 BR/EDR (engl. Basic Rate/Enchanced Data Rate) i BLE (engl. Bluetooth Low Energy). FireBeetle 2 ESP32-E ima 10 digitalnih I/O i 5 analognih pinova, te dva hardverska UART (engl. Universal Asynchronous Receiver/Transmitter) serijska sučelja od kojih je jedno u upotrebi kada je priključena USB (engl. Universal Serial Bus) konekcija. Uz UART, moguće je koristiti SPI (engl. Serial Peripheral Interface), I2C (engl. Inter-Integrated Circuit) i I2S (engl. Inter-Integrated Circuit Sound). Što se tiče napajanja, moguće je koristiti USB utor za napon od 4.75V do 5.5V, Li-ion sučelje za baterije napona od 3.5V do 4.2V te preko VCC pina s naponom 5V. Svi izvori napajanja moraju biti istosmjerne struje [1][2]. Sve zajedno, radi se moćnoj razvojnoj platformi koja više nego zadovoljava potrebe ovog diplomskog rada. Na tržištu postoji značajan broj mikroupravljača i većina bi mogla preuzeti ulogu FireBeetle 2 ESP32-E. Glavni uvjeti su 5 GPIO pinova koji će komunicirati s komponentama, napajanje od 5V i Bluetooth komunikacija koja ne mora nužno biti ugrađena na sam mikroupravljač. Na primjer, umjesto FireBeetle 2 ESP32-E bi se mogao koristiti Arduino Nano 33 IoT koji je isto baziran na ESP32 ili čak bilo koji drugi Arduino mikroupravljač uz dodatni, zasebni Bluetooth modul. Jedna prednost koju FireBeetle 2 ESP32-E ima nad ostalim mikroupravljačima je IO shield koji omogućava jednostavnije napajanje svih spojenih komponenti. Svaki podatkovni pin ima svoje zasebne pinove za napajanje i uzemljenje.

### Hobbywing Quicrun 16BL30 ESC

U normalnim okolnostima daljinski upravljač šalje naredbe prijamniku u automobilu i on upravlja ostalim komponentama, ali za napajanje i upravljanje DC motora potrebna je komponenta zvana ESC. ESC je direktno povezan na izvor električne energije i distribuira električnu energiju između prijamnika i DC motora te upravlja DC motorom prema naredbama koje dobiva od prijamnika. Za ovaj diplomski rad odabran je Hobbywing Quicrun 16BL30 ESC. Ovisno o izvoru koji se gleda, namijenjen je za automobile mjerila 1:12 ili manje [3]. Prema informacijama dostupnim u službenom priručniku, ovaj specifičan ESC bi trebao biti korišten u automobilu mjerila 1:16 ili manje. Iako se ne nalazi u idealnom okruženju, trebao bi funkcionirati bez poteškoća zbog načina na koji je napisan programski kod odgovoran za upravljanje DC motora. Podržava dvije vrste baterija s nizom konfiguracija baterija. Prva podržana vrsta baterija su Li-Po baterije od dvije do tri ćelije spojene u seriju. Druga podržana vrsta baterija su Ni-MH od četiri do devet ćelija. Konfiguracija baterija utječe na jačinu DC motor. U ovom diplomskom radu napajanje se sastoji od dvije 2S baterije u paraleli, napon je 7.4V i dostupno je maksimalno 220A istosmjerne struje. Hobbywing Quicrun 16BL30 ESC podržava kontinuiranu struju od 30A i moguću maksimalnu struju od 180A na kratak period [4]. Na slici 2 prikazan je Hobbywing Quicrun 16BL30 ESC koji se koristi u ovom diplomskom radu.



Slika - Hobbywing Quicrun 16BL30 ESC [3]

Postoji jedan mali lapsus koji se dogodio pri izradi ovog ESC-a. Iako ESC podržava kontinuiranih 30A, konektor Mini-Tamiya koji dolazi s ESC-om podržava maksimalnu kontinuiranu struju od 10A. Potrebno je zamijeniti Mini-Tamiya konektor za XT60 konektor koji podržava kontinuiranu struju od 30A. S korektnim adapterom zalemljenim, ESC se sada može ispravno napajati bez adaptera i ograničenja. No prije nego što se može koristiti, mora se konfigurirati raspon gasa. Taj proces se sastoji od slanja minimuma, neutralne pozicije i maksimuma raspona gasa. Očekivani raspon je od 1000 do 2000, a 1500 je neutralna pozicija. Ne mora se nužno koristiti kontroler za ovaj proces. Moguće je u programskom kodu dati naredbe i simulirati ulazne podatke kontrolera. Svaki put kad se pošalje pojedinačna naredba potrebno je pritisnuti gumb ispod sklopke za uključivanje kako bi se raspon spremio. Postoji niz alternativa ESC-a velikog broja proizvođača, no kako bi ga se zamijenilo mora se paziti na kompatibilnost s DC motorom, tj. vrsti motora i broju konektora te KV ocjeni. Odabrani DC motor, o kojem će biti više podataka u idućem poglavlju, je brushless vrsta motora, ima tri integrirana konektora i KV ocjenu od 4500. ESC mora minimalno podržavati ove specifikacije kako bi bio kompatibilan. Na primjer, TOROX 60 ESC proizvođača Team Corally podržava brushless DC motore s tri konektora i KV ocjenom manjom od 4000 ili između 4000 i 6000 ovisno o konfiguraciji baterija. No TOROX 60 ESC zauzima više prostora od Hobbywing Quicrun 16BL30 ESC-a te ne bi stao na isto mjesto koje zauzima trenutni ESC. Mijenjanjem ESC-a ne bi se postigla nikakva značajna promjena. Hobbywing Quicrun 16BL30 ESC je sposoban pogoniti odabrani DC motor bez ikakvih limita. Zamjenom DC motora s jačim modelom bi zahtijevalo i promjenu ESC-a te bi tek u tom slučaju bolji ESC imao smisla.

### Hobbywing Quicrun 2435 SL Brushless Motor G3

DC motor je direktno spojen na ESC i sve potrebno za rad dobiva od njega. Glavna i jedina uloga je DC motora je upravljati stražnjim kotačima automobila na daljinsko upravljanje. Odabrani ESC i konfiguracija baterija ograničava izbor modela DC motora. S obzirom da se radi o napajanju 2S baterijama, DC motor mora biti 2435 klase ili manji s KV ocjenom manjom od 8000. KV ocjena predstavlja brzinu okretaja motora za jedan volt napona. Manji KV se više koristi za situacije gdje je potreban okretni moment dok se viši KV koristi za veće brzine. Za svrhe ovog diplomskog rada koristi se Hobbywing Quicrun 2435 SL Brushless Motor G3 s KV ocjenom od 4500. Primarna namjena je za automobile mjerila 1:16 ili manje, no zbog omjera između zupčanika pogona koji idu od manjih prema većim zupčanicima, okretni moment je povećan što bi trebalo imati pozitivan utjecaj na kretanje automobila [5]. Na slici 3 je prikazan DC motor Hobbywing Quicrun 2435 SL Brushless Motor G3.



Slika - Hobbywing Quicrun 2435 G3 [5]

Za upravljanje DC motorom potrebno je ESC-u slati broj između 1000 i 2000. Ovisno o broju koji se pošalje, postoje tri stanja. Brojevi od 1000 do 1500 rezultiraju u kretanju unazad, točno 1500 je neutralno što znači da motor miruje i brojevi od 1500 do 2000 rezultiraju u kretanju unaprijed. Blizina rubnim vrijednostima 1000 i 2000 određuje koliko brzo će se motor okretati. Izbor DC motora je ograničen odabranim ESC-om. Osim što mora biti brushless DC motor s tri konektora, ima definiran raspon KV ocjene u koji mora spadati. Prema trenutnoj konfiguraciji, Hobbywing Quicrun 2435 SL Brushless Motor G3 je najbolja dostupna opcija. U slučaju drugačijeg ESC-a koji podržava jače motore, bolja opcija bi bio Hobbywing QuicRun Brushless Motor 3650 SD G2 u 17.5T i 21.5T izvedbi. Oni imaju manju KV ocjenu što znači manji broj okretaja u minuti, ali veći okretni moment. Njihova mana je u dužini motora jer ne bi stali u prostor koji zauzima Hobbywing Quicrun 2435 SL Brushless Motor G3.

### Tower Pro SG90 standard servo motor

Servo motori se koriste u okruženjima koja zahtijevaju mogućnost preciznog pozicioniranja. Otvaranjem kućišta servo motora vidljivo je nekoliko komponenti koje omogućuju njegov rad. DC motor se koristi za pokretanje mehanizma, a njime upravlja neka vrsta mikroupravljač ili neki drugi IC (engl. Integrated Circuit). Za praćenje pozicije upotrebljava se potenciometar koji smanjuje ili povećava svoju otpornost pri pomicanju. Servo motori trebaju ostvariti tri konekcije kako bi funkcionirali. Dvije konekcije su namijenjene za izvor napajanja i uzemljenje. Preostala konekcija je podatkovna veza između servo motora i mikroupravljača. Većina servo motora očekuje podatkovni signal svakih 20ms i širina signala određuje poziciju u koju se treba pomaknuti. Ova vrsta upravljanje se zove PWM (engl. Pulse Width Modulation). Postoje dvije izvedbe servo motora, a razlika je u opsegu kretanja. Prva izvedba je „closed-loop“ i odnosi se na servo motore s opsegom kretanja od 90° do 180°. Ovi servo motori pružaju povratnu informaciju mikroupravljaču o svojem položaju. Druga izvedba je „open-loop“ i predstavlja servo motore koji nisu ograničeni u opsegu kretanja. Mogu se slobodno kretati u 360°, no nije moguće precizno kontrolirati njihovu poziciju već samo brzinu i smjer. Servo motor će se u ovom diplomskom radu koristiti za upravljanje prednjim kotačima te je zbog toga potrebno odabrati „closed-loop“ servo motor [6]. Upravo za tu svrhu nabavljen je Tower Pro SG90 servo motor i vidljiv je na slici 4.



Slika - Tower Pro SG90 [autor]

Ovaj minijaturni servo motor je izrazito značajan. Njegove male dimenzije, težina od svega 9g i iznenađujuća snaga za veličinu čine ga idealnim sredstvom učenja za početnike. Iako je slabiji od servo motora koji se inače koriste za automobile na daljinsko upravljanje, SG90 je veoma impresivan sa svojim okretnim momentom od 2.5kg/cm u tako malom pakiranju. Može primati napajanja od 4.8V do 6V i zahtjeva do 1A električne energije. Napon napajanja ima utjecaja na fizičke karakteristike rada, tj. može promijeniti brzinu rada i okretni moment. Izlaznu osovinu će biti potrebno adaptirati na Lego križnu osovinu i dizajnirati kućište koje će servo motor držati fiksiranim na mjestu. Upravljanje SG90 servo motorom će biti korištenjem funkcija biblioteke ESP32Servo. SG90 pruža dovoljno okretnog momenta u malom pakiranju i zahtjeva veoma malo električne energije. Iako je dovoljan za ovu primjenu, više okretnog momenta ne bi naškodilo. Za razliku od ESC-a i DC motora, servo motor ne ovisi o drugim komponentama pri odabiru što pruža bolji izbor. Na primjer, Savöx SH-0254 servo motor pruža 3.9kg/cm okretnog momenta pri napajanju od 6V i sličnim fizičkim dimenzijama. Za razliku od SG90, treba više električne energije od 1200 do 1400 mA kada je u zakočenom stanju skretanja. To bi značilo da se uz Savöx SH-0254 treba smisliti drugačiji način napajanja, bilo to boljim pretvaračem napona s višim kapacitetom električne energije za mikroupravljača ili zasebnim napajanjem od 6V putem UBEC (engl. Universal Battery Elimination Circuit) uređaja.

### OVONIC 7.4V 2200mAh 50C Li-Po baterije

Postoji velik broj vrsta baterija s širokim spektrom karakteristika. Najčešći sastavi baterija su Pb-Ac (engl. Lead-Acid), Ni-Cd (engl. Nickel-Cadmium), Ni-MH (engl. Nickel-Metal Hybride), Li-Ion (engl. Lithium Ion) i Li-Po (engl. Lithium Polymer). Glavne karakteristike na koje treba obratiti pozornost su učinkovitost, životni ciklus i vijek, napon ćelija, kapacitet, no valja obratiti pozornost na toleranciju pri preopterećenje, rad na visokim temperaturama, otpornost na udarce i pražnjenje s vremenom. Marin-Garcia, Vazquez-Guzman, Sosa i Lopez [7] u svom radu proučavaju karakteristike čestih, komercijalno dostupnih baterija. Spomenuti rad sadrži odličnu tablicu koja pokriva ključne značajke baterija prema njihovom kemijskom sastavu. Na slici 5 je vidljiva spomenuta tablica.

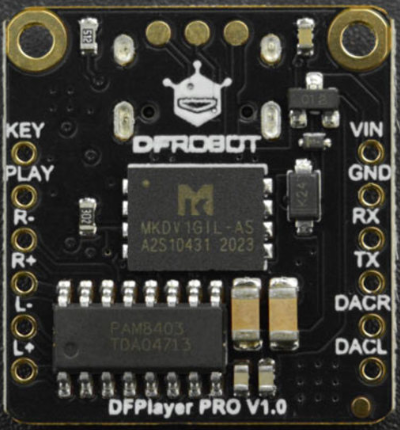


Slika - Tablica čestih sastava baterija i njihove karakteristike [7]

Već odabrani ESC podržava samo Ni-MH i Li-Po što ograničava izbor vrste baterija. Li-Po baterije imaju niz prednosti u odnosu na Ni-MH. Prva prednost je značajno smanjenje u težini i veličini pri istom kapacitetu. Druga prednost je bolja i veća izlazna snaga Li-Po baterija te ujednačeno pražnjenje tijekom korištenja. Treća prednost je mogućnost izrade u različitim veličinama što ih čini jako prilagodljivim. S druge strane, Ni-MH baterije su puno sigurnije od Li-Po baterija i jeftinije za proizvesti [8]. S prednostima i manama definiranim, za ovaj diplomski rad bit će korištene dvije OVONIC Li-Po 7.4V baterije. Prije punjenja Li-Po baterija treba se provjeriti napon pojedinačnih ćelija baterije koji ne smije biti ispod 3.3V po ćeliji. Pražnjenje Li-Po baterije može im naštetiti ako napon po ćeliji padne ispod 3V po ćeliji, tj. mogući rezultat je gubitak performansi ili napuhivanje. Li-Po baterije koje imaju vizualne deformacije poput napuhanosti ili fizička oštećenja poput curenja tekućine i oštećenja pakiranja se ne bi smjele puniti ni prazniti. U slučaju oštećenja bi ih se trebalo odnijeti u lokalno postrojenje za odlaganje opasnog otpada. Za punjenje i pražnjenje se uvijek mora koristiti punjač koji je napravljen specifično za Li-Po baterije ili podržava način punjenja potreban za Li-Po baterije uz ostale vrste baterija. U slučaju da se Li-Po baterije ne planira koristiti na duži period, potrebno ih je spremiti u suhu, relativno hladnu i tamnu okolinu s naponom ćelija u rasponu od 3.7V do 3.9V (otprilike 50% kapaciteta). Preporučeno je povremeno provjeriti stanje napona i prema potrebi ih puniti ili prazniti. Unatoč potencijalnim opasnostima koje mogu nastati pri neispravnom korištenju Li-Po baterija, veoma su bitne veličina i težina baterija. Konfiguracija baterija je 2S što znači da svaka baterija ima 2 ćelije spojene u seriju. Kapacitet baterija je 2200mAh s 50C brzinom pražnjenja. Brzina pražnjenja u amperima se računa tako što se uzme kapacitet, pretvori se iz mAh u Ah i pomnoži s brzinom pražnjenja od 50C. Rezultat je 110A po bateriji. Paralelnim radom dviju istih baterija udvostručava se kapacitet i maksimalna dostupna struja u sustavu. Još jedna prednost paralelnog rada je raspodjela tereta između dviju baterija što rezultira manjim pojedinačnim opterećenjem i manjom temperaturom pri radu. Baterije dolaze s XT60 priključkom za pražnjenje i JST-XH priključkom za punjenje i balansiranje napona između ćelija. Fizičke dimenzije ovih Ovonic Li-Po baterija bi mogle stvarati problem u dostupnom prostoru unutar kabine prototipa. Krivnja ne leži nužno na odabranim baterijama već na manjku prostora u kabini. Li-po baterije manjih dimenzija bi bile bolje, no time bi se narušio kapacitet i maksimalno vrijeme rada prototipa između punjenja. Glavne karakteristike na koje se treba obratiti pozornost su duljina baterija i konektor za pražnjenje. Trenutne baterije su duge 87mm i imaju XT60 konektor koji je kompatibilan s konektorom na ESC-u. Ovdje dolazi u igru prednost koju imaju Li-Po baterije nad ostalim vrstama, a to je mogućnost izrade u različitim veličinama. Konkretan primjer idealne Li-Po baterije bi bilo teško pronaći upravo zbog velikog izbora, no često znaju imati „Shorty“ u imenu. Dužina im je ispod 100mm, manjeg su kapaciteta i pražnjenje im je nešto sporije. Odabrane Ovonic Li-Po baterije spadaju u ovu kategoriju.

