SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2320

PROTOTIP VR APLIKACIJE ZA VOĐENJE PROJEKTA SA SKLOPOVSKOM KOMPONENTOM

Filip Matijević

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2320

PROTOTIP VR APLIKACIJE ZA VOĐENJE PROJEKTA SA SKLOPOVSKOM KOMPONENTOM

Filip Matijević

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2320

Pristupnik:	**Filip Matijević (0036474351)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Željka Mihajlović

Zadatak: **Prototip VR aplikacije za vođenje projekta sa sklopovskom komponentom**

Opis zadatka:

Proučiti potrebne sklopovske komponente potrebne za razvoj uređaja koji će služiti za navigaciju u VR okruženju. Za prikaz VR okruženja koristiti mobilni uređaj te povezati s razvijenim uređajem za navigaciju. Načiniti aplikaciju za vođenje projekata koja će raditi u VR okruženju s razvijenim prototipom uređaja za navigaciju. Načiniti ocjenu rezultata i implementiranih komponenti. Izraditi odgovarajući programski proizvod. Koristiti grafički pogon Unity te programski jezik C#. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

SADRŽAJ

UVOD

Ovaj diplomski rad temelji se na istraživačkom pristupu nekoliko zasebnih komponenti koje se u demonstracijske svrhe povezuju u jednu cjelinu. Rad obuhvaća područja računalne grafike, razvoja interaktivnih aplikacija i video igara, modeliranja trodimenzionalnih predmeta u svrhu implementacije u igre, modeliranje trodimenzionalnih modela u svrhu printanja fizičkog predmeta, implementacija komponenti koristeći C++ i C# programski jezik te proučavanje i izvedbu povezivanja sklopovskih komponenata.

Razvojni proces je strukturiran na način da su se paralelno mogle implementirati različite cjeline, ovisno o dostupnom vremenu ili sklopovlju u danom trenutku.

Fokus rada je stvaranje sklopovskog kontrolera koji se može jednostavno povezati za računalo ili mobitel preko aplikacije načinjene u Unity3D pokretaču.

U radu je detaljno opisan razvojni proces pojedine stavke i njihova međusobna komunikacija. Sav kod je dostupan za izmjenu, iako je napisan na način da je prilagođen za nadogradnje po potrebi.

KONTROLER ZA VIRTUALNU STVARNOST

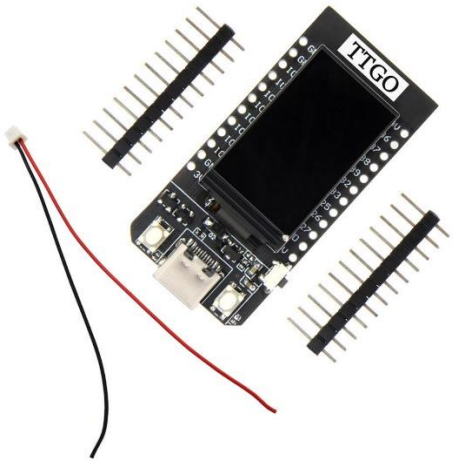
Mobilni VR kontroler napravljen je od niza komponenata koje prikupljaju podatke od korisnia koristeći zasebne senzore. Svaka komponenta ima ulogu prikupljanja ulaznih podataka od korisnika kako bi kontroler u cjelini nesmetano izvršavao svoju ulogu. Pokraj ulaznih podataka, kontroler ima mogućnost prikazivanja podataka na svojem zaslonu. Komponente od kojih je kontroler sastavljen su:

- LilyGO TTGO T-display ESP32 mikrokontroler rezolucije 135 x 240 px (1.14 in)
- Taster
- Joystick (dva potenciometra sa dodatnim tasterom)
- Baterija 850mah
- Sigurnosna sklopka za bateriju
- ICM-20948 IMU senzora

Slika 1 Kontroler za virtualnu stvarnost

MIKROKONTROLER

ESP32 je serija jeftinih sustava mikrokontrolera niske potrošnje s integriranim Wi-Fi i dvostrukim načinom rada Bluetooth modula. Bluetooth je moguće koristiti u modu BT (klasičan bluetooth) ili BLE (Bluetooth Low Energy). Mikrokontroler sadrži Xtensa LX6 32-bitni mikroprocesor koji rad može obavljati na jednoj ili dvije jezgre frekvencijom od 160 ili 240 MHz. Mikrokontroler također posjeduje 520 KiB SRAM memorije.



Slika 2 LILYGO T-Display ESP32

Postoje različite verzije mikrokontrolera koje dolaze s dodatnim komponentama, kao sto su baterije, zasloni u boji, e-ink zasloni, kamere i sl. Verzija mikrokontrolera koja je korištena u radu dolazi sa zaslonom u boji rezolucije 135 x 240 px s prikazom 65000 boja [9]. Kontroler također dolazi s tri tipkala od kojih su sva slobodna za razvoj, a treći sluzi za resetiranje. Razvojnu pločicu ESP32 moguće je, osim preko USB sučelja, napajati i preko punjive litij-polimer baterije. Zaštita od pretpražnjenja te kontrola ciklusa punjenja implementirana je sklopovski na samoj pločici. Od niza dostupnih pinova, za ostvarenje ovog projekta karakteristični su : GPIO, ADC, SDA, SCL i GND.

- GPIO (General-purpose input/output) je ulazno izlazni pin opće namjene kojeg je moguće koristiti po potrebi, ovisno o komponenti koja se implementira.
- ADC (Analogno Digitalni Pretvarač eng. Analog to Digital Converter) je pin koji analogne podatke pretvara u digitalne cime pruža sučelje za korištenje podataka iz stvarnog svijeta.
- SDA (Serial Data) zajedno sa SCL (Serial Clock) omogućuju povezivanje komponenti na razvojnu pločicu kao sto su navigacijski sklopovi ili inercijski senzori.
- GND je pin kojim se komponente uzemljuju.

Za prijenos podataka sa računala na razvojnu pločicu, kao i za napajanje bez baterije se koristi USB-C sučelje.

ICM-20948 INERCIJSKI SENZOR

IMU (Inertial Measurement Unit) je elektronički sklop koji očitava i bilježi sile koje djeluju na tijelo, njegovu kutnu brzinu i orijentaciju u prostoru koristeći kombinaciju akcelerometra, žiroskopa i magnetometra. Navedeni senzor obično se koristi za očitanje stanja letjelica i u navigacijskim sustavima. Najveća mana ovog senzora u navigaciji i praćenju stanja tijela je nakupljanjem pogreške.

Zbog integracije akceleracije tijela, nakuplja se konstantna pogreška koja uzrokuje linearnu pogrešku u brzini, te kvadratnu pogrešku u poziciji.

Zbog ovakvog ponašanja dolazi do “drifta”, sto je zapravo razlika između izračunatog i stvarnog stanja tijela.

Konstantna pogreška očitanja žiroskopa rezultira kvadratnom pogreškom u brzini i kubnom pogreškom u poziciji.

Korištenjem vise senzora istovremeno, moguće je ispravljati ovu pogrešku. S obzirom na to da ne postoji referentna točka izvan kontrolera, nemoguće je ispraviti pogrešku u poziciji kontrolera. Popularni sustavi kao sto su HTC VIVE i Oculus Rift koriste vanjske infracrvene senzore kako bi odrediti poziciju kontrolera u lokalnom koordinatnom sustavu.

Slika 3 HTC Vive senzor pokreta

Za pouzdanu rotaciju tijela potrebno je izračunati promjene u *roll, pitch i yaw* osima. *Roll i pitch* osi su izračunljive iz očitanja akcelerometra u stanju kada tijelo miruje. U idealnom slučaju, kada su dvije komponente akceleracijskog vektora paralelne za Zemljinom površinom, treća će očitavati vrijednost Zemljine gravitacijske akceleracije, dok preostale dvije komponente vektora će biti približno 0. *Yaw* os nije moguće pouzdano izračunati iz vrijednosti akcelerometra jer rotacija oko osi okomite na zemljinu površinu neće bilježiti promjene u iznosu akceleracije.

Za proširenje ovog sustava koristi se žiroskop koji bilježi kutnu brzinu tijela po pojedinim osima. To omogućava izračun *Yaw* komponente koju je potrebno ispraviti zbog spomenutog nakupljanja pogreške.

Kao sto je spomenutu prije, za precizno očitanje stanja tijela, potrebna je vanjska referentna točka. U ovome slučaju pogrešku rotacije ispravlja magnetometar koji pouzdano može izračunati kut kojeg zatvaraju osi tijela sa silnicama Zemljinog magnetnog polja.

Slika 4 ICM20948

ICM20948 sadrži akcelerometar, žiroskop, magnetometar i senzor topline iz kojih se na unutrašnjem procesoru računaju podaci potrebni za rad kontrolera.

KOMANDNA PALICA I TIPKALA

Komandna palica (engl. Joystick) je komponenta koja se koristi za dohvaćanje dvodimenzionalnog nagiba sklopa. To je jednostavan sklop sastavljen od dva potenciometra. Potenciometar je varijabilan otpornik koji nam omogućuje izmjenu očitanog napona nakon njegova pomaka cime se ponaša kao senzor. Pokraj dva potenciometra, komandna palica sadrži i tipkalo koje zatvara strujni krug pritiskom na sklop.

Slika 5 Komandna palica

Vrijednosti napona očitavaju se kao analogni podaci, zbog čeka su povezani na analogne pinove, dok je tipkalo digitalan podataka koji nam govori da li je strujni krug otvoren ili ne. Sklop se povezuje na napon u intervalu od 2V do 6V, a očitanje analognih senzora je u intervalu od 0 do 4095, gdje je stanje mirovanja otprilike na sredini tog intervala.

Slika 6 OMRON B3F tipkalo

Kao sto je prije navedeno, koristi se i dodatno tipkalo kao ulazno sučelje. To je jednostavna sklopka koja zatvara strujni krug i omogućuje izvođenje događaja u trenutku rastućeg ili padajućeg brida. Tipkala se povezuju na digitalne pinove jer je njihova vrijednost 0 ili 1. U ovoj implementaciji korišteno je OMRON B3F Tipkalo s kapicom za zaštitu sklopa

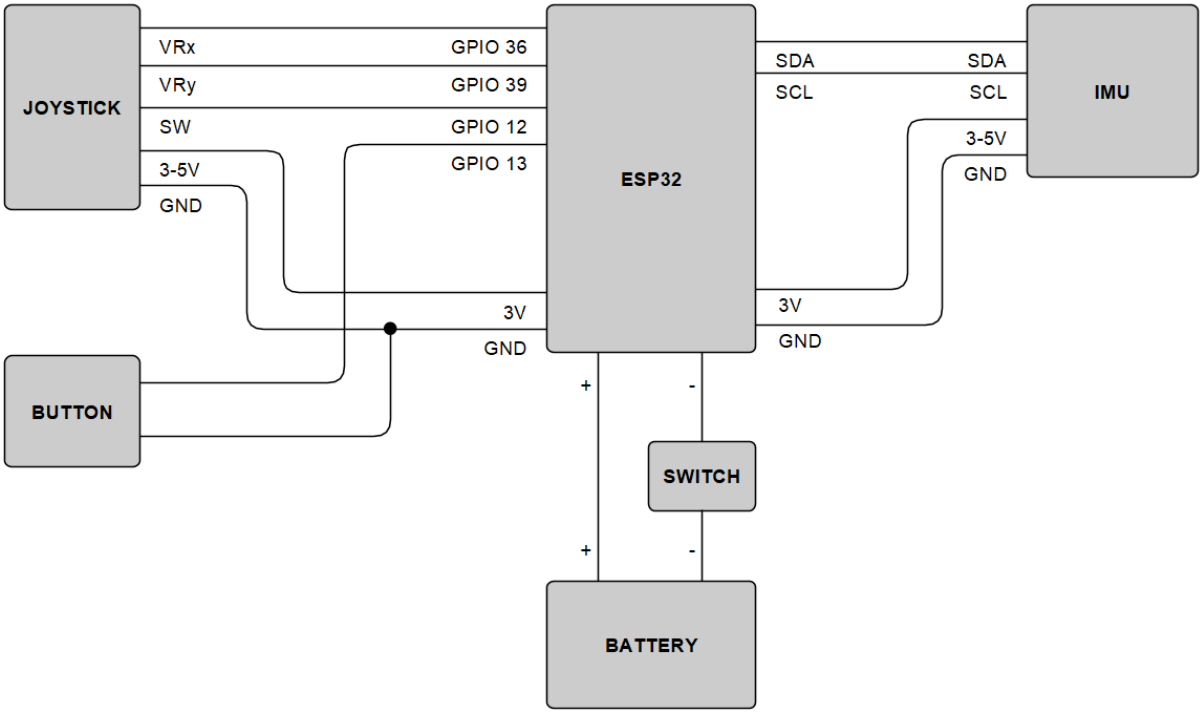
BATERIJA I SIGURNOSNA SKLOPKA

Kontroler je moguće napajati putem USB sučelja i punjive baterije (akumulatora). Ona je na sklop povezana preko dediciranog sučelja koje je tvornički ugrađeno na razvojnu pločicu. Razvojna pločica radi u intervalu od 2.7V do 4.2V.

Slika 7 850 mAh baterija

Implementirana je sigurnosna sklopka između razvojne pločice i baterije kako bi se mogla odspojiti dok se ne koristi.

SKICA POVEZIVANJA KOMPONENTI

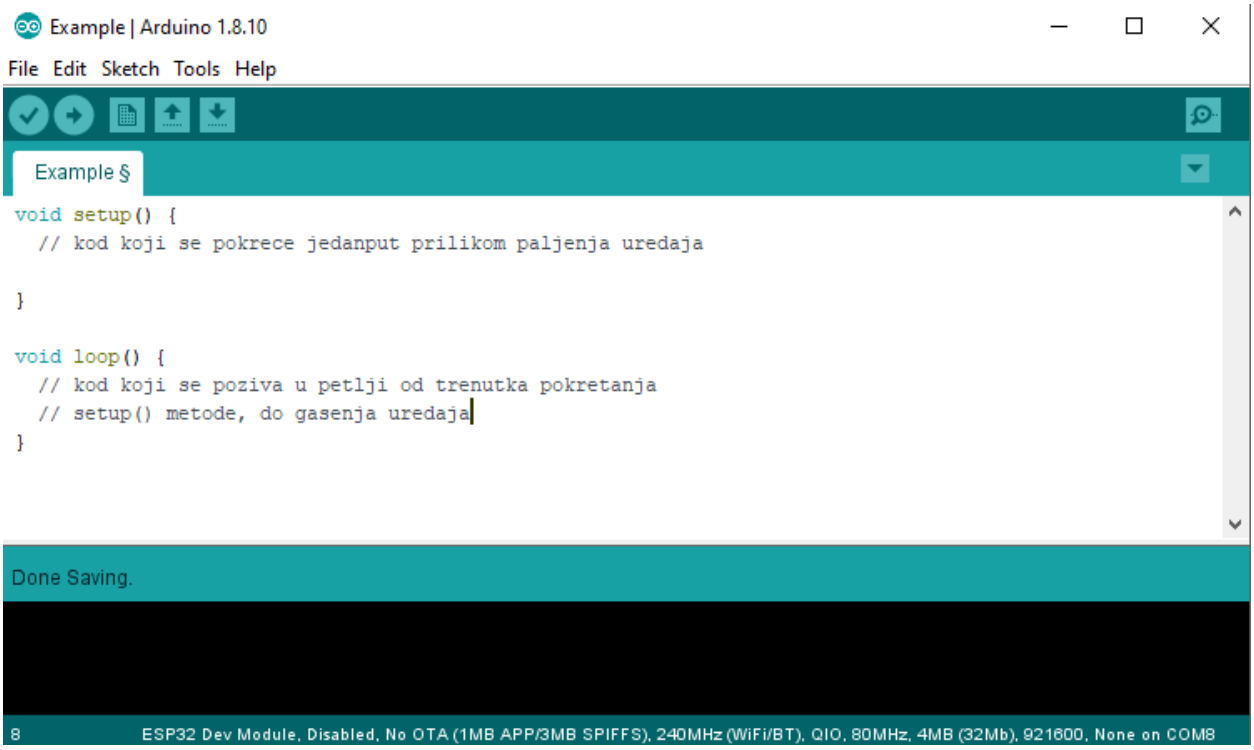


Slika 8 Prikaz povezivanja sklopova

Bitno za napomenuti da nemaju svi GPIO pinovi sposobnost čitanja digitalnih signala pa je potrebno odabrati one koji to mogu. Dodatno, potrebno je definirani način čitanja signala koji je opisan o implementacijskoj cjelini rada.

PROGRAMSKO OKRUŽENJE ZA RAZVOJ KONTROLERA

Sklopovi kontrolera su povezani i implementirani u Arduino okruženju koristeći Visual Studio Code [6]. Alternativa VS Code alata jest Arduino [4] uz nekoliko mana. Arduino okruženje generira *.ino datoteke, ima donekle nepregledno korisničko sučelje, ali pruža veću kontrolu nad sklopovima priključenim za računalo.

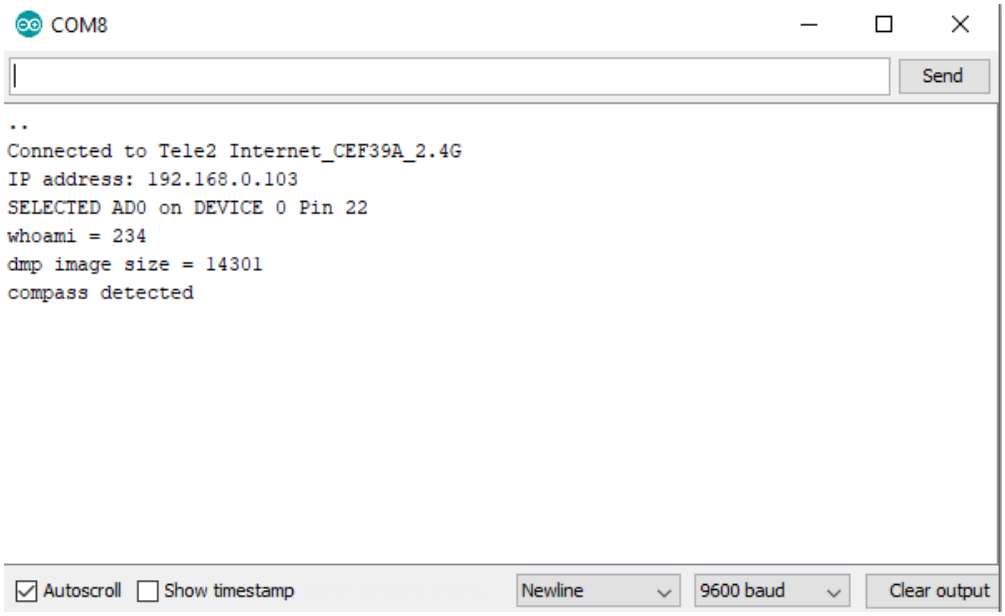


Slika 9 Arduino razvojno okruzenje

Karakteristične metode za razvoj Arduino sustava su setup() i loop(). Setup metoda se poziva prilikom paljenja uređaja, dok se loop() ponavlja u petlji od trenutka neposredno nakon završetka poziva setup metode, sve do gašenja uređaja. Ako postoje vrijednosti koje se postavljaju ovisno o vanjskim parametrima, bitno je da sustav ima mogućnost nastavljanja rada nakon nekog prekida bez da se mora ponovo pokrenuti. Za to je najbolje implementirati slušače koji vode evidenciju ponaša li se sustav kako bi trebao ovisno o trenutnom stanju.