### DFPlayer Pro

Jedna od funkcionalnosti ovog diplomskog rada je simulacija zvukova motora automobila za što su potrebni zvučnici. Pošto se radi isključivo s digitalnim signalima, potrebno je imati uređaj imena DAC (engl. Digital to Analogue Converter) koji pretvara digitalne signale u analogne signale. Navedeni analogni signali zatim idu u uređaje koji ih mogu reproducirati, tj. zvučnike. Zbog limitacija u vremenu i sposobnosti, nije moguće napraviti pravi, u potpunosti responzivan zvučni sustav koji bi prema radnjama korisnika istovremeno kreirao odgovarajući zvuk motora. Zvukovi koji će biti proizvedeni su unaprijed snimljeni i dio programskog koda odgovoran za upravljanje zvukom je osmišljen da što je bolje moguće oponaša pravi responzivan zvučni sustav. ESP32 mikroupravljač dolazi s ugrađenim 8-bitnim DAC-om, no ne i s dovoljno memorije za zvučne zapise motora [1]. Upravo iz tog razloga odabran je zaseban MP3 (engl. MPEG Audio Layer 3) modul zvan DFPlayer Pro na kojem će se nalaziti sve potrebne datoteke. Na slici 6 prikazan je DFPlayer Pro modul.

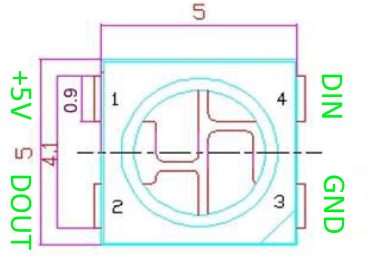


Slika - DFPlayer Pro [9]

DAC funkcionalnost ovog modula nudi 24-bitnu dubinu zvuka i prostor pohrane od 128MB. Zvučne zapise je moguće prenijeti s računala na modul putem USB-C konekcije. Podržava dva kanala reprodukcije zvuka i niz čestih formata poput MP3, WAV (engl. Waveform Audio File Format), WMA (engl. Windows Media Audio), FLAC (engl. Free Lossless Audio Codec), AAC (engl. Advanced Audio Coding) i APE (engl. Monkey's Audio). DFPlayer Pro komunicira s mikroupravljačem putem hardverske UART serijske veze. Za napajanje je potreban izvor napona u rasponu od 3.3V do 5V, a za ne ometani rad 20mA ili više. Ima četiri podržana načina rada od kojih će u upotrebi biti Arduino način. Ostali načini rada su AT naredbe, tipke prisutne na modulu, i ADKEY pina na modulu za koji je potrebno nabaviti dodatnu komponentu ili samostalno rekreirati dodatnu komponentu pomoću niza otpornika specifičnih vrijednosti. Za razliku od DFPlayer Mini MP3 modula, nudi opciju kretanja naprijed i nazad u zvučnim zapisima što je veoma bitno za zvukove motora kada se dogodi nagli prekid dok se zvučni zapis još reproducira. Na pinove označene s R+/- i L+/- potrebno je povezati zvučnike koji će reproducirati zvukove [9]. U odnosu na ostale komponente, DFPlayer Pro je najbolji MP3 modul koji je u trenutku nabave bio dostupan. DFPlayer Mini varijanta gubi podršku za jedan od kanala reprodukcije, treba zasebnu memorijsku karticu za pohranu zvučnih zapisa, podržava manji broj zvučnih formata, tri načina upravljanja i nema mogućnost kopiranja zvučnih zapisa direktno na modul.

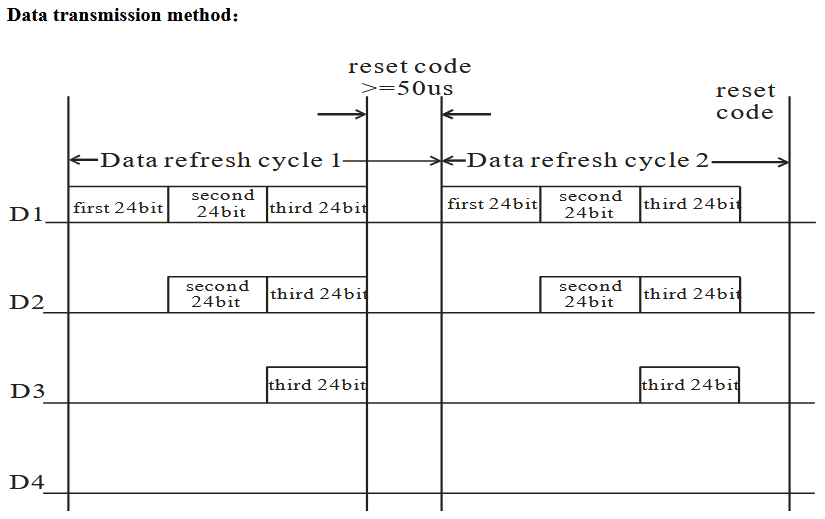
### WS2812B RGB LED diode

Klasične diode imaju četiri kontakta, tri namijenjene za crvenu, zelenu i plavu boju te jedna za uzemljenje (engl. Cathode) ili napajanje (engl. Anode), ovisno o izvedbi. No one zauzimaju više mjesta nego što je dostupno u ovom diplomskom radu. Upravo zbog toga odabrana je varijanta WS2812 RGB LED diode, tj. WS2812B s manjim brojem potrebnih pinova. Na slici 7 je prikazan dijagram jedne WS2812B RGB LED diode.



Slika - Dijagram WS2812B RGB LED diode [10]

Veličina ovih dioda je 5x5x2mm (LxWxH) što je puno prihvatljivije od običnih dioda. Slično kao i obične diode, moguće je pojedinačno podesiti jačinu crvene, zelene i plave boje. Sveukupno može pokriti 16777216 (256^3) kombinacija boja. Zahtjeva napajanje u rasponu od 3.5V do 5V. Potrebne su četiri diode za ovaj diplomski rad, dvije na prednjoj strani i dvije na stražnjoj strani automobila. Za upravljanje svim diodama potrebna je samo jedna konekcija što je moguće zbog načina rada ovih dioda. Svaka dioda očekuje paket od 24 bita, po 8 bitova za svaku primarnu boju. Spajanjem DOUT konekcije jedne diode na DIN konekciju druge diode rezultira vezom koja se zove lančano povezivanje (engl. Daisy Chain). Kako bi signal bio čitljiv i na kraju lanca, svaka dioda regenerira signal i prosljeđuje ga dalje.



Slika - Prijenos podataka između dioda [10]

Na slici 8 prikazan je način prijenosa podataka između dioda. U primjeru su prisutne četiri diode, no koriste se samo tri. Vidljiva su dva ciklusa osvježavanja podataka na diodama i segment označen kao „reset code“. Diode same znaju za koju od njih je namijenjen podatak prema vremenu slanja u jednom ciklusu. Vrijeme jednog ciklusa se mijenja s obzirom na broj dioda u jednom nizu [10]. Glavne prednosti WS2812B dioda je izvanredna kompaktnost i mogućnost upravljanja više dioda putem jedne podatkovne konekcije. Problem ovih dioda se krije u tehnologiji lemljenja. Riječ je o SMD (engl. Surface Mounted Device) tehnologiji. One nemaju klasične konektore koji bi prolazili kroz rupe već se leme na površinu. Kontakti dioda leže na kontaktima te ih je teško lemiti klasičnom lemilicom. Unatoč tom problemu, nema zamjene koja bi bila bolja opcija.

### Sony DualShock 4

DualShock 4 je četvrta iteracija kontrolera za Sony-eve PlayStation konzole. Moguće ga je dobiti zajedno s PlayStation 4 konzolom ili kupiti zasebno, a dostupan je široj javnosti od 2013. godine. Ima dvije mogućnosti povezivanja, jedna je putem microUSB konektora dok je druga putem Bluetooth v2.1 veze koja će biti korištena u ovom diplomskom radu. Kontroler ima šesnaest digitalnih gumbi, dva analogna okidača, dva analogna joystick-a, kapacitivni touchpad s podrškom za dva istovremena dodira i tri senzora za pozicioniranje u prostoru. Kontroler sadrži Li-ion bateriju napona 3.7V i 1000mAh što omogućava bežični rad. Na slici 9 prikazan je DualShock 4 kontroler.



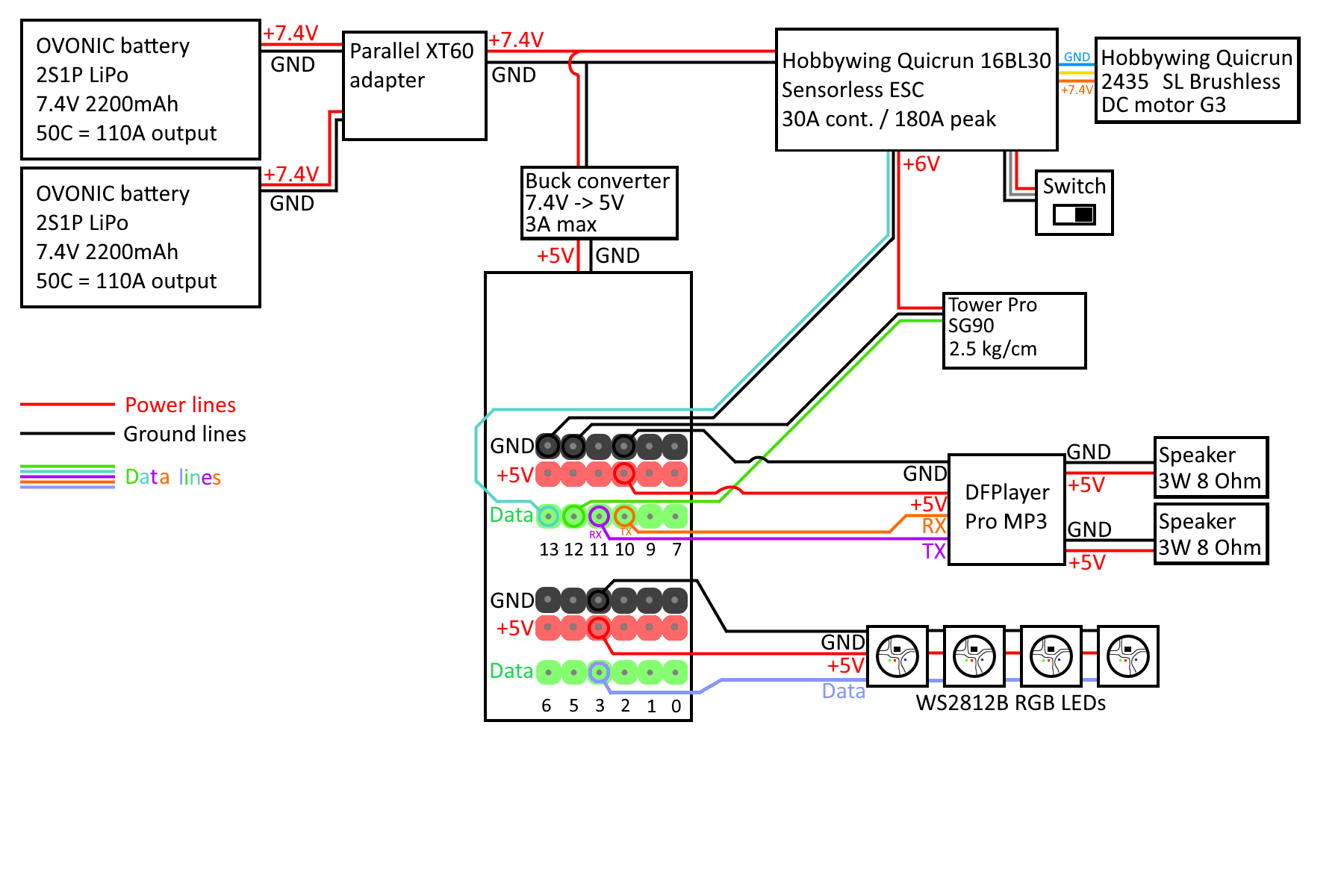
Slika - DualShock 4 kontroler [11]

Za planiranu kontrolnu shemu potrebni su lijevi joystick, gumb sa simbolom za PlayStation, sva četiri gumba sa simbolima geometrijskih likova, oba gumba i okidača na gornjoj strani kontrolera. Zbog iznimno jednostavne implementacije odabran je Sony Dualshock 4 kontroler, ali to ne znači da je ujedno i najbolja opcija. Ulogu daljinskog upravljača bi mogao preuzeti bilo koji drugi uređaj koji podržava Bluetooth tehnologiju uz nužne promjene u programskom kodu. Idealna opcija bi bio bilo koji relativno moderan pametni telefon kako bi se poboljšala kompatibilnost i adaptabilnost. Oni standardno podržavaju Bluetooth tehnologiju, no u tom slučaju bi trebalo kreirati mobilnu aplikaciju sa svim potrebnim ulaznim metodama.

## Logička shema fizičkih konekcija

Sve odabrane komponente je potrebno na neki način spojiti kako bi ispravno funkcionirale. Kao osnovni korak, potrebno im je dati napajanje i uzemljenje prilagođeno pojedinoj komponenti. Odabrane baterije u paralelnoj konfiguraciji nude sveukupno 220A električne energije pri naponu od 7.4V. Dio komponenti treba napon od 5V koji se može dobiti na IO shield-u mikroupravljača. On je spojen na baterije preko silaznog pretvarača koji napon od 7.4V spušta na prihvatljivih 5V. S maksimalnom strujom od 3A, trebalo bi biti više nego dovoljno za sam mikroupravljač, DFPlayer Pro MP3 modul, dva zvučnika i četiri WS2812B diode lančano povezane jedna za drugu. Drugi dio komponenti treba napon od 7.4V koji se može dobiti direktno s baterija. ESC treba imati 30A kontinuirane električne energije s maksimalnim, kratkotrajnim skokom na 180A. DC motor nema specificirani maksimum u dokumentaciji, no navodi se minimum od 2.1A za 4500KV varijantu. Servo motor će koristiti ESC kao izvor napajanja koji nudi 6V i 1A što bi trebalo biti dovoljno. Iako podržava napajanja od 5V, postoji mogućnost da će preopteretiti silazni pretvarač u kombinaciji s drugim komponentama te zbog toga se napaja odvojeno. S napajanjem gotovim, pozornost se može posvetiti ostalim konekcijama koje je potrebno ostvariti. Servo motor treba podatkovnu liniju za komunikaciju s ESP32 mikroupravljačem. ESC treba podatkovnu liniju i povezano uzemljenje za komunikaciju i ispravno funkcioniranje. DC motor ima tri veze koje treba povezati s ESC-om. DFPlayer Pro MP3 modul treba dvije podatkovne veze za serijsku komunikaciju preko RX i TX priključaka koju je moguće ostvariti bez softverske emulacije serijskog sučelja. Preostale su još WS2812B LED diode kojih ima četiri, no potrebna im je samo jedna podatkovna veza na prvu diodu. WS2812B LED diode su sekvencijalno povezane jedna na drugu uz tehniku koja se često može vidjeti pod engleskim nazivom „daisy chain“ ili prevedeno na hrvatski kao lančana veza. Ovaj način povezivanja se ne odnosi samo na podatkovnu vezu, već i na napajanje i uzemljenje. U nastavku diplomskog rada se nalazi slika 10 koja demonstrira logičku shemu svih potrebnih fizičkih konekcija.