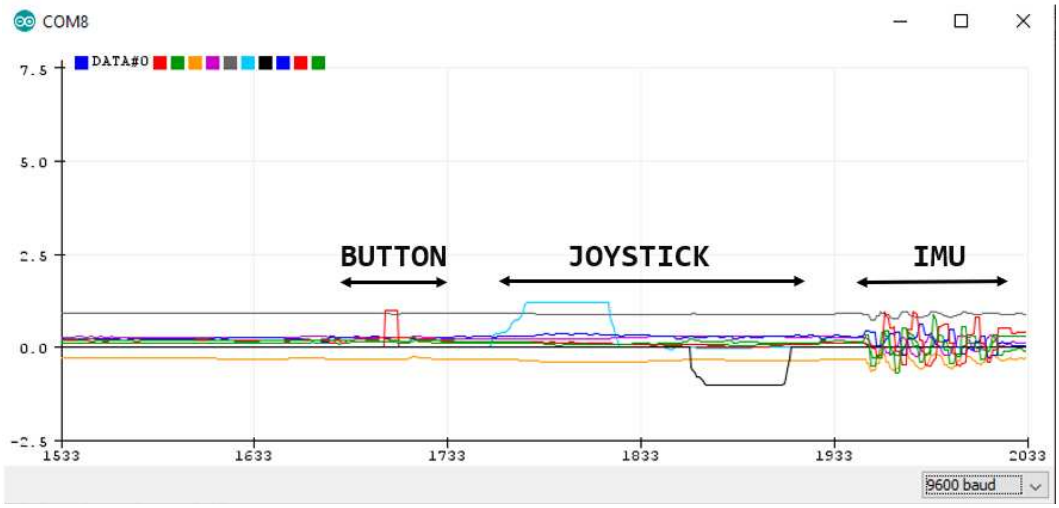
U postavkama okruženja potrebno je odabrati karakteristike kojima se program snima na razvojnu pločicu, port na koji je spojen i frekvenciju kojom se program snima. Prilikom svakog snimanja programa, računalo resetira mikrokontroler kako bi se izvela *setup* metoda.

PREGLED PODATAKA



Slika 10 Serial monitor

Za praćenje rada programa koristi se *serial monitor*. Podaci se ispisuju sa određenom brzinom prijenosa (engl. Baud rate) koja se definira u setup metodi. Ta postavka je također vidljiva na sučelju u donjem desnom kutu prozora.

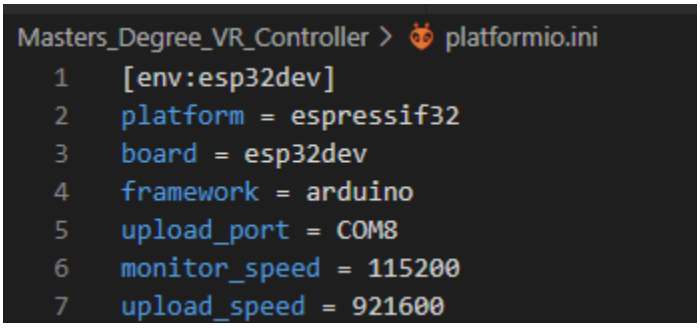


Slika 11 Serial plotter

Pokraj *Serial monitora*, postoji i *Serial plotter* koji grafički prikazuje numeričke podatke u grafu ovisnom o vremenu. Ovaj način prikaza podataka je prikladan za vrijednosti koje učestalo dolaze u velikom broju kako bi se provjerila prisustvo šumova i nepravilnosti u podacima. Bitno je da se ti podaci ispisuju kao numeričke vrijednosti, a ne kao tekst odvojen znakovima.

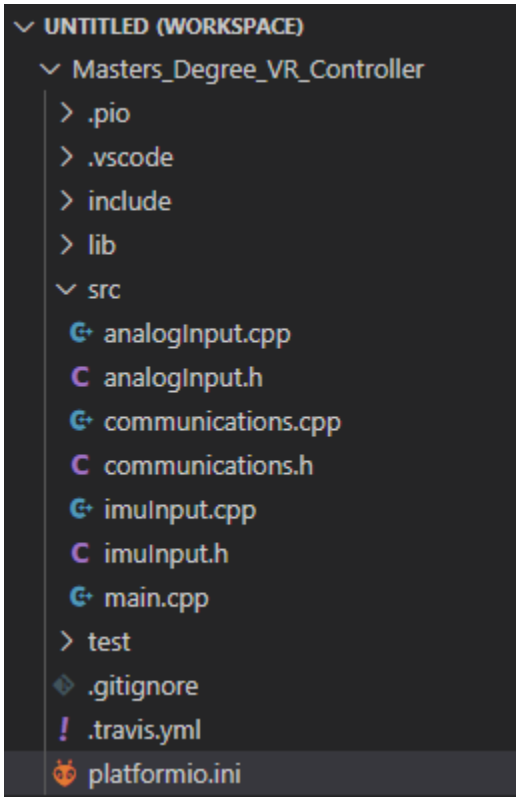
IMPLEMETACIJA UZ KORŠTENJE *VISUAL STUDIO CODE*

Iako gore navedeno Arduino korisničko sučelje je dovoljno za implementaciju manjih programa na razvojnu pločicu, za veće korisno je odvojiti kod na komponente po načelima objektnog programiranja zbog lakih izmjena i uklanjanja pogreška. Visual Studio Code, zajedno sa Platform IO [7] nadogradnjom pruža jednostavan razvoj sustava s detaljnim poveznicama na upozorenja i pogreške u kodu, ako one postoje. Postavke povezivanja računala i kontrolera stvaraju se prilikom otvaranja novog projekta, a dodatne postavke je moguće ručno upisati kroz inicijalizacijsku datoteku platformio.ini



Slika 12 Postavke komunikacije s mikrokontrolerom

Iako sustav sam pronalazi upload_port prilikom prevođenja programa, taj port je nepromjenjiv te ga je korisno definirati u navedenoj datoteci.



Slika 13 Struktura razvojnog okruzenja

Program se izvodi iz datoteke main.cpp koja je ujedno i jedina metoda koja ima setup i loop metode. Ona poziva sve ostale komponente kako bi dohvatila podatke te ih prosljeđuje dalje komunikatoru da ih šalje na računalo ili mobitel.

ČITANJE ANALOGNIH PODATAKA

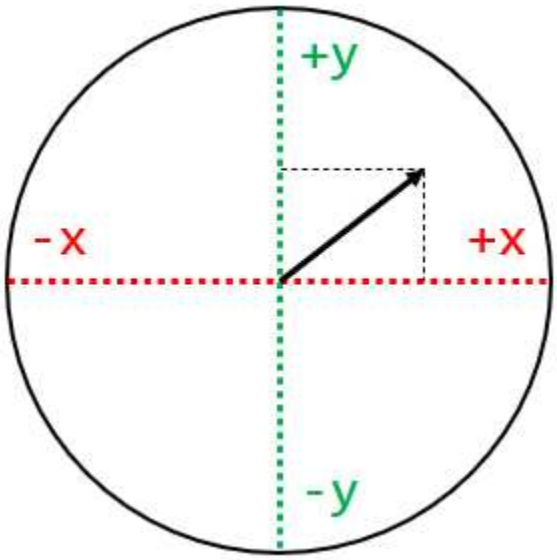
Analogni podaci čitaju se s pinova koristeći metodu analogRead(int PIN_ID) dostupnoj u Arduino biblioteci. Ta metoda vraća vrijednosti u intervalu koji je dostupan specifičnom pinu. Prilikom čitanja vrijednosti ovisne o potenciometrima u palici, ta vrijednost je u intervalu od 0 do 4095.

Eksperimentalnom metodom je određeno stanje mirovanja koje za *x* os iznosi 1870, a za *y* os 1820. To nije srednja vrijednost intervala, što je očekivano zbog nepravilnosti u sklopovskim komponentama palice.

```
void ValidateAnalogInput(){
    xAxis = analogRead(36) - 1870;
    yAxis = analogRead(39) - 1820;
```

Slika 14 Citanje analognih podataka

Nakon čitanja podataka, njihova vrijednost se pomiče za vrijednost mirovanja kako bi te vrijednosti mogle biti upisane u dvodimenzionalni vektor koji je prikladan podatak za daljnju implementaciju.



Slika 15 Podaci za primjenu u aplikaciji

Vektor nije potrebno ograničiti na područje kružnice jer potenciometri ne dozvoljavaju da X i Y os istovremeno budu na maksimalnoj/minimalnoj vrijednosti.

ČITANJE DIGITALNIH PODATAKA

Na kontroler su povezana dva tipkala s kojih se čita digitalni podatak koji opisuje da li tipkala zatvaraju strujni krug ili ne. Kod ovakvih podataka potrebno je definirati način na koji se podaci čitaju.

```
void InitPins(){
    pinMode(triggerPin, INPUT_PULLUP);
    pinMode(joystickPin, INPUT_PULLUP);
}
```

Slika 16 Inicjalizacija tipkala

S navedenim kodom pinovi su postavljeni da vraćaju 0 kada su pritisnuta, a 1 kada nisu. Metoda *pinMode* također je dostupna u Arduino biblioteci.

```
bool GetTriggerState(){
    return digitalRead(triggerPin) == 0;
}

bool GetJoystickState(){
    return digitalRead(joystickPin) == 0;
}
```

Slika 17 Citanje stanja tipkala

Kao i prije navedene metode, digitalRead je metoda dostupna u Arduino biblioteci koja vraća trenutno stanje pina.

ČITANJE PODATAKA S IMU SENZORA

ICM20948 izvršava izračune svih potrebnih podataka na vlastitom procesoru. Implementacija toga dostupna je na GIT repozitoriju [8]. Neki od podataka su

- Nekalibrirani podaci magnetometra, žiroskopa i akcelerometra
- Kalibrirani podaci magnetometra, žiroskopa i akcelerometra
- Neobrađeni podaci akcelerometra i žiroskopa
- Geomagnetna rotacija
- Kalibrirana rotacija

Stavke koje se koriste za ostvarenje rada kontrolera su

- Geomagnetna rotacija
- Kalibrirani podaci akcelerometra

Dostupna implementacija je proširena na način da se ponaša kao komponenta, a ne kao vodeća datoteka sa *setup i loop* metodama. Izmjene nad originalnom metodom su:

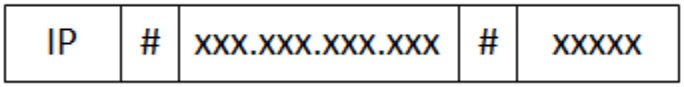
- Uklanjane *setup* i *loop* metode te njihovo refaktoriranje u metode koje čitaju podatke iz registara sa sklopa
- Dodavanje metode za osvježavanje podataka na senzoru
- Dodavanje metode za dohvat izračunatih podataka
- Dodavanje zaglavlja (header file) za pristup podacima iz glavnog programa.

```
void GetRotationData(float* data){
  data[0] = QuatX;
  data[1] = QuatY;
  data[2] = QuatZ;
  data[3] = QuatW;
}

void GetAcceleration(float* data){
  data[0] = AccX;
  data[1] = AccY;
  data[2] = AccZ;
}

void TriggerData(){
  int rv = inv_icm20948_poll_sensor(&icm_device, (void *)0, build_sensor_event_data);
}
```

Slika 18 Dohvat podataka IMU senzora

KOMUNIKACIJSKA KOMPONENTA

Za povezivanje razvojne pločice na lokalnu mrežu odgovorna je komunikacijska komponenta. Ona u trenutku paljenja pretražuje pristupne točke u dometu i povezuje se na onu za koju ima odgovarajuću lozinku. Ako povezivanje ne uspije, pločica ponovo poziva metodu za spajanje, sve dok se ne uspostavi veza. Nakon uspješno uspostavljene veze, komunikacijska komponenta počinje slušati na UDP *broadcast* poruke na preddefiniranom portu kako bi čula javljanje računala i pokrenula proces povezivanja na aplikaciju.



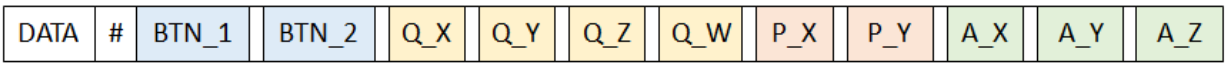
Slika 19 Komuikacijska poruka za uspostavu veze

Svaka komunikacijska poruka koja sadrži podatke podijeljena je na zaglavlje i tijelo. Zaglavlje je karakteristična riječ po kojoj sustavi znaju o čemu se radi. Zbog načina na koji se UDP protokol ponaša, nije potrebno znati koliko je poruka dugačka.

Kada razvojna pločica pročita navedenu poruku za uspostavu komunikacije, ona prestaje osluškivati *broadcast* te šalje odgovor dobrodošlice koja ne sadrži tijelo.

Slika 20 Poruka za potvrdu veze

Nakon slanja poruke dobrodošlice, razvojna pločica ulazi u stanje rada te prikuplja sve potrebne podatke i šalje ih aplikaciji. Poruke su male veličine i dovoljno učestale da bi se gubitak paketa mogao zanemariti. Bilo kakav zastoj u porukama koji traje manje od razumnog vremena se može programski kompenzirati na strani aplikacije.

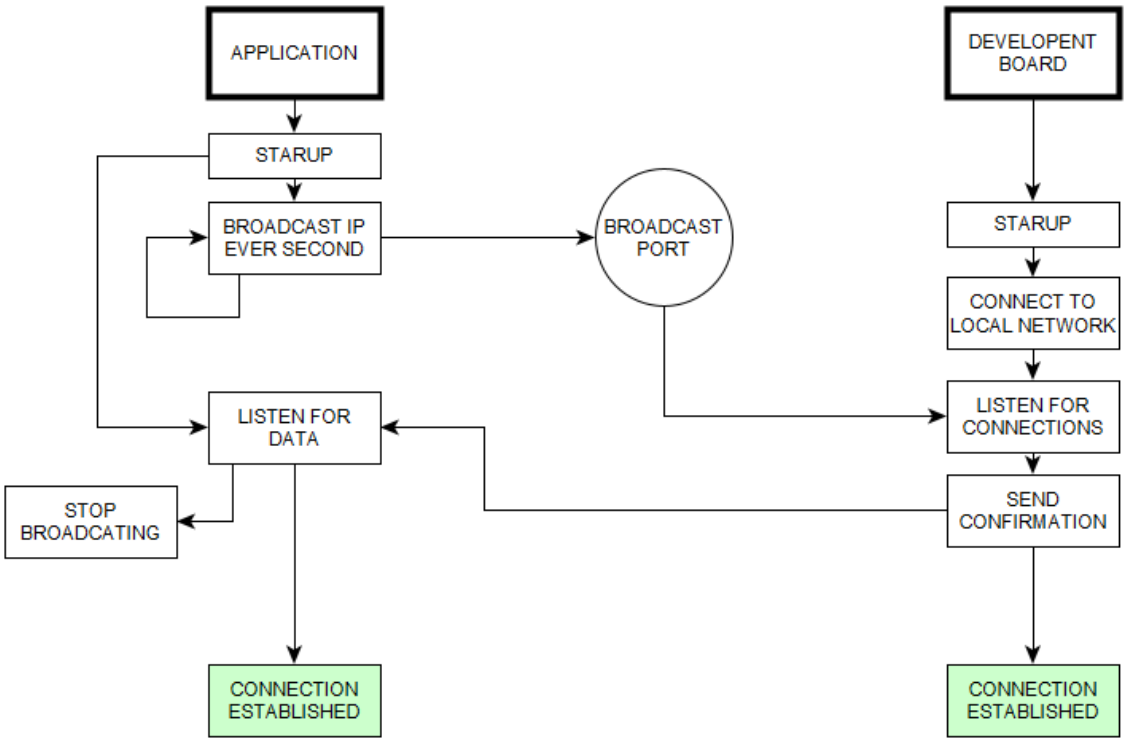


Slika 21 Podatkovna poruka za rad aplikacije

U radnom stanju, razvojna pločica šalje poruke sa zaglavljem “DATA”, dok u tijelu ima sve ostale podatke odvojene razmakom. Ti podaci opisuju:

- Stanje tipkala na poleđini kontrolera
- Stanje tipkala na palici
- Rotacijski kvaternion IMU senzora
- Stanja potenciometara na palici
- Akceleracijski vektor IMU senzora

Ovakav način komunikacije omogućava da se razvojna pločica poveze na bilo koju platformu koja ima pristup lokalnoj mreži. Mane ove komunikacije su iznimno lagano prisluškivanje podataka i njihovo presretanje. Moguće je i lažiranje podataka sa strane uljeza u lokalnoj mreži koje bi se moglo izbjeći kriptiranjem podataka, ali bi to dodatno opteretilo oba sudionika u komunikaciji.