Slika - Logička shema fizičkih konekcija [autor]



Crvena boja predstavlja naponske veze bez obzira na napon, crna boja predstavlja uzemljenje, sve ostale boje predstavljaju podatkovne linije. Za podatkovne linije je korišten spektar boja kako ne bi došlo do zabune gdje koja linija ide. Fizičke konekcije su ostvarene na pomalo nelogičnim mjestima, tj. priključne točke nisu korištene nekim redoslijedom već su rezultat razvoja u hodu i bez upotrebe IO shield-a. Promjene je moguće napraviti uz nužne izmjene u programskom kodu u svrhu bolje fizičke organizacije, no trenutan raspored konekcija ne utječe na funkcionalnost prototipa. Fizički raspored komponenti u samom modelu automobila bit će pojašnjen u idućem poglavlju nakon što se opišu određene modifikacije napravljene na LEGO Technic modelu.

## Modifikacija LEGO Technic modela

Logička shema služi isključivo za jednostavniji pregled konekcija bez obzira o stvarnom rasporedu komponenti dok je za modifikacije Lego Technic modela potrebno imati fizički raspored. Odabrani Lego Technic model je Ford GT 2022. Riječ je od modelu mjerila 1:12 u odnosu na stvarni automobil pušten u prodaju 2023. godine. Dimenzije modela su 39x18x9 cm (LxWxH) i teži 1.7kg bez ikakvih dodatnih komponenti. Na slici 11 je prikazan spomenuti Lego Technic set.



Slika - Ford GT 2022 Lego Technic set [12]

Unutrašnjost kabine je relativno čista, no za ugradnju svih komponenti potrebno je napraviti još mjesta. Pošto je model već sastavljen od prije, oprezno se treba rastavljati u grupiranim dijelovima. Iz unutrašnjosti bez većih problema se mogu izvaditi sjedala, no mehanizam za podizanje stražnjeg aerodinamičnog krila je bio među prvim koracima sastavljanja modela. Potrebno je skinuti gotovo cijeli vanjski sloj dijelova koji čine karoseriju modela, zatim odvojiti bočne stranice koje spajaju prednji i stražnji dio te odvojiti prednji i stražnji dio. Uz navedeno treba ukloniti sustav za upravljanje prednjim kotačima kako bi se mogao zamijeniti sa servo motorom. Dok je model još u rastavljenom stanju, odmah se može otkloniti i model V6 motora te ga zamijeniti s DC motorom jer taj prostor kasnije neće biti dostupan. Na slici 12 je prikazan modificirani model. S lijeve strane se nalaze nepotrebni dijelovi, s desne strane se nalaze dijelovi za krov koji će biti vraćeni na model.



Slika - Modificirani model [autor]

Nakon uklanjanja svih dijelova koji smetaju i ugradnje DC motora, model je spreman za ostale komponente. No prije nego što se to može napraviti, moraju se dizajnirati adapteri za servo i DC motor kako bi odgovarali križnim Lego osovinama. Na slici 12 su servo i DC motor već ugrađeni jer je slika nastala tek nakon procesa dizajna adaptera i ugradnje. Preostalo je ugraditi još nekoliko komponenti, a one su ESC, dva zvučnika, MP3 modul, pretvarač napona, baterije i mikroupavljač. Iznad DC motora ima dovoljno mjesta za ugradnju ESC-a. Pošto je u kontaktu s krovom automobila i povezan s baterijama te DC motorom, nema straha da će se micati. Baterije će ići u kabinu modela jer nigdje drugdje nema mjesta. Njihovi priključci će nažalost morati izlaziti kroz okvir prozora kako bi mogli doći do ESC-a i pretvarača napona. Pretvarač napona i mikroupravljač će se nalaziti s vanjske strane krova modela u posebno dizajniranom dijelu koji će ih držati na mjestu. Estetski bi bilo bolje kada bi ih se moglo staviti u kabinu modela, no zbog veličine pretvarača napona to nije moguće. Dodatna prednost je lakši pristup mikroupravljaču. Prije ugradnje komponenti treba se napraviti adapter koji će spajati baterije s ESC-om i pretvaračem napona. Adapter se sastoji od dva XT60 konektora i dvije 14AWG žice koje idu prema pretvaraču napona. Preostalo je pozicionirati zvučnike i MP3 modul. MP3 modul će se zbog svojih malih dimenzija pozicionirati unutar kabine modela. S donje strane MP3 modula će biti izolacijska traka kako slučajno ne bi došlo do kratkih spojeva u kontaktu s drugim komponentama. Zvučnici su puno većih dimenzija od MP3 modula. Jedan od zvučnika će biti smješten ispred prednjih kotača ispod haube modela. Uz manju modifikaciju, moguće je osigurati dovoljno mjesta da stane bez problema. S donje strane modela direktno ispod motora se nalazi nešto mjesta gdje bi drugi zvučnik mogao stati. Ta lokacija je idealna s jednom velikom manom, a to je gravitacija. Ništa ne drži zvučnik na mjestu i bilo bi potrebno dizajnirati adapter koji bi fiksirati zvučnik. Micanjem pretvarača napona i mikroupravljača na krov oslobađa dovoljno prostora u kabini za drugi zvučnik. S ovim korakom je završena hardverska implementacija diplomskog rada te se prelazi na programski dio.

# Programski aspekt

Kroz ovo poglavlje proći će se programska strana diplomskog rada koja se primarno odnosi na programski kod za ESP32 mikroupravljač. Upravljanje automobilom pomoću DualShock 4 kontrolera je glavna funkcionalnost koju je potrebno implementirati. Dodatne funkcije koje će biti dodane su prednja svjetla, reaktivna stražnja svjetla koja se uključuju pri kočenju, indikatori smjera i svjetla za opasnost te zvukovi motora povezani s trenutnim stanjem u kojem se automobil nalazi. Za korištenje odabranih komponenti potrebno je koristiti određene biblioteke koje to omogućuju na jednostavniji način. Programski kod je pisan u Arduino IDE v2.3.4 okruženju jer nudi pregledno sučelje, menadžer biblioteka i širok spektar podržanih mikroupravljača. Fokus je na jednostavnost implementacije i responzivnost samog automobila umjesto na optimiziranosti programskog koda. Osnovna struktura programskog koda je uključivanje potrebnih biblioteka, deklariranje varijabli za pohranu bitnih podataka i stanja, početna inicijalizacija komponenti, sve funkcije zadužene za funkcioniranje automobila te na kraju glavna funkcija koja kontinuirano poziva funkcije za kretanje, zvukove i svjetla. No prije nego što se programski kod može pisati i testirati, neophodno je pronaći biblioteke koje će se koristiti kako bi se olakšao proces programiranja.

## Korištene biblioteke

Biblioteke su kolekcije programskog koda namijenjene za višekratnu upotrebu. Služe kao pomoćni moduli koji nude funkcije i klase pri izradi vlastitog programskog koda. Isti koncept se koristi i u programiranju mikroupravljača, a koriste se u svrhe upravljanja neke vrste komponente. Za ispravan rad odabranih komponenti, potreban je niz biblioteka koje će olakšati komunikaciju i upravljanje nad njima. Prva na redu je „PS4Controller“ biblioteka koja je odgovorna za ispravnu komunikaciju sa Sony-evim kontrolerom DualShock 4. Omogućuje kontroleru da šalje informacije o statusu svih joystick-a, gumbiju, okidača i ugrađenih senzora kako bi mikroupravljač mogao reagirati na temelju tih statusa. Druga grupa biblioteka je zadužena za bluetooth funkcionalnosti ESP32 mikroupravljača. Iako se mogu koristiti direktno u programskom kodu, u ovom radu ih je dovoljno samo uključiti. Treća biblioteka je „FastLED“ i ona je zadužena za upravljanje WS2812B RGB LED diodama. Iz nje su potrebne samo četiri funkcije. Spomenute funkcije su deklariranje ulančanih WS2812B RGB LED dioda, postavljanje svjetline dioda, postavljanje boje dioda i ažuriranje stanja dioda. Četvrta biblioteka je „ESP32Servo“. Ova biblioteka je varijanta „Servo“ biblioteke namijenjena specifično za ESP32 mikroupravljače jer nisu sve biblioteke kompatibilne sa svim mikroupravljačima. Ona omogućava upravljanje servo i DC motorima pomoću nekoliko funkcija. Prvi korak je deklaracija motora, zatim određivanje priključnog pina i načina komunikacije. Nakon tih koraka, motori su spremni za rad. Peta i šesta biblioteka su namijenjene za upravljanje DFPlayer Pro MP3 modulom. Biblioteka „DFRobot\_DF1201S“ sadrži naredbe za upravljanje modulom dok „HardwareSerial“ kreira hardversku serijsku vezu između mikroupravljača i modula preko koje se komunikacija odvija.

## Implementacija funkcija automobila

Sa svim bibliotekama odabranim i definiranim, važno je smisliti što će i kako će sve biti implementirano te način na koji će korisnik moći upravljati prototipom. Sveukupan programski kod se sastoji od otprilike 580 linija, a u njemu se nalazi sve potrebno za upravljanje cijelim prototipom. Uz devet biblioteka i niza varijabli, napisano je petnaest funkcija s dodatnih dvije funkcije dostupne u primjerima „PS4Controller“ biblioteke. Fokus nije bio na optimizaciji programskog koda već na općenitoj implementaciji funkcionalnosti. Na slici 13 je vidljiva cijela kontrolna shema, tj. koja funkcija je povezana s kojom metodom unosa podataka.



Slika - Legenda funkcija mapiranih na Sony DualShock 4 kontroleru [autor]

Kao što je vidljivo na slici, za upravljanje kretanjem automobila koriste se lijevi i desni analogni okidač te lijevi analogni joystick. Sve ostale funkcije povezane su na digitalne gumbe i imaju samo uključeno/isključeno stanje. Za uspostavljanje veze između DualShock 4 kontrolera i ESP32 mikroupravljača zadužen je gumb između analognih joystick-a. Kada kontroler emitira konzistentno svjetlo na gornjoj strani, znači da je povezivanje bilo uspješno. Što se tiče programskog dijela, preostalo je još samo detaljno proći kroz implementirane funkcije.

## Detaljan pregled implementiranih funkcija

Kroz ovo poglavlje će se proći sve funkcije i što one pojedinačno rade te kako zajedno čine cjelinu. Korištene varijable će po potrebi biti pojašnjene dok će isječci programskog koda funkcija biti djelomično ili u potpunosti vidljive. Sve kreće od „setup()“ funkcije koja je iznimno bitna. U njoj se sustav priprema na rad. Ovdje se definiraju diode kojima upravljaju funkcije biblioteke „FastLED“, započinju se serijske veze, podešava se način rada DFPlayer Pro MP3 modula, određuje se koje funkcije će biti pozvane pri događajima povezanim s DualShock 4 kontrolerom te na kojim pinovima se nalaze servo i DC motor. U nastavku je tekstualni okvir koji sadrži sve važne linije programskog koda iz funkcije „setup()“.

void setup(){

  FastLED.addLeds<WS2812, LED\_PIN, GRB>(leds, NUM\_LEDS); //Deklariranje dioda

  FastLED.setBrightness(96); //Podešavanje svjetline dioda

  brakeReg(); //Priprema stražnjih svjetala

  Serial.begin(115200); //Serijska veza s PC-om

  DFP.begin(115200, SERIAL\_8N1, D11, D10); //Serijska veza s MP3 modulom

  DF.setVol(0);

  DF.switchFunction(DF.MUSIC); //Stavljanje MP3 modula u MUSIC način rada

  delay(2000);

  DF.setPlayMode(DF.SINGLE); //Postavka da MP3 pusti jednu datoteku i stane

  DF.setVol(15); //Podešavanje glasnoće zvučnika

  DF.playFileNum(1); //Naredba da počne puštati prvu datoteku

  mp3CurFile = millis(); //Spremanje trenutnog vremena u varijablu

  PS4.attach(stateCheck); //Povezivanje funkcija s

  PS4.attachOnConnect(onConnect); //događajima vezani za

  PS4.attachOnDisconnect(onDisConnect); //DualShock 4 kontroler

  PS4.begin();

  steering.attach(D12); //Servo motor na pinu D12

  motor.attach(D13, 1000, 2000); //DC motor na pinu D13 i treba milisec

  steering.write(90); //Izravnjava prednje kotače

}

Tekstualni okvir - Funkcija setup()

Iduće na redu su funkcije preuzete iz primjera vezanih za biblioteku „PS4Controller“. Prve dvije služe samo kao ispis statusa, tj. da li je kontroler uspješno ostvario konekciju s mikroupravljačem. Njihova jedina svrha je da preko serijskog sučelja pošalju poruku računalu o statusu kontrolera, tj. povezanosti.

void onConnect(){

  Serial.println("Connected!");

}

void onDisConnect(){

  Serial.println("Disconnected!");

}

Tekstualni okvir - Funkcije onConnect() i onDisconnect()

Nakon toga slijedi veoma bitna funkcija zvana „steer()“ zadužena za upravljanje servo motorom. Sastoji se od sedam linija. Prva linija čita stanje X-osi lijevog joystick-a koje može biti između -127 i 127. U drugoj liniji se uzima ta vrijednost i mapira se na raspon prirodnih brojeva od 0 do 165. Inicijalni plan je bio mapirati na raspon između 60 i 120, no zbog manjih nedostataka servo motora potrebno je dopustiti servo motoru da se općenito okreće više u oba Nakon što je vrijednost mapirana, vrijeme je da se pošalje impuls servo motoru i dogodi pomicanje kotača. Iako nije obavezna, dodana je provjera koja gleda koji je rezultat mapirane vrijednosti i ako je između 85 i 95, šalje impuls za centriranje kotača. Kontroleri znaju ponekad imati fenomen poznat kao „stick drift“, a ukratko se odnosi na situaciju gdje se joystick ne može u potpunosti centrirati te konstantno šalje impuls različit od nula. U slučaju da rezultat nakon mapiranja nije u rasponu od 85 do 95, pretpostavlja se da korisnik daje pravi impuls i šalje se servo motoru.

void steer(){

  LstickX = PS4.LStickX();

  mappedLstickX = map(LstickX, -127, 127, 30, 165);

  if(mappedLstickX >= 85 && mappedLstickX <= 95){

    steering.write(90);

  }else{

    steering.write(mappedLstickX);

  }

}

Tekstualni okvir - Funkcija steer()

Između funkcija „steer()“ i „drive()“ se nalaze dvije funkcije koje „drive()“ poziva pri svom izvršavanju. Kako bi se njih pojasnilo, prvo se mora proći kroz „drive()“ funkciju. Ona je zadužena za stražnje kotače i upravlja DC motorom. Iako se poziva nekoliko puta u sekundi, izvršava se svakih 25 milisekundi kako bi se DC motoru dopustilo malo vremena za prilagodbu. Kada se napokon krene izvršavati, treba provjeriti nekoliko stanja koje mijenjanju određene aspekte. Provjeravaju se četiri stvari. Prvo se provjerava da li je gumb zadužen za kočnicu pritisnut i ako je, stavlja varijablu „speedCurrent“ na vrijednost nula te ne dopušta primanje ulaznih vrijednosti putem lijevog i desnog okidača. Ako kočnica nije aktivna, dalje se provjerava koji od okidača je pritisnut te se vrijednost čita od onog koji je pritisnut. Ako nijedan nije pritisnut, pretpostavlja se da nema ulaznog podataka i čita se vrijednost desnog okidača iako realno nije bitno jer će rezultat biti nula. Idući korak poziva funkciju „accelspeed()“ koja je zadužena za ubrzanje automobila. Ona podešava varijablu „accspeed“ na način da je poveća ili smanji ovisno o vrijednosti desnog okidača. Početni plan gdje je brzina DC motora mogla biti između 0 i 255 je dopušta vrijednosti akceleracije koje su jedan, dva ili četiri, no radi očuvanja je sustav ograničen na raspon od 0 do 144 što znači da vrijednost može biti jedan ili dva. Nakon što je brzina akceleracije ispravna, radi se još jedna provjera. Ako je pritisnut gumb sa simbolom kruga na kontroleru, to označava da je lažno kvačilo aktivno te motor neće zapravo primati impuls za kretanje već će s vremenom usporavati i na kraju stati. Ako lažno kvačilo nije pritisnuto, funkcija radi normalno i impuls se šalje. U oba slučaja se poziva funkcija „accdeacc()“ koja je zadužena za rad s dvije varijable: „speedCurrent“ i „speedCatchup“. Kao što je prije spomenuto, „speedCurrent“ predstavlja vrijednost koju trenutno daje kontroler dok „speedCatchup“ kroz vrijeme raste ili pada, a cilj joj je ujednačiti se da vrijednost u varijabli „speedCurrent“. Funkcija „accdeacc()“ samo povećava ili smanjuje vrijednost „speedCatchup“ varijable prema vrijednosti u „speedCurrent“. Ako su jednake, ništa se događa. Ako je „speedCurrent“ veća od „speedCatchup“, onda se „speedCatchup“ povećava za vrijednost u varijabli „accspeed“. Ako je „speedCurrent“ manja od „speedCatchup“, onda se „speedCatchup“ smanjuje za vrijednost u varijabli „deaccspeed“. Varijabla „deaccspeed“ se mijenja isključivo kada je pritisnuta kočnica i može imati vrijednosti jedan ili četiri ovisno o stanju kočnice. Kada funkcija „accdeacc()“ završi, ide mapiranje vrijednosti. Vrijednost varijable „speedCatchup“ se mapira na raspon vrijednosti od 1000 do 2000 što predstavlja broj mikrosekundi koji će se poslati ESC-u. Nakon slanja mapirane vrijednosti, izvršava se mali komad programskog koda koji svakih deset ciklusa funkcije „drive()“ ažurira varijablu „prevSpeed“. Ta varijabla je bitna za iduću funkciju „sound()“. Na samom kraju se sprema vrijeme zadnjeg ulaska u funkciju i nastavlja se dalje s radom.

void drive(){

if (millis() - lastTimeStamp > 25){

    if(PS4.Cross()){ speedCurrent = 0;

    }else{

      if(PS4.R2() && speedCatchup >= 0){

        speedCurrent = PS4.R2Value();

      }else if(PS4.L2() && speedCatchup <= 0){

        speedCurrent = PS4.L2Value()\*(-1);

      }else if(!PS4.R2() || !PS4.L2()){

        speedCurrent = PS4.R2Value();

      }

    }

    accelspeed();

    if (!ci){ //Ovisi o lažnom kvačilu

      speedCatchup = accdeacc(speedCurrent, speedCatchup);

      actualSpeed = map(speedCatchup, -255, 255, 1000, 2000);

    }else{

      speedCatchup = accdeacc(speedCurrent, speedCatchup);

      speedCaCircle = accdeacc(speedCuCircle, speedCaCircle);

      actualSpeed = map(speedCaCircle, -255, 255, 1000, 2000);

    }

    motor.writeMicroseconds(actualSpeed);

    cycleTrack++;

    if(cycleTrack >= 9){

      cycleTrack = 0;

      prevSpeed = speedCatchup;

    }

    lastTimeStamp = millis();

}

}

Tekstualni okvir - Funkcija drive()

Funkcija „sound()“ je nesumnjivo najveća funkcija ovog programskog koda sa svojih 166 linija. Uz to je dosta kompleksna zbog implementirane logike. Prvi korak je provjeriti u kojem je stanju funkcija. Ako se nedavno izvršila, postoji mogućnost da se odabrani zvuk i dalje izvodi. Postoje dva načina na koje se može ući u funkciju. Prvi način je kada trenutni zvuk bude pri kraju izvođenja i vrijeme je da se pokrene novi. Drugi način je kada dođe do neke iznenadne promjene poput puštanja desnog okidača što znači da automobil počinje usporavati. Ovdje spada niz uvjeta koji se provjerava. Iduća provjera određuje način na koji se ulazak dogodio. Ovdje se gledaju stanja okidača, vrijednost varijable „prevSpeed“ i sl. Za svaki način i kombinaciju, varijabli „triggered“ se dodjeljuje jedinstvena vrijednost koja se koristi u idućem korak. Ako je ulazak prirodan, varijabla „triggered“ je nula i ne utječe na izvršavanje funkcije. Ako nije nula, pokreće se izvođenje specifičnog zvuka ovisno o stanju automobila i praćenih varijabli. Zbog količine programskog koda, u tekstualnom okviru se nalazi samo provjera koja mijenja vrijednost varijable „triggered“. Na kraju funkcije „sound()“ prati se stanje u kojem se nalaze lijevi i desni okidač kako bi se pri idućem iznenadnom ulasku u funkciju mogao znati uzrok.

if(PS4.R2() && trackR2 == 0 && prevSpeed == 0){

Serial.printf("Sudden acceleration from idle!");

triggered = 3;

}else if(PS4.R2() && trackR2 == 0){

Serial.printf("Sudden acceleration!");

if(shiftComplete == 0){

triggered = 4;

}else{

triggered = 5;

}

}else if(!PS4.R2() && trackR2 == 1){

Serial.printf("Sudden deacceleration!");

if(shiftComplete == 1){

triggered = 6;

}else{

triggered = 7;

}

}else if(PS4.L2() && trackL2 == 0 && prevSpeed == 0){

Serial.printf("Sudden reverse acceleration from idle!");

triggered = 8;

}else if(PS4.L2() && trackL2 == 0){

Serial.printf("Sudden reverse acceleration!");

triggered = 9;

}else if(!PS4.L2() && trackL2 == 1){

Serial.printf("Sudden reverse deacceleration!");

triggered = 10;

}else{

Serial.printf("Normal entry!");

triggered = 0;

}

Tekstualni okvir - Dio funkcije sound() zadužen za određivanja uzroka ulaska u funkciju

Nakon funkcije „sound()“ se nalazi niz manjih funkcija koje su zadužene za promjenu svjetala. Napisane su kao funkcije kako se isti kod ne bi morao ponavljati na više mjesta nego da se samo pozove potrebna funkcija koja će obaviti traženi rezultat. Veoma su jednostavne. Neke imaju for petlje koje mijenjaju vrijednosti svih dioda za crvenu, zelenu i plavu boju dok druge rade samo sa specifičnim svjetlima.

void head(){

  for(int i = 0; i < 2; i++){

    leds[i] = CRGB(frontHead[0], frontHead[1], frontHead[2]);

  }

}

void rLane(){

  leds[1] = CRGB(hazard[0], hazard[1], hazard[2]);

  leds[2] = CRGB(hazard[0], hazard[1], hazard[2]);

}

Tekstualni okvir - Primjer funkcija head() koja radi sa svim diodama i rLane() koja radi s diodama desne strane automobila

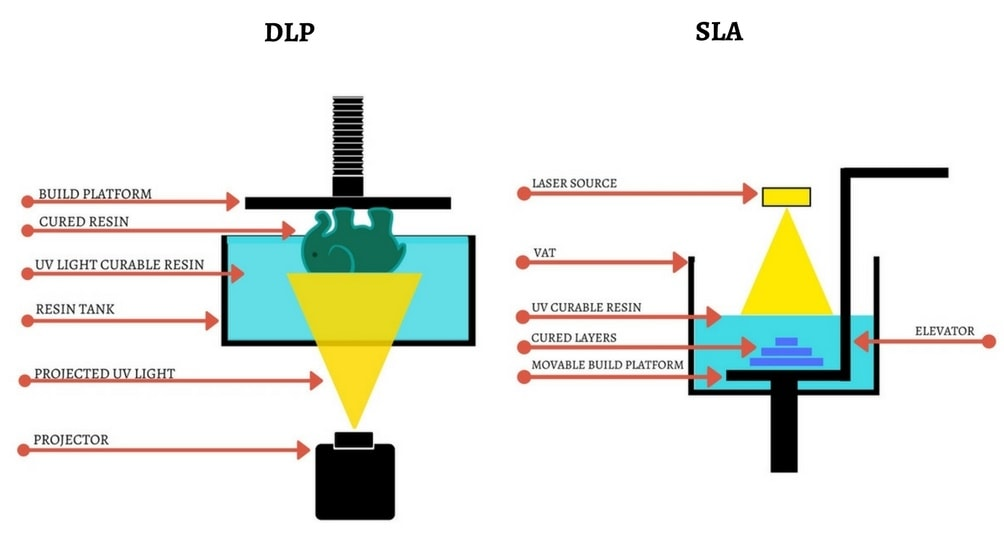
Iduća funkcija je „lights()“ i ona zapravo zove spomenute manje funkcije za manipulaciju nad svjetlima automobila. Nju je moguće izvršiti svakih 500 milisekundi kako ne bi se minimizirali slučajevi brzog uključivanja i isključivanja svjetala. Funkcija sadrži nešto logike za upravljanje svjetlima u slučaju svjetla za opasnost ili promjenu smjera. Iduće dvije funkcije rade zajedno. Prva je „stateCheck()“, a druga je „stateChange()“. Prva funkcija se pozove svaki put kada kontroler primijeti promjenu u stanju nekog gumba te poziva drugu funkciju koja sadrži logiku da napravi izmjene temeljene na promjeni stanja gumba. Ovdje se događa provjera i prilagodba koda za lažno kvačilo, promjena stanja stražnjih svjetala i nešto logike zadužene za indikatore promjene trake. Na samom kraju dokumenta je funkcija „loop()“ čiji je zadatak pozivati četiri glavne funkcije dokle god mikroupravljač ima električne energije. Te četiri funkcije su „steer()“, „drive()“, „sound()“ i „lights()“.

# 3D dizajn i ispis

Opisom softverske implementacije preostaje samo dizajnirati i ispisati dva 3D modela te zaključak. Kako bi kupljene komponente mogle obavljati svoju namijenjenu funkciju potrebno je dizajnirati posebne dijelove i adaptere koji će prilagoditi izlazne osovine servo i DC motora na Lego kompatibilne dijelove poput Lego križne osovine. Iako postoje poduzeća koja se bave izradom dijelova po narudžbi, vrijeme potrebno za izradu i netočne veličine mogu značajno usporiti napredak i biti veoma skupo. Srećom, razvojem i sve širom adaptacijom tehnologija 3D ispisa, ulazna barijera ovog hobija postaje sve pristupačnija. Rana istraživanja provođena su 60-ih godina 20. stoljeća u Battelle Memorial institutu sa sjedištem u Columbusu, saveznoj državi SAD-a Ohio. Začetna ideja je funkcionirala s dvije laserske zrake na drugačijim frekvencijama, no tehnologija do 80-ih godina nije rezultirala nikakvim komercijalnim sustavima. Za razliku od danas čestih tehnologija ispisa poput FDM (engl. Fused Deposition Modeling), inicijalna tehnologija je bila sličnija SLA (engl. Stereolithography). Prelaskom na jednu lasersku zraku krenuo je brži razvoj i drugačije tehnologije ispisa. Značajniji napredak napravljen je 1989. godine kada su Steven Scott Crump i Lisa Crump kreirali tvrtku Statasys i podnijeli zahtjev za patent nad FDM tehnologijom. Istekom Statasys-ovog patenta za FDM tehnologiju, oko 2005. godine pokreću se dva open-source projekta: RepRap i Fab@Home [13]. S njima kreće revolucija i širenje hobija te široka adaptacija tehnologija za 3D ispis. Tehnologija je našla primjenu u mnogim industrijama poput zrakoplovne, automobilske, prehrambene, medicinske, građevinske, modne i elektroničke industrije [14]. Široku primjenu je pronašla upravo zbog benefita koje nudi, ovisno o tehnologiji ispisa. Glavne prednosti su brza izrada prototipa po potrebi, niska cijena materijala i izrade. Neke od mana su limitirana veličina objekta koji se ispisuje, limitiran izbor materijala, relativno spor ispis, toksične pare koje ispuštaju određeni materijali i mogući problemi s autorskim pravima. Razlog korištenja ove tehnologije u svrhu diplomskog rada je mogućnost brze izrade prototipa i neovisnosti o trećim stranama. Najčešća metoda koja se komercijalno koristi je FDM te će ona biti korištena za ovaj diplomski rad, no postoje i druge metode o kojima će biti više pisano u idućem poglavlju.

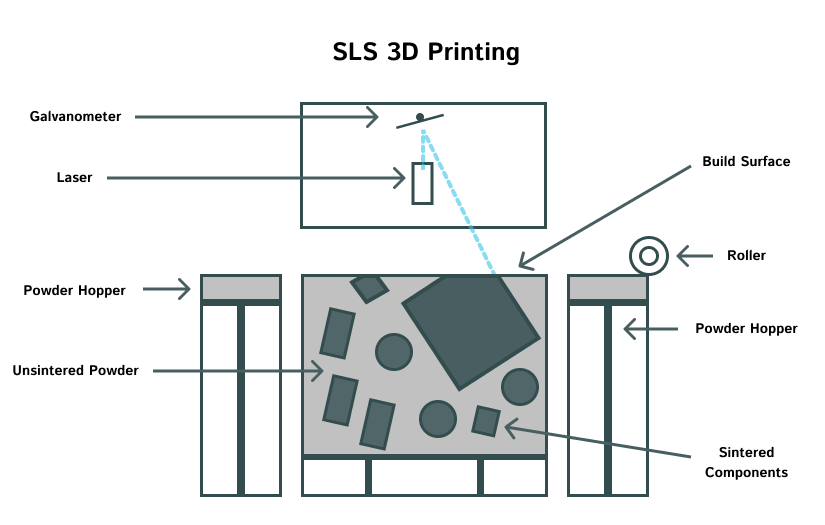
## Pregled tehnika 3D ispisa

Od ranih istraživanja do današnjeg dana, razvijen je niz tehnologija 3D ispisa. Svaka tehnologija ima svoj način rada, primjenu, prednosti i mane. Tehnologije 3D ispisa dijele se prema načinu naslagivanja materijala. Prva kategorija poznata je pod nazivom „fotopolimerizacija u kadi“ (engl. Vat Photopolymerization) gdje UV svjetlost pri kontaktu utječe na smolu i stvrdnjava je u željeni oblik [14][15]. SLA (engl. Stereolithography) koristi UV (engl. Ultraviolet) laser koji precizno otvrdne površinu smole pri kontaktu [16]. DLP (engl. Digital Light Projection) radi na sličnom konceptu, no uz određene razlike. Umjesto laserske zrake, upotrebljava se projektor UV zraka. Dok se kod SLA laserska zraka miče i konstantnog je intenziteta, projektor kod DLP-a je stacionaran i može kontrolirati intenzitet UV zraka. SLA printeri su poznati po kvaliteti rezultata, ali i po višoj cijeni te vremenu potrebnom za ispis. DLP, s druge strane, radi brže i jeftiniji je u odnosu na SLA [17]. Na slici 14 prikazan je način rada DLP i SLA ispisa.