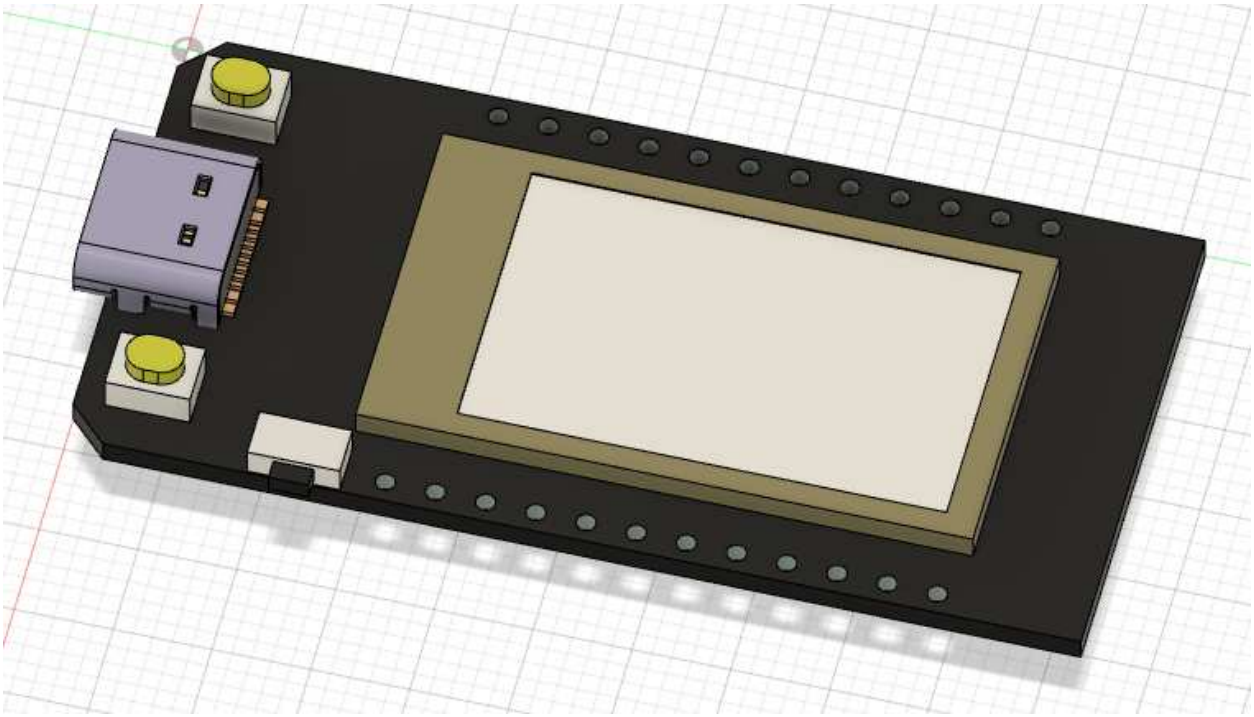


Slika 22 Dijagram povezivanja

KUĆIŠTE KONTROLERA

Za realizaciju kućišta kontrolera korišten je alat Autodesk Fusion 360 [3]. Alat se koristi za 3D modeliranje predmeta u svrhu prototipiranja, stvaranja realnih prikaza predmeta ili modeliranja objekata za printanje. Zbog povećane popularnosti 3D printera i izrade prototipova komponenti, postoji velika baza 3D modela komponenti koje su u prodaju. Tako su i komponente koriste u izradi ovog kontrolera dostupne na stranicama www.grabcad.com. Sve komponente izmodelirane su po mjeri i spremne su za korištenje u Autodesk Fusionu.

Osnovna komponenta od koje je započeto modeliranje je razvojna pločica. Nedostatak ove pločice je manjak proreza za pričvršćivanje na kućište zbog čega su se morali modelirati dodatni držači.

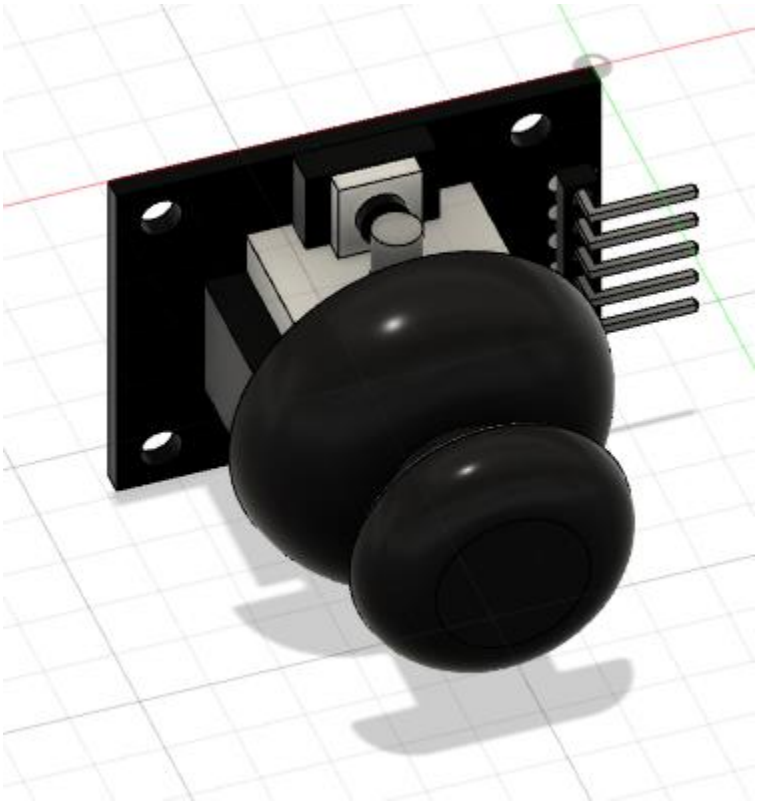


Slika 23 3D Model razvojne pločice

Prorezi se nisu mogli dodatno napraviti jer je površina pločice u potpunosti iskorištena. Kućište je dizajnirano tako da pločicu na mjestu drži kućište koje stvara pritisak na USB C priključak vidljiv na lijevoj strani slike.

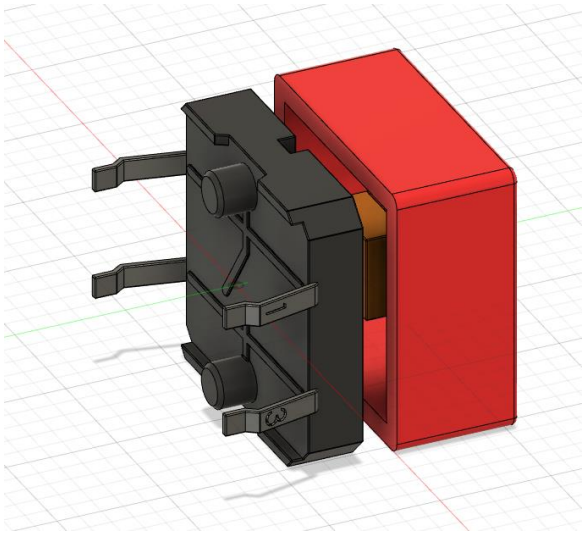
Za razvojnu pločicu je izrezan prozor za zaslon kako bi se mogao koristiti za bitne informacije o podacima koje nudi pločica.

Isto kao i model razvojne pločice, model kontrolne palice je također dostupan na navedenim stranicama. Za postavljanje ove komponente u kućišta bilo je potrebno izračunati pomak jer gljiva kontrolera ne nalazi se na središtu komponente.



Slika 24 Model upravljačke gljive

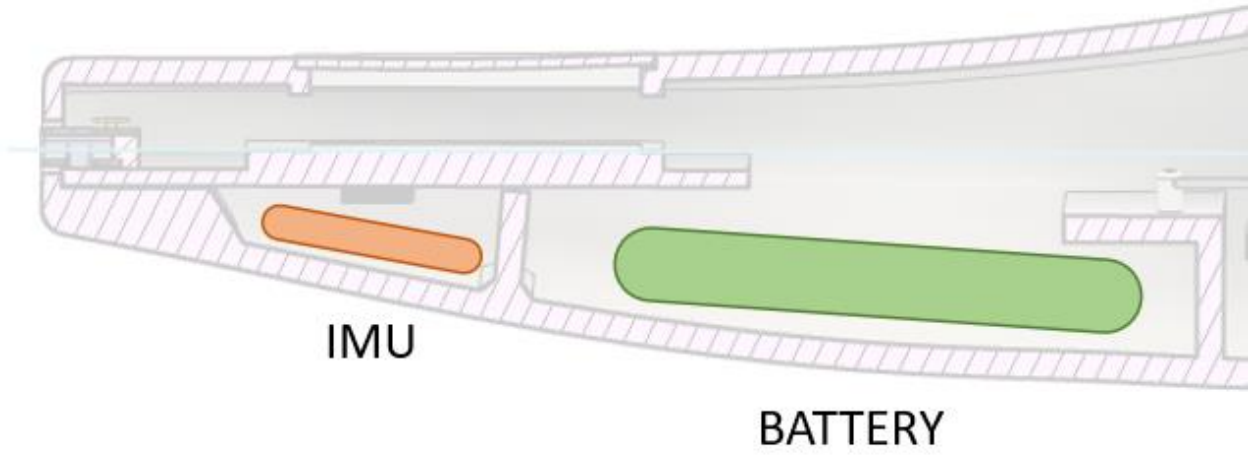
Ova komponenta dolazi sa prorezima kojima se mogla jednostavno učvrstiti za tijelo kućišta koristeći 4 kratka vijka.



Slika 25 Model tipkala

Tipkalo je ukomponirano u kontroler koristeći vruće ljepilo i izolator zbog njegove pozicije. Tipkalo se nalazi odmah ispod potenciometara na koje se oslanja sto joj daje stabilnost. Postoji mogućnost modeliranja drugačije kapice za tipkalo kako bi se ljepše uklopilo u estetiku kontrolera.

Za ugradnju baterije i inercijskog senzora koristio se *Fusion360* za modeliranje približnih oblika sklopova jer oni nisu dostupni na webu. Korišteni su jednostavni poligoni koji opisuju volumen sklopa.



Slika 26 Pozicije baterije i inercijskog senzora

Inercijski senzor je također došao s prorezima za vijke kako bi se osigurala njegova pozicija u kućštu. Čak i mali pomaci u senzoru bi mogli prouzročiti nepravilnosti u radi. Senzor je dodatno izoliran prije ugradnje u kućište zbog njegovog kontakta s razvojnom pločicom.

Prilikom modeliranja kućišta korišten je "*sculpt mode*" za stvaranje zakrivljenih ploha i oblikovanje tijela oko komponenti postavljenih na željena mjesta.

Za izradu kontrolera korišten je Creality CR10 3D Printer. Materijal kontrolera je PLA plastika s visinom sloja od 1 mm, 50% unutrašnjeg popunjenja, brzinom ispisa od 80% normalne brzine i temperaturom od 200 C.