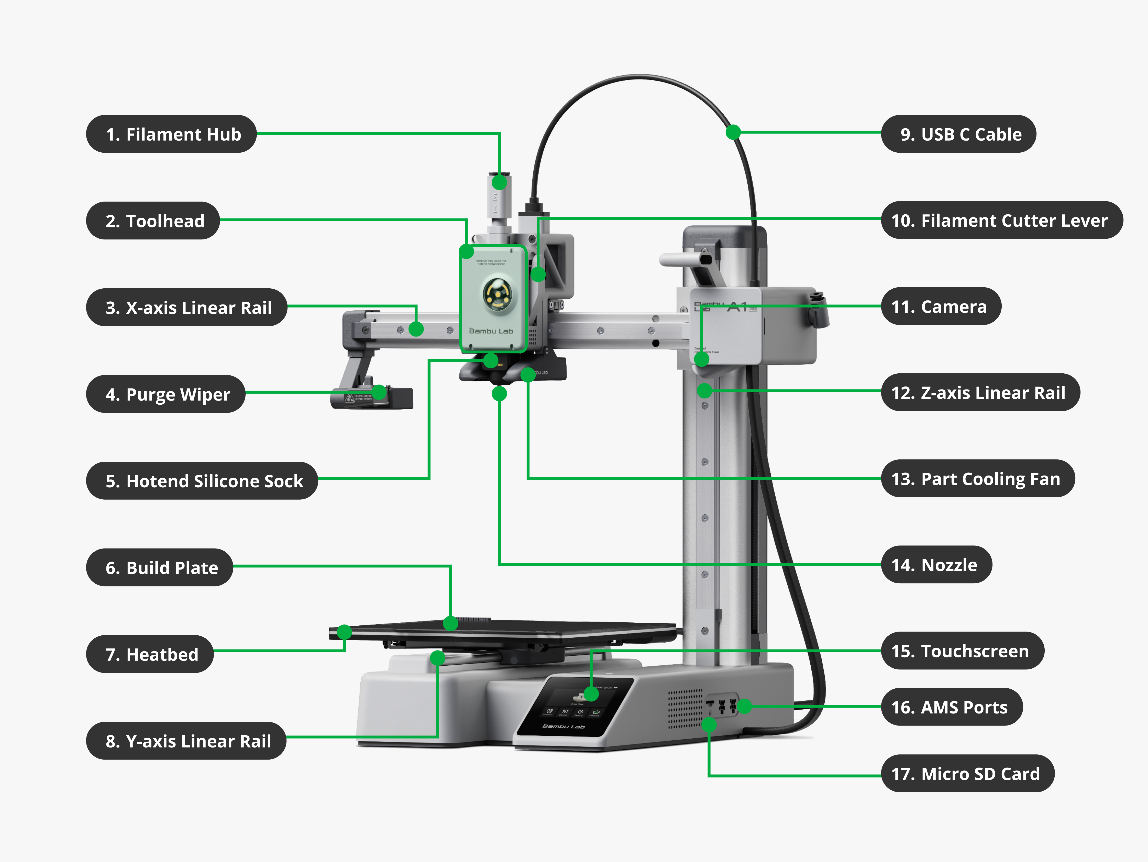
Slika - Način rada DLP i SLA tehnologija ispisa [17]

Ime druge kategorije je „fuzija praha“ (engl. powder bed fusion), a radi na principu otapanja i spajanja praha pomoću elektronskog snopa ili lasera. Podržava materijale poput metala, keramike, termoplastični polimera, kompozita i hibrida [14]. U ovoj kategoriji se nalaze tehnologije SLS (engl. Selective Laser Sintering), DLSM (engl. Direct Metal Laser Sintering), SLM (engl. Selective Metal Sintering), EBM (engl. Electron Beam Melting), SHS (engl. Selective Heat Sintering) i 3DP (engl. 3D Printing) [14][15][16]. Na slici 15 je prikazan pojednostavljen proces SLS 3D ispisa.



Slika - Pojednostavljen dijagram tehnologije SLS 3D ispisa [18]

Treća kategorija je „laminacija ploča“ (engl. Sheet Lamination) bazirana je na slaganju i laminacijom tankih slojeva materijala metodama vezivanja, lemljenja te ultrazvučnog zavarivanja. U ovu kategoriju spadaju tehnologije poput LOM (engl. Laminated Object Manufacturing) i UAM (engl. Ultrasound Additive Manufacturing). Metal je odabrani materijal za ovu tehnologiju. Tijekom rada 3D printera koji funkcionira na LOM tehnologiji, valjak s materijalom se polako odmotava, prelazi preko platforme za 3D ispis i pod utjecajem lasera se spaja s prijašnjim slojem. Neiskorišteni materijal se vuče na drugu stranu i namotava na drugi valjak [14][19]. Četvrta kategorija je „ekstruzija materijala“ (engl. Material Extrusion) i jedna je od najkorištenijih metoda upravo zbog jednostavne implementacije. Primarno radi s termoplastnim polimerima koji se sa špule povlače u mlaznicu printera. Materijal se tamo zagrijava na potrebnu temperaturu koja ovisi o vrsti polimera i u polu-otopljenom stanju se slaže na prethodni sloj. 3D model se ispisuje sloj po sloj. Ovisno o modelu printera, kretanje kroz tri dimenzije se radi na niz načina. Pod ovu kategoriju spada FDM (engl. Fused Deposition Modeling) [14][15][16]. Na slici 16 je prikazan 3D printer baziran na FDM tehnologiji.



Slika - Bambu Lab A1 Mini 3D printer [20]

Peta kategorija je „usmjereno odlaganje energije“ (engl. Directed Energy Deposition) i kompleksniji je proces ispisa od ostalih metoda te se više koristi za popravljanje ili dodavanje materijala na postojeće objekte. Princip rada je sličan metodi ekstruzije materijala, no mlaznica iz koje materijal izlazi nije na fiksiranoj osi već se može slobodnije kretati. Isto tako, razlika se vidi u korištenim materijalima. Iako su keramika i termoplastni polimeri podržani, češće se koriste metali i metalni hibridi u obliku žice ili praha. Primjer ove tehnologije su LENS (engl. Laser Engineered Net Shaping), DMT (engl. Direct Metal Tooling) i BeAM (engl. Be Additive Manufacturing) [14][15]. Šesta kategorija je „mlazanje materijala“ (engl. Material Jetting). Princip rada je sličan klasičnom printeru gdje se fotoosjetljiv materijal u kapljicama pušta iz mlaznice i stvrdnjava se pod utjecajem UV svjetla. Rezultat ispisa odlikuje se veoma glatkom površinom i visokom preciznošću. Podržava više materijalni ispis i širok spektar materijala [14][15]. Sedma kategorija je „mlazanje veziva“ (engl. Binder Jetting) gdje se tekuće, vezivno sredstvo polaže na materijal u obliku praha. Kemijskom reakcijom vezivnog sredstva s prahom formira čvrsti sloj materijala. Ova tehnologija podržava niz materijala poput metala, pijeska, polimera, hibrida i keramike. Odlike ove tehnologije su jednostavnost, brzina, cijena i mogućnost ispisa velikih proizvoda [14][15].

## Filamenti za 3D ispis

Sve prethodno navedene metode ispisa zahtijevaju neku vrstu filamenta koji će se koristiti za ispis modela. Filament je drugi naziv za materijal koji se često koristi u kontekstu 3D ispisa. Ovisno o tehnologiji koja se koristi, postoji niz materijala s različitim karakteristikama i primjenama. No prosječnom potrošaču najdostupniji su materijali u obliku tekuće smole za SLA/SLS tehnologije i termoplastični polimeri u obliku niti namotane na špulu za FDM tehnologiju. Tehnologija koja će biti korištena za svrhu ovog diplomskog rada je FDM zbog čega će fokus biti na filamentima korištenima u ovoj tehnologiji. Na slici 17 prikazan je primjer filamenta u obliku niti namotanog na špulu.



Slika - Primjer filamenta u obliku niti na špuli [21]

Najčešći i najdostupniji filamenti za FDM tehnologiju su termoplastični polimeri, a pod tu kategoriju spadaju PLA (engl. Poly-Lactic Acid), PETG (engl. Polyethylene Terephthalate Glycol-Modified), ABS (engl. Acrylo-nitrile Butadiene Styrene), ASA (engl. Acrylonitrile Styrene Acrylate), PVA (engl. Polyvinyl Alcohol), PC (engl. Polycarbonate), PPS (engl. Polyphenylene Sulfide), TPE (engl. Thermoplastic Elastomers), TPU (engl. Thermoplastic Polyurethane) i termoplastični polimerni kompoziti ojačani s karbonskim/staklenim vlaknima [22][23]. Od navedenih, PLA je najjednostavniji za korištenje i često je preporučen početnicima. Odlikuje se biorazgradivošću i boljim ekološkim karakteristikama, nedostatku toksičnih para tijekom procesa ispisa, visokom podržanom brzinom i rezolucijom ispisa te otpornosti na savijanje i skupljanje. PLA je dostupan u tvrdoj i mekanoj varijanti što utječe na fleksibilnost ispisanih dijelova. Mane PLA filamenta su niska otpornost na toplinu i lošija mehanička svojstva u odnosu na idući filament. ABS se odlikuje boljim mehaničkim svojstvima, visokoj otpornosti na toplinu, glatkoj površini i brzom stvrdnjavanju. Korištenje ABS filamenta kreira toksične pare zbog kojeg je potrebno imati zatvoreno kućište. Uz to, ABS nije biorazgradiv te s vremenom degradira pod utjecajem UV zraka. PETG filament ima mješavinu karakteristika već navedenih materijala. Ima malo bolja mehanička svojstva od ABS-a, bolju otpornost na UV zrake, bolje prianjanje između slojeva i manje savijanja od ABS-a, bolju reciklažu od ABS-a iako također nije biorazgradiv. Otpornost na toplinu je lošija od ABS-a, ali još uvijek je bolja od PLA-a. Uz navedeno, apsorbira vlagu kao i ABS što s vremenom utječe na kvalitetu ispisa [22][23][24]. Odabrani filament mora imati dobra mehanička svojstva, otpornost na toplinu koja može nastati kao rezultat trenja i ne proizvodi toksične pare tijekom ispisa. Uračunavajući sve karakteristike PLA, ABS i PETG, filament koji najbolje odgovara kriterijima je PETG. Kao i ABS, ima dobra mehanička svojstva i dovoljno visoku otpornost na toplinu, no ne proizvodi toksične pare pri ispisu 3D modela. Iako bi najbolja opcija bio ABS filament jer su i Lego dijelovi napravljeni od ABS plastike, nedostatak kućišta sprečava njegov odabir i kao kompromis preostaje PETG filament.

## Upoznavanje s Bambu Lab A1 Mini 3D printerom

Na tržištu se nudi velik broj 3D printera raznih proizvođača, specifikacija i podržanih filamenata. Od najjeftinijih do najskupljih modela, svatko može pronaći odgovarajući printer uz dovoljno istraživanja, a gotovo svi podržavaju ispis pomoću PLA i PETG filamenta. Glavna karakteristika na koju bi se trebala obratiti pozornost je cijena printera, no ne u vakuumu. Cijena može biti dobar indikator kvalitete i značajki printera, ali bitno je detaljno proučiti specifikacije printera. Ono na što bi trebalo obratiti pozornost pri odabiru su veličina površine za ispis, brzina ispisa, koje filamente printer podržava, promjer mlaznice, maksimalna temperatura površine ispisa i mlaznice. Osim osnovnih specifikacija, valja provjeriti s kojim dodatnim značajkama printer dolazi za bolje iskustvo korištenja. Primjer dodatnih značajki može biti aktivna kompenzacija protoka filamenta, aktivno poništavanje buke motora printera, ispis s više boja filamenata, automatska kalibracija i negiranje vibracija. Na temelju specifikacija, značajki i cijene odabran je 3D printer proizvođača Bambu Lab. Riječ je o Bambu Lab A1 Mini 3D printeru, najjeftinijem iz njihove ponude. Namijenjen je početnicima i na glasu je najjednostavnijeg printera za postavljanje i korištenje. U kutiji dolazi većinom složen. Neke manje dijelove treba ugraditi i pritegnuti nekoliko vijaka. Pri prvom uključivanju je potrebno obaviti kratak postupak postavljanja i kalibracije što sveukupno traje 20 do 30 minuta ležernim tempom. Na slici 18 je vidljiv Bambu Lab A1 Mini printer u neaktivnom stanju.

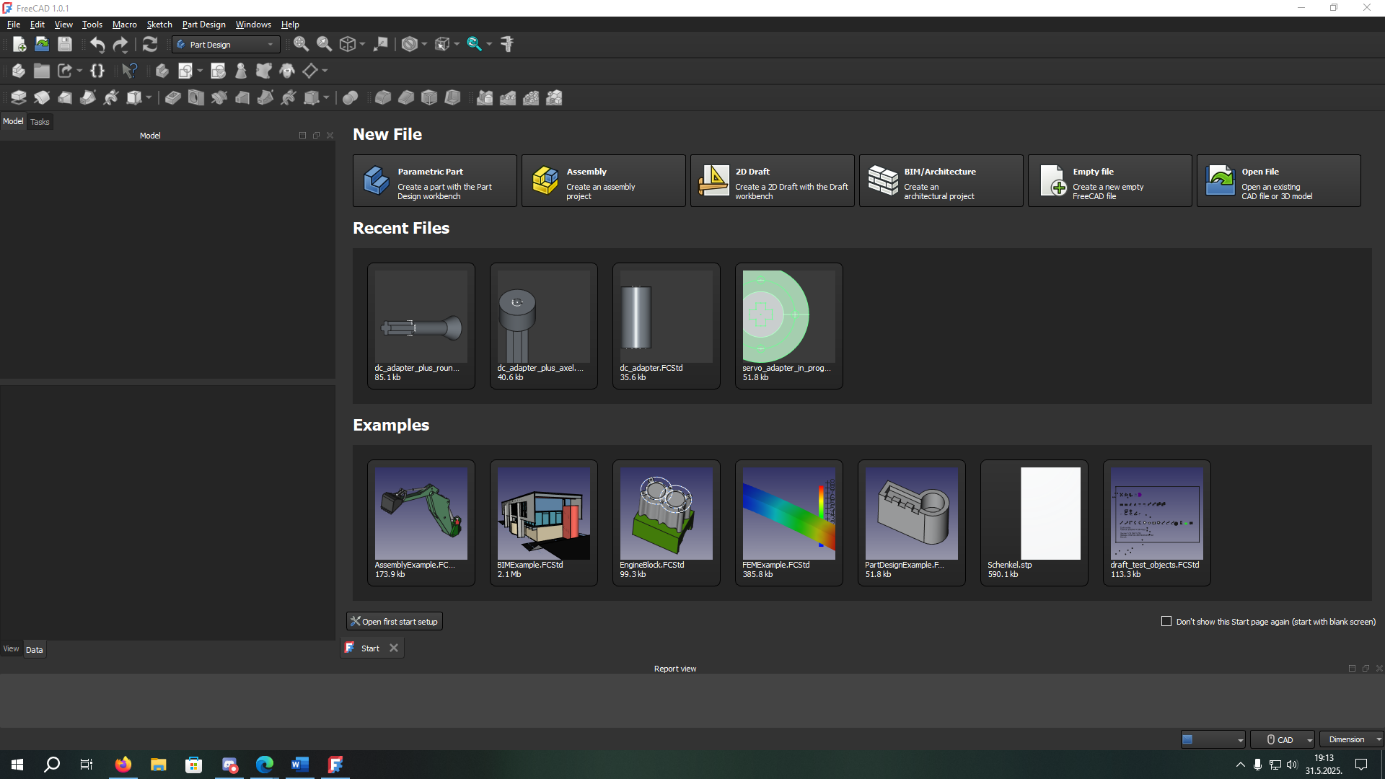


Slika - Bambu Lab A1 Mini u neaktivnom stanju [autor]

Po pitanju konkretnih specifikacija printera, maksimalna površina ispisa je 180x180x180 mm3 što je manje od konkurentnih 3D printera slične cijene. Printer dolazi s mlaznicom promjera 0.4mm od nehrđajućeg čelika što je standardno za većinu printera, postiže optimalan balans za dnevno korištenje po pitanju brzine, rezolucije ispisa i mehaničkih karakteristika ispisanih modela. Površina za ispis je teksturirana i idealna je za početnike zbog minimalne potrebne brige o površini te odličnom prianjanju filamenta. Podržani su PLA, PETG, TPU i PVA filamenti. Dodatne značajke koje su prije spomenute pisane su prema značajkama koje nudi ovaj printer, no za podršku višebojnog ispisa potrebno je zasebno nabaviti AMS (engl. Automatic Material System). Za prijenos 3D modela na printer zaslužna je podrška za WiFi mrežu ili microSD kartica koja dolazi s 3D printerom [25]. Daljinski nadzor nad printerom je moguć putem mobilne aplikacije Bambu Handy koja pruža sekundarnu uslugu pronalaženja 3D modela za ispis. U slučaju kakvih grešaka pri ispisu, mobilna aplikacija će korisniku poslati obavijesti o problemu dokle god su 3D printer i korisnik spojeni na mrežu. Uz mobilnu aplikaciju, moguće je preuzeti aplikaciju za računala zvanu Bambu Studio koja osim funkcija sličnih mobilnoj aplikaciji nudi opciju dodatne obrade modela za ispis. Moguće je podesiti visinu slojeva, odabrati jedan od niza profila za optimizaciju ispisa ovisno o potrebama, dodavanje potpore za dijelove koji vise i još mnogo drugih opcija. Osim ove specifične aplikacije, mogu se koristiti aplikacije treće strane poput OrcaSlicer, PrusaSlicer, Chitubox, Lychee i Cura. Neke aplikacije su besplatne, neke se plaćaju, neke su open-source. No prije nego što se može se model može ispisati, potrebno ga je dizajnirati u nekom od brojnih programa za 3D dizajn.

## Odabir programa za 3D dizajn

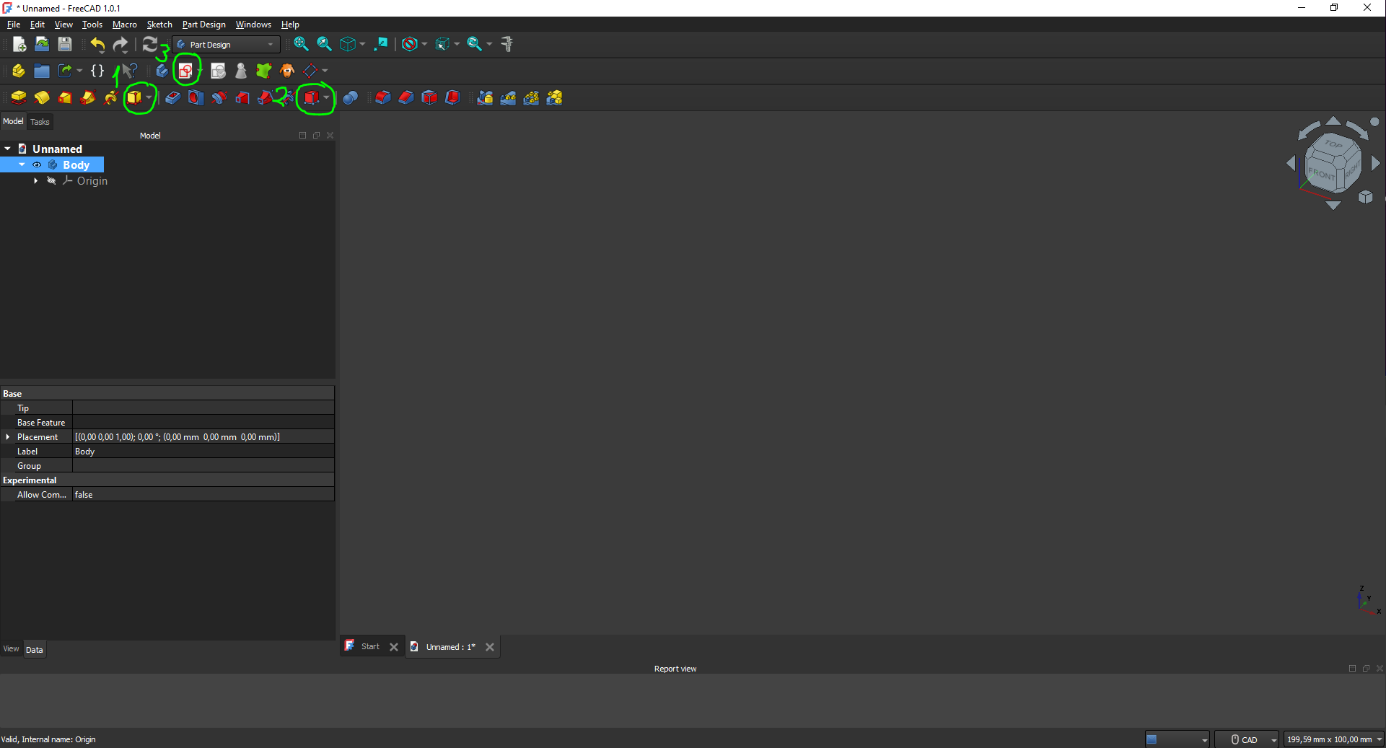
Programi za 3D dizajn dolaze s raznim sučeljima, mogućnostima, krivuljama učenja i cijenama. Potrebe korisnika diktiraju odabrani program i njegove mogućnosti, a razlike u programima su prema upotrebi. Poznatiji programi za 3D dizajn su oni s mogućnošću organskog modeliranja i animiranja kao što su Blender, Maya i Autodesk 3ds Max dok se za izradu preciznih komponenti koriste CAD (engl. Computer-Aided Design) programi poput Fusion 360, AutoCAD, Tinkercad i FreeCAD. CAD softver se koristi za izradu, izmjenu i optimizaciju digitalnih modela. Glavna prednost CAD programa je u preciznosti izrade modela. Većina ih radi na principu parametarskog modeliranja, tj. moguće je definirati modele koristeći parametre poput varijabli za dimenzije i kreirati veze između modela. Jedan od takvih programa s parametarskim modeliranjem i cijenom od 0€ je FreeCAD. Na slici 19 prikazano je sučelje FreeCAD programa.



Slika - Početno sučelje FreeCAD programa [autor]

FreeCAD i svi ostali programi za 3D dizajn su poprilično kompleksni za korištenje. Potrebno ih je dobro proučiti i isprobati kako bi se upoznale sve funkcije. Za svrhu ovog diplomskog rada neće biti potrebno detaljno proučavati FreeCAD, dovoljno će biti poznavanje nekoliko opcija. Prvi korak bi bio kreiranje nove datoteke pritiskom na gumb „Parametric Part“. Otvorit će se prazno sučelje koje je spremno za dizajn modela. Na slici 20 je prikazano prazno FreeCAD sučelje spremno za dizajn 3D modela i zelenom bojom su zaokružene tri najbitnije opcije koje su minimalno potrebne za osnovni rad.

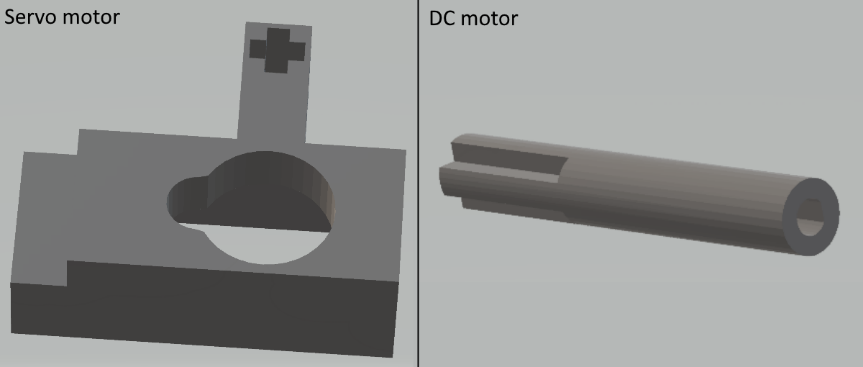
Slika - Prazno FreeCAD sučelje spremno za dizajn modela [autor]



Za početak, potrebno je dodati model kao bazu što se radi pomoću prve zaokružene opcije. Modeli koji se mogu dodati su kocka, cilindar, sfera, stožac, elipsoid, torus, prizma i klin. Isti modeli mogu se oduzeti od glavnog tijela modela korištenjem druge zaokružene opcije. Treća opcija dopušta korisniku stavljanje vizualnih markera za lakšu orijentaciju i određivanje mjera.

## Proces dizajna

Kao što je spomenuto u prijašnjem poglavlju, FreeCAD je odabrani program za proces dizajna. Uz nekoliko sati rada s programom moguće je steći dovoljno iskustva za dizajniranje osnovnih 3D modela. Proces dizajna može započeti tek nakon odabira opcije „Parametric Part“. Pritiskom na navedenu opciju, otvara se sučelje s praznim prostorom za dizajn. Prvi korak bi bio kreirati osnovu koja će se dorađivati dok se ne dobije željeni oblik. Za to služi opcija „Create an additive primitive“, a omogućava dodavanje niza osnovnih oblika. Ovisno o modelu koji je potreban, može se odabrati drugačija baza. Dodavanje materijala na bazu se radi korištenjem iste opcije. No u slučaju da se dio modela treba ukloniti postoji opcija „Create a subtractive primitive“. Ona će kreirati odabrani oblik i oduzet ga od baze na mjestima gdje se preklapaju. Orijentiranje modela moguće je preko kocke koja se nalazi u gornjem desnom dijelu prostora za modeliranje. Za kreiranje skica na plohama modela koristi se opcija „Create sketch“ koja omogućuje dodavanje raznih oznaka na odabranoj plohi i korisna pri preciznom pozicioniranje novih oblika. Na slici 21 se nalaze adapteri kreirani za servo i DC motore.

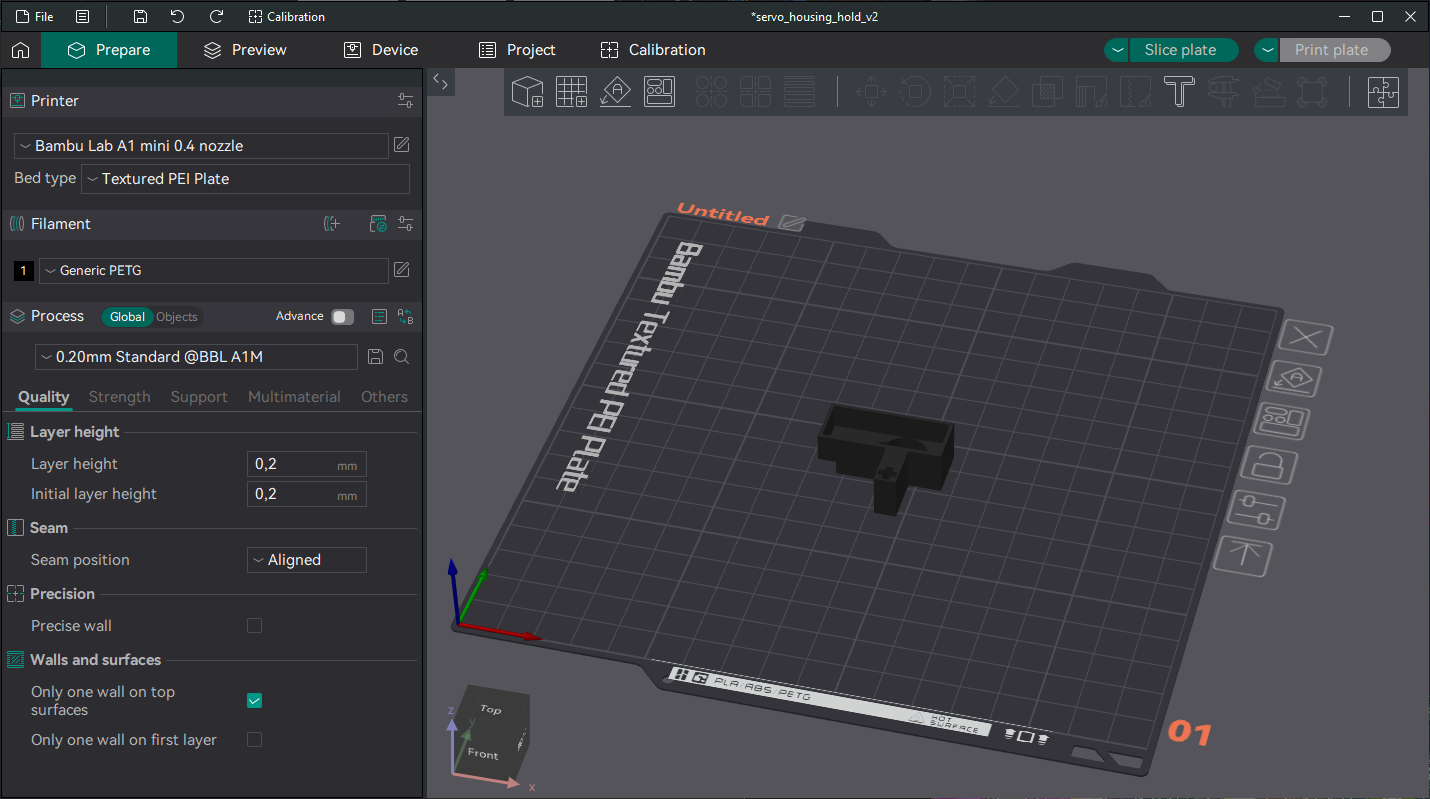


Slika - Adapteri za servo i DC motore [autor]

Za servo motor je bilo potrebno kreirati kućište koje će motor držati na mjestu. Adapter koji prilagođava izlaznu osovinu Tower Pro SG90 servo motora već postoji i bio je javno dostupan, a kreirao ga je korisnik imena HorcikDesigns. Stranica na kojoj je adapter bio više nije dostupna, no sve zasluge za izradu adaptera idu tom korisniku. Model adaptera za DC motor je u potpunosti dizajn autora ovog diplomskog rada i specifično je napravljen s Hobbywing Quicrun 2435 SL Brushless Motor G3 DC motorom u planu. Na desnoj strani desne slike vidljiva je rupa u koju ulazi izlazna osovina DC motora dok je lijeva strana Lego osovina kompatibilna s pravim Lego dijelovima.

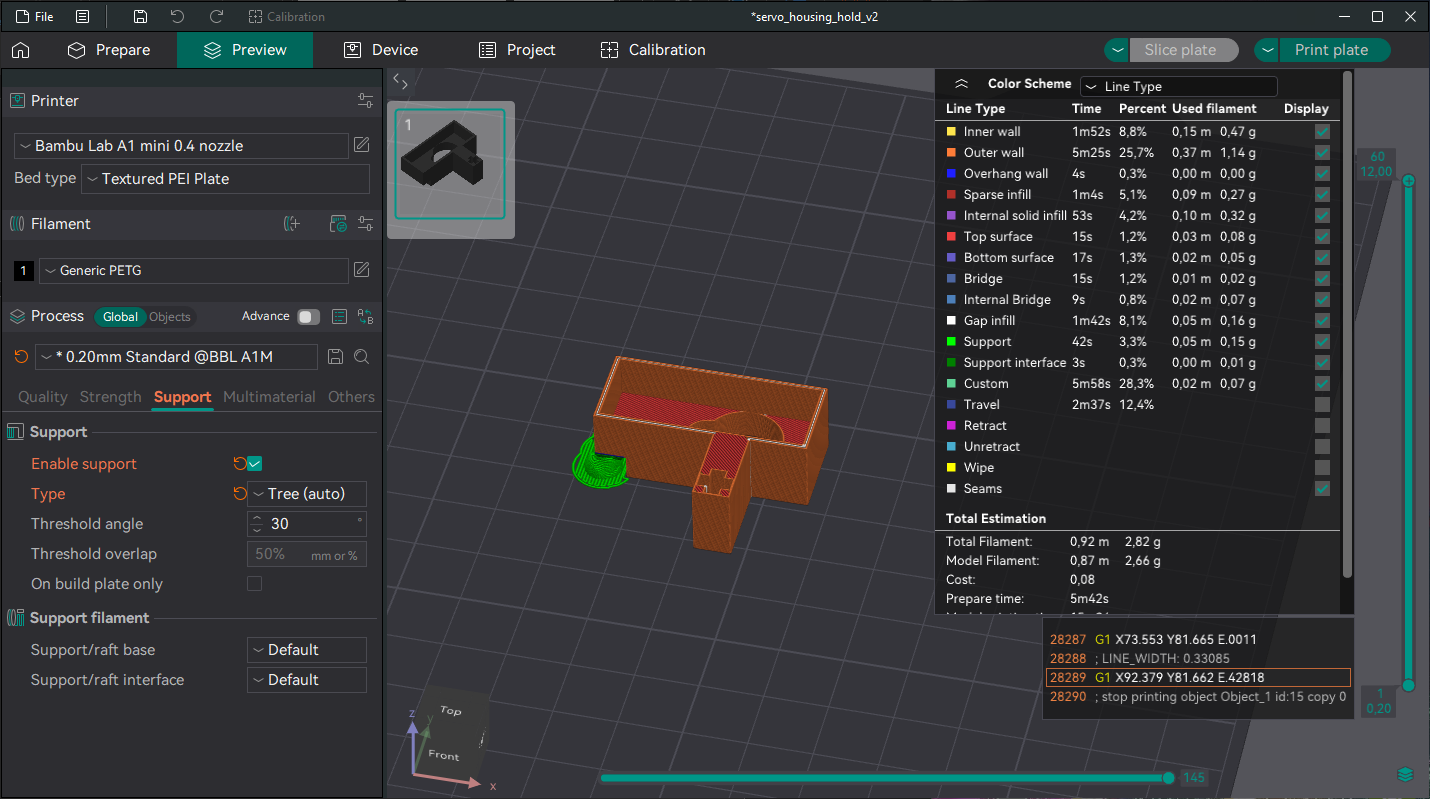
## Proces ispisa

S dizajnom gotovim, vrijeme je za ispis napravljenih 3D modela. Proces počinje s odabirom filamenta koji će odgovarati potrebama. Za kućište servo motora odabir nije toliko bitan jer adapter služi isključivo za fiksiranje servo motora u mjestu. Iako će se adapter za osovinu servo motora micati, odabir filamenta ne igra veliku ulogu. Stoga se za sve adaptere servo motora može koristiti PLA ili PETG filament. No za adapter koji će ići na DC motor filament je veoma bitan zbog trenja koje će stvarati toplinu. PETG filament je otporniji na toplinu od PLA, ali količina trenja će i dalje postepeno fizički uništavati sam adapter i sve plastične Lego dijelove koji su u kontaktu s adapterom. Adapter za DC motor je potrebno podmazati kako bi mogao podnijeti visok broj okretaja DC motora jer trenje može izazvati blago omekšavanje PETG filamenta korištenog za ispis adaptera. Nakon što je filament odabran i pripremljen za ispis, potrebno je proći proces koji se zove „slicing“ u alatima za ispis. „Slicing“ se odnosi na rezanje modela na slojeve i generiranje takozvanog G koda koji printeru daje podatke kako i gdje micati glavu za ispis. Alat korišten u ovu svrhu je „OrcaSlicer“. U njemu je potrebno minimalno podesiti postavke poput modela printera, širine mlaznice, vrsta podloge za ispis i vrstu filametna. Na slici 22 prikazano je sučelje alata OrcaSlicer s učitanim modelom kućišta servo motora.