Slika 27 Bočni prikaz

Slika 28 Gornji prikaz

Slika 29 Donji prikaz

Slika 30 Prednji prikaz

RAZVOJNO SUČELJE KONTROLERA ZA VIRTUALNU STVARNOST

Razvojno sučelje kontrolera implementirano je kroz dvije cjeline koristeći C# [5] i Unity3D pokretač [1]. Te cjeline su opisno razdvojene na komunikacijske kontrolere i upravljačku jedinicu. Komunikacijski kontroler odgovoran je za povezivanje kontrolera na aplikaciju, dok se upravljačka jedinica koristi za pretplatu na akcije koje se izvode u karakterističnim trenucima.

Za povezivanje kontrolera na aplikacijsko sučelje korišten je UDP protokol. UDP (engl. User Datagram Protocol) se nalazi na transportnoj razini modela za otvoreno povezivanje sustava (eng. Open Systems Interconnection), te je uz TCP (engl. Transmission Control Protocol) jedan od temeljnih Internet protokola. UDP posjeduje funkcije multipleksiranja i provjeravanja pogreške prilikom prijenosa podataka, ali nema mogućnost provjere primitka poruke. Zbog toga se ovaj protokol koristi u komunikaciji gdje pouzdanost manje bitna od efikasnosti i brzine slanja. Najpopularnija primjena UDP protokola je za prijenos govora u stvarnom vremenu (VoIP).

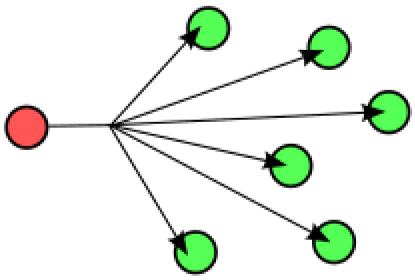
Upravljačka jedinica temelji se na arhitekturi pretplata i pozivanja akcija. Algoritmi koji ulazne podatke procesiraju i zaključuju koji događaj se treba izvršiti, vrte se u petlji u stvarnome vremenu i ne ometaju izvođenje ostatka aplikacije. Za razliku od komunikacijskih komponenti koje se odvijaju na zasebnim dretvama (engl. thread), upravljačka jedinica se vrti na glavnoj dretvi jer komponente Unity3D pokretača se mogu pozivati isključivo iz glavne dretve.

KOMUNIKACIJSKI KONTROLERI

Komunikacijski sustav odvojen je u dvije komponente :

- UDPSender
- UDPReceiver

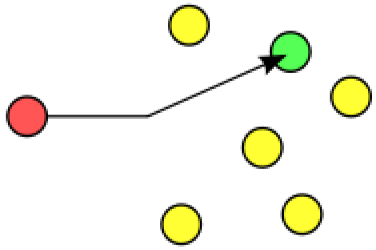
U primjeni, platforma koja pokreće aplikaciju ima ulogu servera tijekom izvođenja. Komunikacija započinje iz UDP Servera koji emitira (engl. Broadcast) svoje podatke lokalnoj mreži [10]. Emitiranje je proces slanja podataka na čitavu lokalnu podmrežu u svrhu otkrivanja kompatibilnih uređaja za izvođenje rada aplikacije. Prilikom emitiranja, ako IP adresa se koristi adresa podmreže 255.255.255.255, a port je definiran u inspektoru razvojnog sučelja ili u kodu Arduino programa.



Slika 31 Emitiranje poruke

Emitiranje se ponavlja beskonačno, svake sekunde dok kompatibilna komponenta ne odgovori na poruku. Oblik poruke koja se emitira je opisan u poglavlju vezanom za implementaciju komunikacijskog protokola na razvojnoj pločici.

U trenutku početka emitiranja poruke, sustav također započinje novu dretvu na kojoj sluša dolazak odgovora. Aplikacija saznaje vlastitu lokalnu IP adresu i na njoj osluškuje usmjerene poruke (engl. Unicast). Usmjerena poruka definirana je mrežnom adresom primatelja. Port osluškivanja je također definiran u inspektoru korisničkog sučelja. Pri početku rada dretve za osluškivanje poruka, sustav stvara pretplatu na akciju koja zaustavlja emitiranje podataka. Ta akcija se poziva kada sustav dobije odgovor na svoju emitiranu poruku. Prilikom primitka takve poruke, ta pretplata se uklanja sve dok nije potrebno ponovo pokrenuti proces otkrivanja.



Slika 32 Komunikacija jedan na jedan

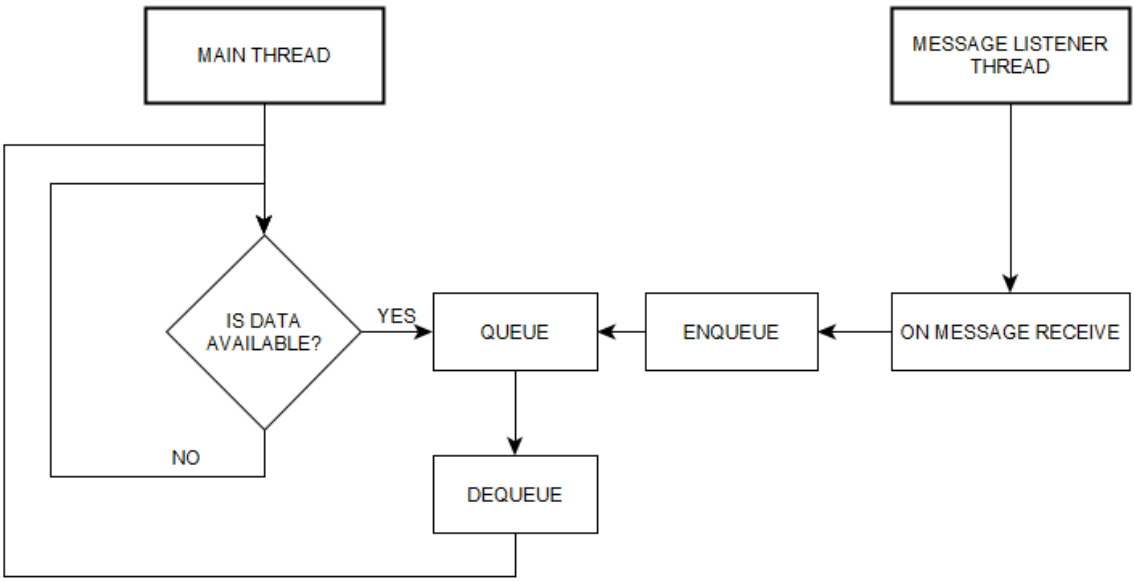
Cikličko emitiranje poruke izvodi se s glavne dretve jer slanje poruke nije blokirajuća operacija i ne narušava brzinu ili efikasnost izvođenja ostatka sustava. Za kontrolu učestalosti poruke, koriste se korutine koje se izvode u istoj okvirnoj stopi (eng. Frame rate) kao i ostatak aplikacije.

Komponenta slušanja poruka izvedena je kroz sporednu dretvu koja neprestano drži aktivnu blokirajuću funkciju čitanja podataka. Kako bi taj podatak bio dostupan ostatku sustava, koristi se red. Red je implementacija povezanih podataka slična listi, uz razliku da se u redovima koriste samo krajnji podaci. Red u ovoj implementaciji sluzi kao dijeljena memorija između dvije dretve pomoću koje one komuniciraju. Prilikom dospijeća poruke, sporedna dretva sadržaj poruke stavlja u red pomoću metode

void Enqueue (string value);

Glavna dretva provjerava stanje reda u svakom okviru, te ako su u njoj dostupni podaci, oni se prosljeđuju algoritmu za parsiranje tih podataka.

Opisana izmjena podataka je asinkrona jer glavna dretva ne zna točan trenutak kada je poruka stigla na računalo, nego zna samo kada je taj podatak dostupan u redu poruka. S obzirom na to da je učestalost slanja poruka s kontrolera na aplikaciju 30 puta u sekundi, a aplikacija se odvija brzinom od 60 okvira po sekundi, ta razlika je zanemariva.



Slika 33 Slijedni dijagram dretvi

Trenutno implementirana zaglavlja poruka mogu biti WELCOME, DATA ili PING. Nakon uzimanja poruke iz reda, ona se prosljeđuje metodi

void ProcessEvent(string value)

Ta metoda na temelju zaglavlja poziva prikladnu akciju. Akcije za zaglavlje uspostavljanja veze i *ping*-a ne primaju argumente, dok akcija zaglavlja “DATA” poziva akciju OnDataReceive s argumentom vrijednosti tijela poruke. Preko te akcije Upravljačka jedinica preuzima podatke i započinje evaluaciju.

UPRAVLJAČKA JEDINICA

Upravljačka jedinica prilikom pokretanja aplikacije postavlja pretplatu na staticu Akciju "OnDataReceived" te preko metode EvaluateData parsira podatke i donosi zaključke.

```
UDPReceive.OnDataReceive += EvaluateData;
```

Podaci se u implementaciji tretiraju kao niz podataka (engl. Stream) i promjene su vidljive između dva skupa podataka. Iz tog niza podataka pozivaju se akcije koje su podijeljene u dvije skupine:

- Akcije ponavljajucih dogadaja
- Jednokratne akcije

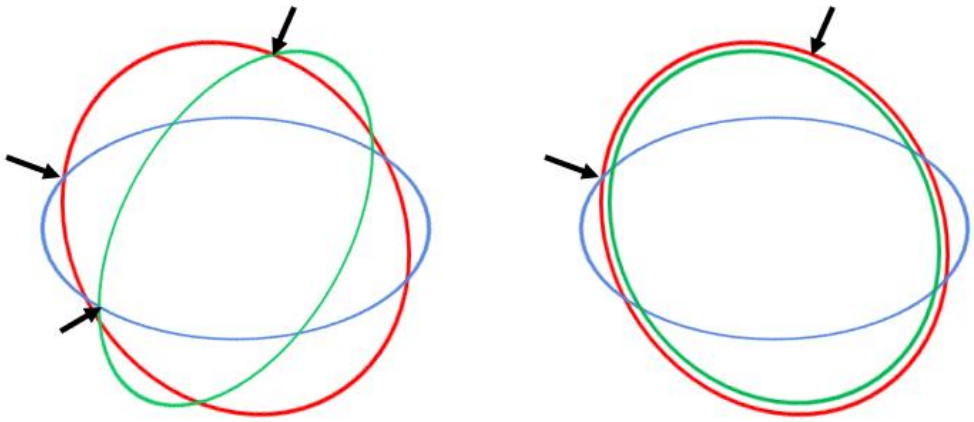
Akcije ponavljajućih događaja su akcije koje jednostavno prosljeđuju informacije aplikaciji nakon sto su razumljive Unity3D pokretaču. Svi podaci dolaze kao tekstualni zapisi, a upravljačka jedinica ih prevodi i pretvara u ispravne strukture koje su iskoristive prilikom stvaranja komponenata igre ili aplikacije.