Slika - OrcaSlicer sučelje [autor]

Kada se model učita u alat, njime se još može manipulirati. Moguće je promijeniti veličinu modela, prerezati ga prema nekoj osi, rotirati u nekom smjeru, nadodati na ili oduzeti od njega dodatne objekte i još niz opcija. Modeli su u FreeCAD alatu rađeni prema mjerama koje trebaju imati u stvarnosti zbog čega dodatne manipulacije nisu potrebne. Jedino što se eventualno može podesiti je orijentacija modela kako bi se smanjila ili u potpunosti otklonila potreba za potpornom strukturom pri ispisu. U slučaju da su sve postavke onakve kakve trebaju biti ili su napravljene promjene, vrijeme je za ispis modela. Pritiskom na gumb „Slice plate“ alat OrcaSlicer će generirati G kod potreban za rad 3D printera. Model će poprimiti niz boja vidljivih na popisu uz desni rub prozora alata i što predstavljaju. Na slici 23 je vidljiva spomenuta lista.



Slika - Primjer sučelja spremnog za ispis [autor]

Ako se sve čini u redu, vrijeme je za ispis modela. Pritiskom na gumb „Print plate“ šalje se zadatak 3D printeru na ispis. Potrebno vrijeme za ispis modela ovisi o veličini i kompleksnosti modela te postotku popunjenosti i veličini mlaznice. Kada ispis završi, 3D printer će stati i stajati tako dok mu se ne zada novi zadatak ili ga se ugasi. Ugradnjom adaptera servo i DC motora završava se rad na prototipu te preostaje samo zaključak.

# Zaključak

Cijeli ovaj rad pisan je u drugačijem redoslijedu nego što bi se moglo zaključiti iz poglavlja. Sve je započelo s pisanjem programskog koda i testiranjem komponenti bez ugradnje u Lego Technic set. Plan je bio prvo kreirati sustav koji radi „na suho“ i neće slučajno oštetiti dijelove Lego modela. Iako je trebalo vremena da sve proradi kako je zamišljeno, krajnji rezultat i dalje ima manje mane što se tiče svjetala automobila, no ne utječe na cjelokupnu funkcionalnost rada. Najveći problem je ograničen fizički prostor prototipa. Kabina modela je prostorno ograničena što je rezultiralo pomicanjem pretvarača napona i mikroupravljača na krov modela. Ovim korakom je prostor oslobođen za ostale komponente, no narušen je estetski izgled prototipa. Kompletna hardverska implementacija je naišla na niz problema koji se nisu uvijek mogli predvidjeti. Na primjer, pronalazak XT60 konektora i odgovarajućih Li-Po baterija respektabilnog kapaciteta po razumnoj cijeni je teži od očekivanog. Postoji niz optimizacija i promjena koje bi trebalo napraviti kako bi diplomski rad bio kvalitetniji. Odabir komponenti je mogao biti bolji i orijentiran na efikasniju alokaciju prostora. Baterije su mogle biti manjeg kapaciteta. DC motor je mogao biti fokusiran na okretni moment umjesto na brzinu, tj. manje rotacija izlazne osovine. Mogla se izraditi posebna vanjska ljuska koja bi sjedila na platformi Lego Technic seta umjesto vanjskih Lego panela koji čine tijelo vozila. Glavna promjena koja bi najviše utjecala na ovaj rad bi bila izrada posebne platforme od Lego Technic dijelova koja bi od početka bila napravljena da akomodira sve komponente. Unatoč problemima, rezultat diplomskog rada je funkcionalan prototip baziran na varijanti ESP32 mikroupavljača i upravljan putem Bluetooth veze. Sve funkcije koje su spomenute na početku samog rada su implementirane i funkcioniraju onako kako su predstavljene.

# Literatura

1. DFRobot. „FireBeetle Board ESP32 E SKU DFR0654“ *DFRobot.com*., preuzeto 12.6.2025. s <https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654> (pristupano 24.6.2025.)
2. DFRobot. „FireBeetle 2 ESP32-E IoT Microcontroller with Header (Supports Wi-Fi & Bluetooth)“ *DFRobot.com*., preuzeto 12.6.2025. s <https://www.dfrobot.com/product-2231.html> (pristupano 24.6.2025.)
3. EuroRC. „Hobbywing Quicrun 16BL30 Waterproof 30A Sensorles Esc“ *EuroRC.com*., preuzeto 12.6.2025. s <https://www.eurorc.com/product/12346/hobbywing-quicrun-16bl30-waterproof-30a-sensorles-esc> (pristupano 24.6.2025.)
4. Hobbywing. „Quicrun user manual“ *Hobbywing.com*., preuzeto 12.6.2025. s <https://www.hobbywing.com/en/uploads/file/20221015/f9de7d2e620af2db1841526eabab2092.pdf> (pristupano 24.6.2025.)
5. EuroRC. „Hobbywing QuicRun 2435 SL Brushless Motor G3“ *EuroRC.com*., preuzeto 12.6.2025. s <https://www.eurorc.com/product/35996/hobbywing-quicrun-2435-sl-brushless-motor-g3> (pristupano 24.6.2025.)
6. Sparkfun Electronics. „Servos Explained“ *sparkfun.com*., preuzeto 12.6.2025. s <https://www.sparkfun.com/servos> (pristupano 24.6.2025.)
7. G. Marin-Garcia, G. Vazquez-Guzman, J. M. Sosa, A. R. Lopez, P. R. Martinez-Rodriguez i D. Langarica, (2020). „Battery Types and Electrical Models: A Review“ *2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, str. 1-6, Dostupno: IEEE Xplore, <https://ieeexplore.ieee.org/document/9258711> (preuzeto 14.4.2025.)
8. EuroRC. „Differences Between NiMH and LiPo Batteries“ *EuroRC.com*., preuzeto 12.6.2025. s <https://www.eurorc.com/page/69/differences-between-nimh-and-lipo-batteries> (pristupano 12.6.2025.)
9. DFRobot. „Fermion: DFPlayer Pro - Dual Channel MP3 Module with 128MB Storage (Arduino / Raspberry Pi Compatible)“ *DFRobot.com*., preuzeto 12.6.2025. s <https://www.dfrobot.com/product-2232.html> (pristupano 24.6.2025.)
10. Worldsemi. „WS2812B“ *adafruit.com*., preuzeto 12.6.2025. s <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf> (pristupano 24.6.2025.)
11. Evan-Amos. „The Vanamo Online Game Museum - Sony PlayStation 4“ *WikiMedia.org*., preuzeto 12.6.2025. s <https://commons.wikimedia.org/wiki/User:Evan-Amos/VOGM/PlayStation4> (pristupano 24.6.2025.)
12. Lego Certified Store. „2022 Ford GT“ *lstore.hr*., preuzeto 12.6.2025. s <https://lstore.hr/products/lego-technic-42154> (pristupano 24.6.2025.)
13. A. Su i S. J., Al’Aref, (2018). „History of 3D Printing“ *3D Printing Applications in Cardiovascular Medicine*, str. 1-10, Dostupno: ScienceDirect, <https://doi.org/10.1016/B978-0-12-803917-5.00001-8> (preuzeto 12.5.2025.)
14. N. Shahrubudin, T.C. Lee i R. Ramlan, (2019). „An overview on 3D printing technology: Technological, materials, and applications“ *Procedia Manufacturing*, sve. 35, str. 1286-1296, Dostupno: ScienceDirect, <https://doi.org/10.1016/j.promfg.2019.06.089> (preuzeto: 12.5.2025.)
15. C. M. Madla, S. J. Trenfield, A. Goyanes, S. Gaisford i A. W. Basit, (2018). „3D printing technologies, implementation and regulation: An overview“ *AAPS Advances in the Pharmaceutical Sciences Series*, sve. 31, str. 21-40, Dostupno: Springer Verlag, <https://doi.org/10.1007/978-3-319-90755-0_2> (preuzeto 12.5.2025.)
16. M. Ntousia i I. Fudos, (2019). „3D Printing Technologies and Applications: An Overview“ *CAD Conference and Exibition*, str. 243-248, Dostupno: Cad Conference, <https://doi.org/10.14733/cadconfp.2019.243-248> (preuzeto: 12.5.2025.)
17. Manufactur3D Team. „The Difference between DLP and SLA 3D Printing Technology“ *manufactur3dmag.com*., preuzeto 12.6.2025. s <https://manufactur3dmag.com/difference-dlp-sla/> (pristupano 24.6.2025.)
18. G. Shields. „Design for SLS 3D Printing: The Ultimate Guide“ *PrintPool.co.uk*., preuzeto 12.6.2025. s <https://www.printpool.co.uk/articles/design-for-sls-3d-printing-the-ultimate-guide> (pristupano 12.6.2025.)
19. Additive Manufacturing Research Group. „About Additive Manufacturing - Sheet Lamination“ *lboro.ac.uk*., preuzeto 12.6.2025. s <https://www.lboro.ac.uk/research/amrg/about/the7categoriesofadditivemanufacturing/sheetlamination/> (pristupano 12.6.2025.)
20. Bambu Lab. „A1 mini“ *bambulab.com*., preuzeto 12.6.2025. s <https://wiki.bambulab.com/en/a1-mini> (pristupano 12.6.2025.)
21. 3DJake. „SUNLU PLA Green, 1,75 mm / 1000 g“ *3djake.hr*., preuzeto 12.6.2025. s <https://www.3djake.hr/sunlu/pla-green-8> (pristupano 12.6.2025.)
22. M. Kamran i A. Saxean, (2016). „A Comprehensive Study on 3D Printing Technology“ *MIT International Journal of Mechanical Engineering*, sve. 6, br. 2, str. 63-68, Dostupno: Google Scholar, <https://d1wqtxts1xzle7.cloudfront.net/88880384/21002-book-mechanical-eng.3-libre.pdf?1658560030=&response-content-disposition=inline%3B+filename%3DA_Comprehensive_Study_on_3D_Printing_Tec.pdf&Expires=1750779827&Signature=M2Y9fp0fDtIBlUstcfQxRJMukoYwfUoosBN6kzc~9GjzREuiFod5CXaAAwSwt2y7-xDq5zJ2YG-1r554X3CkgWbHP0bwsJNtIPzL2xhG4kG2F9V6ZAfwp~0zRy65efPdUxE3Tm-ik~FmQ07thXC4XSfD4bziAK9J7N~2~WCoh~RFZJNstvXcmenrqbNStmoD3wKg0kTF0reRgg5U-sk~qVRObb~l8alYbgrMjnQ8SEjRc0~frOPlYHNLGtUUvNIvmGUP1rU3sXadzFli7O~YeGOaA09941lIC7QetBsxbJcVD-jqBQaIOAlcBMs~6hhJpc6g-IA-1tXvmW6h~N4Xjg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA> (preuzeto: 12.5.2025.)
23. Bambu Lab. „FDM 3D Printing Filament guide for Beginners“ *bambulab.com*., preuzeto 12.6.2025. s <https://wiki.bambulab.com/en/knowledge-sharing/filament-guide-beginners> (pristupano 24.6.2025.)
24. K. de Naoum i R. Piccoli. „ABS vs. PETG: What’s the Difference?“ *Xometry.com*., preuzeto 12.6.2025. s <https://www.xometry.com/resources/3d-printing/abs-vs-petg-3d-printing/> (pristupano 12.6.2025.)
25. Bambu Lab. „Bambu Lab A1 mini 3D Printer“ *bambulab.com*., preuzeto 12.6.2025. s <https://eu.store.bambulab.com/products/a1-mini?from=navigation> (pristupano 12.6.2025.)

# 

# Prilozi

Kompletan programski kod:

#include <PS4Controller.h>

#include "esp\_bt\_main.h"

#include "esp\_bt\_device.h"

#include "esp\_gap\_bt\_api.h"

#include "esp\_err.h"

#include "FastLED.h"

#include <ESP32Servo.h>

#include <DFRobot\_DF1201S.h>

#include <HardwareSerial.h>

int speedCurrent = 0, speedCatchup = 0, cycleTrack = 0, prevSpeed = 0, gearShift = 0, shiftComplete = 0, triggered = 0;

int trackL2 = 0, trackR2 = 0, tempCurTime = 0, tempTotTime = 0, tempDiff = 0;

int speedCuCircle = 0, speedCaCircle = 0;

int actualSpeed = 0;

int lightTurn = 0, onlyOne = 0;

int frontHead[3] = {220, 247, 241}, rearBrakeReg[3] = {31, 0, 0}, rearBrakeAct[3] = {255, 0, 0}, hazard[3] = {255, 100, 0};

int accspeed = 1, deaccspeed = 1;

int mp3CurFile = 0;

int LstickX = 0, mappedLstickX = 0, trTrack = 0, sqTrack = 0, ciTrack = 0, crTrack = 0, rBumpTrack = 0, lBumpTrack = 0;

unsigned long lastTimeStamp = 0, LTSbuttons = 0, hazardTS = 0;

bool tr = PS4.Triangle(), sq = PS4.Square(), ci = PS4.Circle(), cr = PS4.Cross(), rt = PS4.R2(), lt = PS4.L2();

bool rBump = PS4.R1(), lBump = PS4.L1();

HardwareSerial DFP(2);

DFRobot\_DF1201S DF;

Servo steering, motor;

#define LED\_PIN D3

#define NUM\_LEDS 4

#define EVENTS 0

#define BUTTONS 1

#define JOYSTICKS 1

#define SENSORS 0

CRGB leds[NUM\_LEDS];

void setup(){

  FastLED.addLeds<WS2812, LED\_PIN, GRB>(leds, NUM\_LEDS);

  FastLED.setBrightness(96);

  brakeReg();

  Serial.begin(115200);

  DFP.begin(115200, SERIAL\_8N1, D11, D10);

  if (!DF.begin(DFP)) { // If the object did not initialize, then its configuration is invalid

    Serial.println("Invalid EspSoftwareSerial pin configuration, check config");

    while (1) { // Don't continue with invalid configuration

      delay (1000);

    }

  }

  DF.setVol(0);

  DF.switchFunction(DF.MUSIC);

  delay(2000);

  DF.setPlayMode(DF.SINGLE);

  DF.setVol(15);

  DF.playFileNum(1);

  mp3CurFile = millis();

  PS4.attach(stateCheck);

  PS4.attachOnConnect(onConnect);