Najbitnije akcije ponavljajućih događaja su one odgovorne za rotaciju kontrolera i poziciju upravljačke palice. To su akcije:

```
public static Action<Quaternion> OnRotationUpdate;
public static Action<Vector3> OnAccelerationUpdate;
public static Action<Vector2> OnJoystickUpdate;
```

Prilikom primitka podataka, oni se razdvajaju po znaku praznine i parsiraju u ispravne vrijednosti. Za navedene tri akcije sve vrijednosti su decimalne vrijednosti (tip float) koje se spremaju u strukture Quaternion, Vector3 ili Vector2.

Rotacijski kvaternion je struktura koja najbolje opisuje rotaciju tijela. U usporedbi s Eulerovim kutevima, jednostavniji je prilikom kompozicije i u potpunosti uklanja mogućnost nastanka zastoja osi (engl. Gimbal lock)



Slika 34 Zastoj osi i gubitak stupnja slobode

Kvaternion sadrži četiri komponente koje opisuju tijelo. Moguća je transformacija iz kvaterniona u Eulerove kutove bez gubitka informacija.

Parsiranjem vrijednosti stvara se struktura koja se koristi za rotaciju tijela.

```
Quaternion rotationQuat = new Quaternion(float.Parse(qx), float.Parse(qy),
float.Parse(qz), float.Parse(qw));

OnRotationUpdate?.Invoke(rotationQuat);
```

Sve metode pretplaćene na navedenu akciju će se pozvati kada god stigne nova informacija o rotaciji tijela.

Moguće je pretplatiti samo metode koje kao argument prihvaćaju strukturu Quaternion dostupnu u UnityEngine biblioteci.

Informacije o akceleraciji prenose se preko strukture Vector3. Ta informacija se ne koristi u implementaciji demonstracijske aplikacije, ali je dostupna za korištenje ako je ona potrebna. Akceleracijski vektor se prenosi u neobrađenom obliku sa upravljačke pločice te u stanju mirovanja , njegova magnituda je približno jednaka gravitacijskoj akceleraciji Zemlje. Neobrađeni akceleracijski vektor može se koristiti da bi se izračunali kutovi koji se zatvaraju sa površinom zemlje (npr. Ukoliko želimo napraviti libelu koristeći kontroler).

$$\Phi = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \qquad \rho = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right)$$

Rotacija oko osi okomite na zemlji je teza za izračunati zbog nakupljajuće pogreške koja se očitava na *Yaw* kutu.

Priprema podataka obavlja se na sličan način kao i kod kvaterniona i pristupa mu se na isti način s pretplatom prikladne metode.

```
Vector3 accelerationVector = new Vector3(float.Parse(ax), float.Parse(ay),
float.Parse(az));

OnAccelerationUpdate?.Invoke(accelerationVector);
```

Nagib upravljačke palice opisan je sa strukturom Vector2. Osi tog vektora su neovisne i mogu se koristit pojedinačno za implementaciju željenih funkcionalnosti. Nagib upravljačke palice također podržava jednokratne akcije.

```
Vector2 joystickPositin = new Vector2(float.Parse(joyx), float.Parse(joyy));

OnJoystickUpdate?.Invoke(joystickPosition);
```

Jednokratne akcije su one akcije koje se dogode jednom, u nekom karakterističnom trenutku. Obično je riječ o promjeni stanja nekog sklopa ili skupa sklopova. Te akcije imaju mogućnost međusobnog isključivanja i posjeduju prioritete. Osnovna skupina jednokratnih akcija se temelji na promjeni stanja tipkala. One nisu vremenski ovisne te razvojno sučelje ne prati vremenske razlike između dva događaja. Ako je to potrebno, takve funkcije se trebaju implementirati unutar aplikacije. Kao sto postoje jednostavne akcije za promjenu stanja tipkala, tako postoje i kontinuirane akcije koje se pozivaju za vrijeme pritiska. Slična implementacija se nalazi i u UnityEngine biblioteci kod registriranja pritiska tipke na tipkovnici.

```
public static Action OnJoystickButtonUp;
public static Action OnJoystickButtonDown;
public static Action<bool> OnJoystickButtonChange;

public static Action OnTriggerButtonUp;
public static Action OnTriggerButtonDown;
public static Action<bool> OnTriggerButtonChange;
```

Za navedene akcije promjena stanja se biljezi između dvije dospjele poruke. Ako je stanje tipkala u prijašnjem okviru bilo “FALSE” a u trenutnom je “TRUE”, tada se poziva akcija “OnButtonDown”. U suprotnom slučaju, ako je u prijašnjem okviru stanje tipkala bilo “TRUE”, a u trenutnom je “FALSE”, pozvati ce se akcija “OnButtonUp”.

Akcija “OnButtonChange” poziva se kada god se stanje nekog tipkala promjeni, neovisno o vrijednosti. Ove akcije također šalju argument koji opisuje stanje tipkala u koje su došli. Ova implementacija omogućuje implementaciju svih mogućnosti koje pružaju Up i Down akcije uz dodatno kodiranje prilikom izrade aplikacije.

```
public static Action OnJoystickButtonHold;
public static Action OnTriggerButtonHold;
```

Hold akcije, iako su izvedene tipkalom se ponašaju kao kontinuirane akcije i pozivaju se dok korisnik drži pritisnuto tipkalo. Prigodne su za izvedbu funkcija pomicanja predmeta u virtualnom svijetu.

Postoji i skup jednokratnim akcija koje su izvedene pomoću upravljačke palice. Usporedive su sa strelicama na tipkovnici i pozivaju se kada vrijednost komponente vektora palice pređe polovinu puta. Također se poziva akcija kada se palica vrati u neutralno stanje.

```
public static Action OnJoystickLeft;
public static Action OnJoystickRight;
public static Action OnJoystickUp;
public static Action OnJoystickDown;

public static Action OnJoystickIdle;
```

Kombinirani oblik jednokratnih akcija implementiran je na način da prati promjenu oba tipkala. Ova akcija ima svoje prekidne točke, isto kao sto blokira pozive jednostavnih akcija kako ne bi došlo do kolizije.

```
    public static Action OnBothButtonsDown;
    public static Action OnBothButtonsUp;
```

Ove akcije pozivaju se u prvom okviru kada su oba tipkala u istinitom stanju, ili u okviru kada jedno od tipkala vise nije u istinitom stanju, a u prijašnjem okviru su bili. Ulaskom drugog tipkala u istinito stanje, njegova akcija OnButtonDown neće biti pozvana jer "BothButtonsDown" ima veći prioritet.

U aplikaciji ove akcije koriste se pretplatama i ne treba se brinuti o ne pretplaćenim akcijama jer postoji provjera koja izbjegava nedefinirane pozive.

```
JoystickManager.OnTriggerButtonDown += OnButtonDown;

private void OnButtonDown()
{
       if (InteractableGO != null)
        {
              InteractableGO.GetComponent<InteractableComponent>().OnPointerDown(new
              PointerEventData(EventSystem.current));
        }
}
```

Blok koda prikazuje kako se metoda pretplaćuje na akciju kontrolera.