  PS4.attachOnDisconnect(onDisConnect);

  PS4.begin();

  Serial.println("");

  steering.attach(D12);

  motor.attach(D13, 1000, 2000);

  steering.write(90);

  /\*Serial.printf("Throttle range setup!!!\n");

  Serial.printf("Neutral!\n");

  motor.writeMicroseconds(1500);

  delay(3000);

  Serial.printf("Forward!\n");

  motor.writeMicroseconds(2000);

  delay(3000);

  Serial.printf("Reverse!\n");

  motor.writeMicroseconds(1000);

  delay(3000);

  Serial.printf("Done!\n");\*/

}

void onConnect(){

  Serial.println("Connected!");

}

void onDisConnect(){

  Serial.println("Disconnected!");

}

void steer(){

  LstickX = PS4.LStickX();

  mappedLstickX = map(LstickX, -127, 127, 0, 165);

  if(mappedLstickX >= 85 && mappedLstickX <= 95){

    steering.write(90);

  }else{

    steering.write(mappedLstickX);

  }

}

void accelspeed(){

  if(speedCurrent == 0){

    accspeed = 1;

  }else{

    if(PS4.L2()){

      accspeed = 1;

      //Serial.printf("accspeed for reverse is only 1\n");

    }else{

      if(speedCurrent >= 1 && speedCurrent <= 85){

        accspeed = 1;

        //Serial.printf("accspeed is 1\n");

      }

      else if (speedCurrent >= 86 && speedCurrent <= 170){

        accspeed = 2;

        //Serial.printf("accspeed is 2\n");

      }

      /\*else if (speedCurrent >= 171 && speedCurrent <= 255){

        accspeed = 4;

        //Serial.printf("accspeed is 4\n");

      }\*/

    }

  }

}

int accdeacc(int current, int catchup){

  if(PS4.L2() || catchup < 0){

    if (current == catchup){

      return catchup;

    }else if(current > catchup){

      catchup += deaccspeed;

    }else if (current < catchup){

      catchup -= accspeed;

    }

    if (catchup < -96 && PS4.L2()){

      catchup = -96;

    }

    if (catchup > 0 && !PS4.L2() && !PS4.R2()){

      catchup = 0;

    }

  }else if(PS4.R2() || catchup > 0){

    if (current == catchup){

      return catchup;

    }else if(current > catchup){

      catchup += accspeed;

    }else if (current < catchup){

      catchup -= deaccspeed;

    }

    if (catchup > 144 && PS4.R2()){

      catchup = 144;

    }

    if (catchup < 0 && !PS4.L2() && !PS4.R2()){

      catchup = 0;

    }

  }

  return catchup;

}

void drive(){

  //Add rumble that will give haptic feedback with what's going on, rumble responds to speed

  //A bit too late for this, although it probably isn't too hard to implement

  //Once the controller is connected, the small motor constantly vibrates

  //The heavy motor vibrates only when the car is moving

  //PS4.setRumble(small, large);

  if (millis() - lastTimeStamp > 25){

    if(PS4.Cross()){

      speedCurrent = 0;

    }else{

      if(PS4.R2() && speedCatchup >= 0){

        speedCurrent = PS4.R2Value();

      }else if(PS4.L2() && speedCatchup <= 0){

        speedCurrent = PS4.L2Value()\*(-1);

      }else if(!PS4.R2() || !PS4.L2()){

        speedCurrent = PS4.R2Value();

      }

    }

    accelspeed();

    if (!ci){

      speedCatchup = accdeacc(speedCurrent, speedCatchup);

      //Map to correct values; 1.0 full reverse, 1.5 neutral, 2.0 full forward

      actualSpeed = map(speedCatchup, -255, 255, 1000, 2000);

    }else{

      //Alternative when circle is pressed

      speedCatchup = accdeacc(speedCurrent, speedCatchup);

      speedCaCircle = accdeacc(speedCuCircle, speedCaCircle);

      //Map to correct values; 1.0 full reverse, 1.5 neutral, 2.0 full forward

      actualSpeed = map(speedCaCircle, -255, 255, 1000, 2000);

    }

    //Send impulse to DC motor

    motor.writeMicroseconds(actualSpeed);

    cycleTrack++;

    if(cycleTrack >= 9){

      cycleTrack = 0;

      prevSpeed = speedCatchup;

    }

    lastTimeStamp = millis();

  }

}

void sound(){

  if((millis() - mp3CurFile >= DF.getTotalTime()\*995-tempDiff\*995) || (PS4.R2() && trackR2 == 0 && prevSpeed == 0) || (PS4.R2() && trackR2 == 0) || (!PS4.R2() && trackR2 == 1) || (PS4.L2() && trackL2 == 0 && prevSpeed == 0) || (PS4.L2() && trackL2 == 0) || (!PS4.L2() && trackL2 == 1)){

    Serial.printf("Entry reason:\t");

    if(PS4.R2() && trackR2 == 0 && prevSpeed == 0){

      Serial.printf("Sudden acceleration from idle!");

      triggered = 3;

    }else if(PS4.R2() && trackR2 == 0){

      Serial.printf("Sudden acceleration!");

      if(shiftComplete == 0){

        Serial.printf("\tTriggered 4");

        triggered = 4;

      }else{

        Serial.printf("\tTriggered 5");

        triggered = 5;

      }

    }else if(!PS4.R2() && trackR2 == 1){

      Serial.printf("Sudden deacceleration!");

      if(shiftComplete == 1){

        Serial.printf("\tTriggered 6");

        triggered = 6;

      }else{

        Serial.printf("\tTriggered 7");

        triggered = 7;

      }

    //This part is for reverse sound logic

    }else if(PS4.L2() && trackL2 == 0 && prevSpeed == 0){

      Serial.printf("Sudden reverse acceleration from idle!");

      triggered = 8;

    }else if(PS4.L2() && trackL2 == 0){

      Serial.printf("Sudden reverse acceleration!");

      triggered = 9;

    }else if(!PS4.L2() && trackL2 == 1){

      Serial.printf("Sudden reverse deacceleration!");

      triggered = 10;

    }else{

      Serial.printf("Normal entry!");

      triggered = 0;

    }

    Serial.printf("\n");

    if(speedCatchup >= -16 && speedCatchup <= 16 && !PS4.R2() && !PS4.L2()){

      Serial.printf("Idle\tSpeedCatchup = %d\n", speedCatchup);

      DF.playFileNum(2);

      tempDiff = 0;

      gearShift = 0;

      shiftComplete = 0;

    }

    else if((speedCatchup > 0 && prevSpeed < speedCatchup && gearShift == 0) || triggered == 3 || triggered == 4){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime && shiftComplete == 0 && DF.getCurFileNumber() != 2){

        Serial.printf("Accel gear 1 w/ FF!\n");

        DF.playFileNum(3);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Accel gear 1\n");

        DF.playFileNum(3);

        tempDiff = 0;

      }

      gearShift = 1;

    }

    else if((speedCatchup > 0 && prevSpeed <= speedCatchup && gearShift == 1 && triggered != 6) || triggered == 5){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime && shiftComplete == 1){

        Serial.printf("Accel gear 2 w/ FF!\n");

        DF.playFileNum(4);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Accel gear 2\n");

        DF.playFileNum(4);

        tempDiff = 0;

      }

      shiftComplete = 1;

    }

    else if((speedCatchup > 0 && prevSpeed >= speedCatchup && gearShift == 1 && shiftComplete == 1) || triggered == 6){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime){

        Serial.printf("Deaccel gear 2 w/ FF!\n");

        DF.playFileNum(5);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Deaccel gear 2\n");

        DF.playFileNum(5);

        tempDiff = 0;

      }

      gearShift = 0;

    }

    else if((speedCatchup > 0 && prevSpeed >= speedCatchup && gearShift == 0 && shiftComplete == 1 && speedCatchup <= 184) || triggered == 7){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime){

        Serial.printf("Deaccel gear 1 w/ FF!\n");

        DF.playFileNum(6);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Deaccel gear 1\n");

        DF.playFileNum(6);

        tempDiff = 0;

      }

      shiftComplete = 0;

      gearShift = 0;

    }else if((speedCatchup < 0 && gearShift == 0) || triggered == 8 || triggered == 9){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime){

        Serial.printf("Reverse accel w/ FF!\n");

        DF.playFileNum(7);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Reverse accel!\n");

        DF.playFileNum(7);

        tempDiff = 0;

      }

      gearShift = 1;

    }else if(speedCatchup < 0 && gearShift == 1 && triggered != 10){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime){

        Serial.printf("Reverse top end w/ FF!\n");

        DF.playFileNum(8);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Reverse top end!\n");

        DF.playFileNum(8);

        tempDiff = 0;

      }

    }else if((speedCatchup < 0 && gearShift == 1) || triggered == 10){

      tempCurTime = DF.getCurTime();

      tempTotTime = DF.getTotalTime();

      if(tempCurTime < tempTotTime){

        Serial.printf("Reverse deaccel w/ FF!\n");

        DF.playFileNum(9);

        DF.fastForward(tempTotTime - tempCurTime);

        tempDiff = tempTotTime - tempCurTime;

      }else{

        Serial.printf("Reverse deaccel!\n");

        DF.playFileNum(9);

        tempDiff = 0;

      }

      gearShift = 0;

    }

    if(PS4.R2()){

      trackR2 = 1;

    }else{trackR2 = 0;}

    if(PS4.L2()){

      trackL2 = 1;

    }else{trackL2 = 0;}

    mp3CurFile = millis();

  }

}

//Shuts off all front lights

void shut(){

  for(int i=0; i < 2; i++){     //Cycles though the first and second LEDs

    leds[i] = CRGB(0, 0, 0);    //Sets the R, G & B values to 0 which is OFF

  }

  Serial.printf("Off!\n");      //Prints status to serial monitor when connected to PC

}

//Makes front light white/very light blue and act as headlights

void head(){

  for(int i = 0; i < 2; i++){

    leds[i] = CRGB(frontHead[0], frontHead[1], frontHead[2]);

  }

}

//Makes all lights turn orange

void haza(){

  for(int i = 0; i < 4; i++){

    leds[i] = CRGB(hazard[0], hazard[1], hazard[2]);

  }

}

//Right hand side lights act as right lane change indicators

void rLane(){

  leds[1] = CRGB(hazard[0], hazard[1], hazard[2]);

  leds[2] = CRGB(hazard[0], hazard[1], hazard[2]);

}

//Left hand side lights act as left lane change indicators

void lLane(){

  leds[0] = CRGB(hazard[0], hazard[1], hazard[2]);

  leds[3] = CRGB(hazard[0], hazard[1], hazard[2]);

}

//Makes the brake lights normal (no brakes are applied)

void brakeReg(){

  for(int i = 2; i < 4; i++){

    leds[i] = CRGB(rearBrakeReg[0], rearBrakeReg[1], rearBrakeReg[2]);

  }

}

//Makes the brake lights brighter (technically just more red)

void brakeAct(){

  for(int i = 2; i < 4; i++){

    leds[i] = CRGB(rearBrakeAct[0], rearBrakeAct[1], rearBrakeAct[2]);

  }

}

//Whole bunch of light stuff

void lights(){

  if (millis() - hazardTS > 500){   //Only fires if half a second passed since the last time this fired

    if (trTrack == 1){              //Checks if hazard lights should be on

      if(lightTurn == 0){           //Checks which state should the lights be in; 0 = hazard, 1 = headlights/off  |

        haza();                     //Turns on hazard lights                                                      |

        lightTurn = 1;              //Switches state for next loop                                               /

      }else{                        //                                                                      <---

        if(sqTrack == 0){           //Checks if front lights should act as headlights or be off

          shut();                   //Turns front lights off

        }else{

          head();                   //Turns front lights into headlights

        }

        if(crTrack == 0){           //Checks if brakes are applied

          brakeReg();               //Makes rear lights normal

        }else{

          brakeAct();               //Makes rear lights as if brakes applied

        }

        lightTurn = 0;              //Switches state for next loop

      }

    }

    if(rBumpTrack == 1){            //Same concept, just for right side lights

      if(lightTurn == 0){

        rLane();

        lightTurn = 1;

      }else{

        if(sqTrack == 0){

          shut();

        }else{

          head();

        }

        if(crTrack == 0){

          brakeReg();

        }else{

          brakeAct();

        }

        lightTurn = 0;

      }

    }

    if(lBumpTrack == 1){            //Same concept, just for right side lights

      if(lightTurn == 0){

        lLane();

        lightTurn = 1;

      }else{

        if(sqTrack == 0){

          shut();

        }else{

          head();

        }

        if(crTrack == 0){

          brakeReg();

        }else{

          brakeAct();

        }

        lightTurn = 0;

      }

    }

    hazardTS = millis();            //Updates timekeeping variable

  }

  FastLED.show();                   //Applies whatever state the lights are in

}

//Gets the state of all buttons

void stateCheck(){

  #if BUTTONS

    tr = PS4.Triangle();            //Triangle button - Hazard lights

    sq = PS4.Square();              //Square button - Headlights

    ci = PS4.Circle();              //Circle button - Clutch

    cr = PS4.Cross();               //Cross button - Brakes

    rBump = PS4.R1();               //Right bumper - Right turn indicators

    lBump = PS4.L1();               //Left Bumper - Left turn indicators

    stateChange();                  //Calls function to update states of other components

  #endif

}

//Changes a lot of things, from clutch state to light states

void stateChange(){

  //Flag alteration, npr. when circle is pressed or hazard lights change

  //Circle is pressed, DC motor winds down while the sound plays as usual

  if (ci && ciTrack == 0){

    Serial.printf("Alt mode entered!\n");

    ciTrack = 1;

    speedCuCircle = 0;

    speedCaCircle = speedCatchup;

  }else if(!ci && ciTrack == 1){

    Serial.printf("Alt mode exited!\n");

    ciTrack = 0;

  }

  //While brakes are applied, the deacceleration variable is increased and the car should slow down 4x faster

  //Rear brake lights should light up when X is pressed

  if (cr && crTrack == 0){

    Serial.printf("Brakes applied!\n");

    crTrack = 1;

    deaccspeed = 4; //Maybe 8 if 4 is not enough

    brakeAct();

    FastLED.show();

  }else if(!cr && crTrack == 1){

    Serial.printf("Brakes unapplied!\n");

    crTrack = 0;

    deaccspeed = 1;

    brakeReg();

    FastLED.show();

  }

  //By having a 100ms time buffer, I'm trying to avoid repeat triggering of these ifs for the lights

  if (millis() - LTSbuttons > 100){

    if(tr && trTrack == 0){

      //Turn hazard lights on

      trTrack = 1;

    }else if(tr && trTrack == 1){

      trTrack = 0;

      if (sqTrack == 0){

        shut();

      }else{

        head();

      }

      if(crTrack == 0){

        brakeReg();

      }else{

        brakeAct();

      }

    }

    if(sq && sqTrack == 0){

      Serial.printf("Headlights on!\n");

      head();

      FastLED.show();

      sqTrack = 1;

    }else if(sq && sqTrack == 1){

      Serial.printf("Headlights off!\n");

      for(int i=0; i < 2; i++){

        leds[i] = CRGB(0, 0, 0);

      }

      sqTrack = 0;

      FastLED.show();

    }

    if(rBump && rBumpTrack == 0 && onlyOne == 0){

      rBumpTrack = 1;

      lBumpTrack = 0;

      onlyOne = 1;

    }else if(rBump && rBumpTrack == 1){

      if (sqTrack == 0){

        shut();

      }else{

        head();

      }

      if(crTrack == 0){

        brakeReg();

      }else{

        brakeAct();

      }

      rBumpTrack = 0;

      onlyOne = 0;

    }

    if(lBump && lBumpTrack == 0 && onlyOne == 0){

      lBumpTrack = 1;

      rBumpTrack = 0;

      onlyOne = 1;

    }else if(lBump && lBumpTrack == 1){

      if (sqTrack == 0){

        shut();

      }else{

        head();

      }

      if(crTrack == 0){

        brakeReg();

      }else{

        brakeAct();

      }

      lBumpTrack = 0;

      onlyOne = 0;

    }

    LTSbuttons = millis();

  }

}

void loop(){

  steer();

  drive();

  sound();

  lights();

}