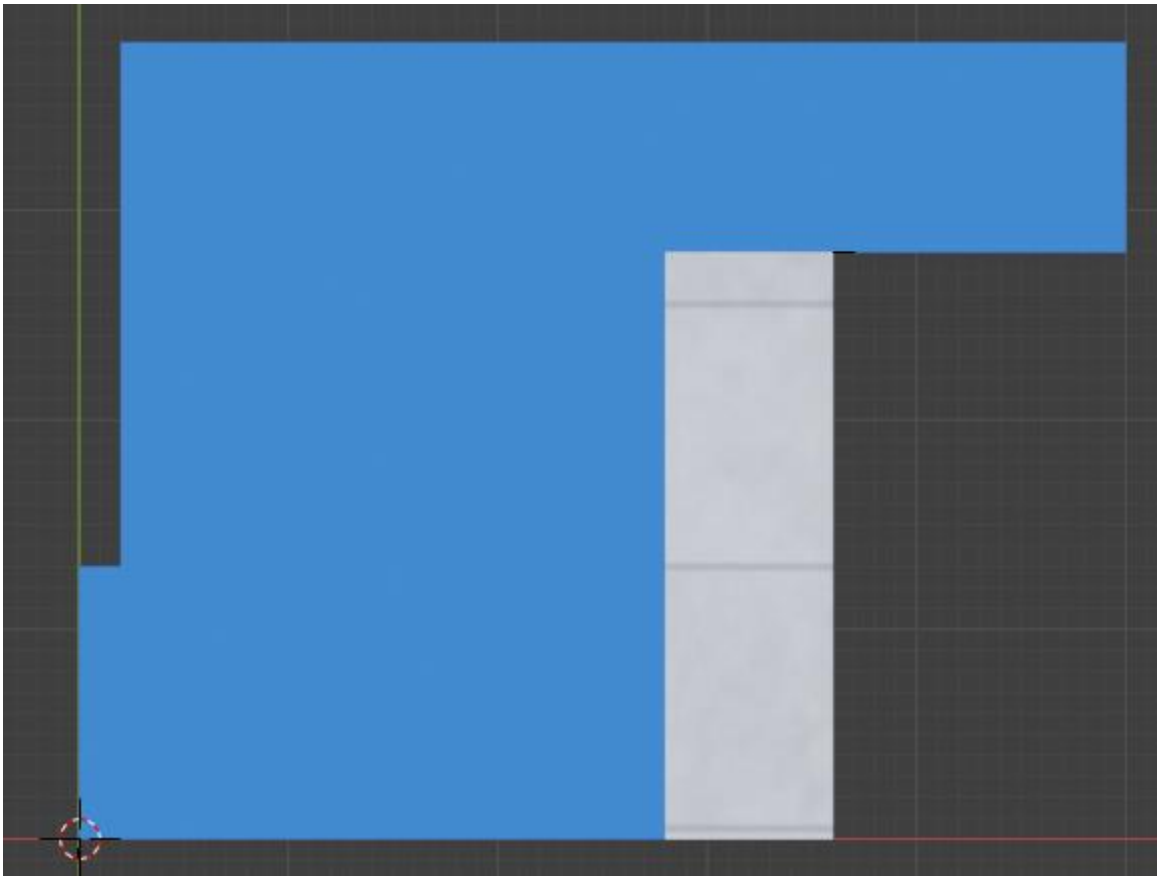
DEMONSTRACIJSKA APLIKACIJA – VR APLIKACIJA ZA VOĐENJE PROJEKTA

Aplikacija za vođenje projekata implementirana je kao demonstracijska aplikacija kako bi se pokazale mogućnosti i korištenje kontrolera u korisničkom i razvojnom okruženju. Ta aplikacija inspirirana je već postojećim desktop aplikacijama za praćenje projekata kao što je Trello.

VIRUALNO OKRUŽENJE

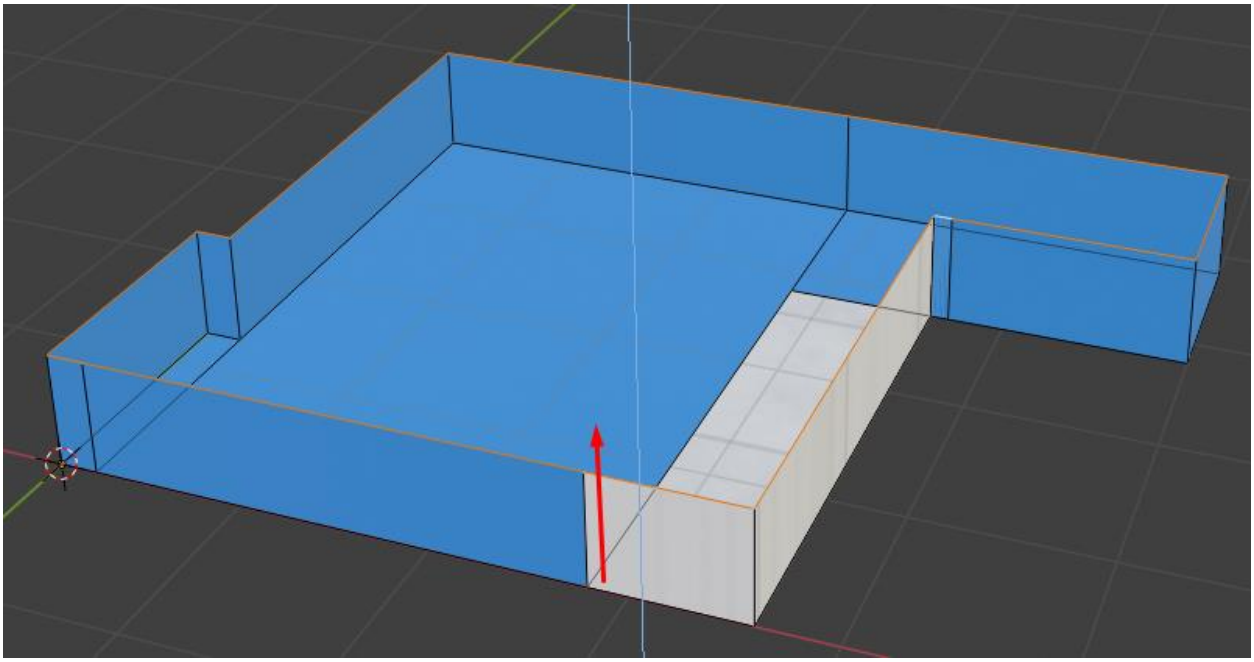
Virtualni prostor u kojem se izvode i demonstriraju mogućnosti kontrolera modelirane su koristeći Blender [2]. Blender je besplatan alat za modeliranje, teksturiranje i animaciju trodimenzionalnih predmeta i većinski se koristi za stvaranje modela koje se koriste u video igrama. Kompatibilan je sa Unity3D pokretačem sto omogućuje pregled modela u pokretaču dok se na tom istom modelu radi u Blenderu. Model je napravljen u stvarnoj mjeri kako bi bio usporediv s modelom kontrolera izvezenog iz Fusion3D aplikacije.

Modeliranje prostorije počinje s tlocrtom u ispravnim mjerama.



Slika 35 Tlocrt prostorije

Nakon crtanja tlocrta, plohe zidova stvaraju se koristeći alat za izvlačenje (engl. extrude). Naknadno je moguće izvučene zidove izbušiti u svrhu modeliranja prozora, ali se taj korak izostavio jer se aplikacija temelji na postavljanju elemenata na zidove cime bi se izgubio interaktivni prostor.

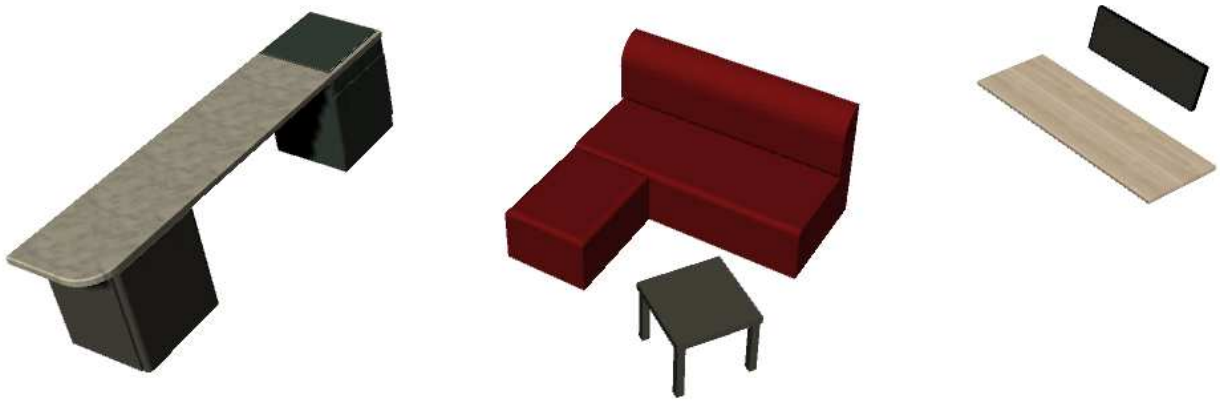


Slika 36 Izvlačenje rubova

Kada se rubovi izvuku na ispravnu razinu, posljednji korak je dodati plohu stropa. Zbog lakšeg razvijanja, normale ploha prostorije uvijek su okrenute prema unutrašnjosti prostorije. Na taj način uvijek je iz vanjske perspektive vidljiva unutrašnjost prostorije, ali za vrijeme izvođenja aplikacije iz perspektive korisnika, te plohe izgledaju normalno.

Čitava prostorija u kojoj se izvodi aplikacija je interaktivni element zbog čega je podijeljena u tri cjeline. Te cjeline su strop, pod i zidovi. Strop nije interaktivan i nije dozvoljeno postavljanje elemenata na njega. Pod sluzi za navigaciju kroz prostor koristeći teleportaciju, a zidovi su slobodni da se koriste za postavljanje sadržaja.

Za nadopunjavanje prostora i vizualnu estetiku modeliran je namještaj uz pomoć Blender aplikacije.



Slika 37 Namještaj korišten u aplikaciji

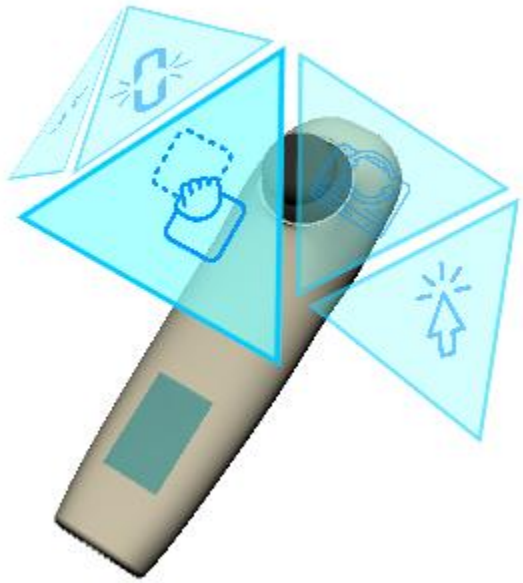
Modeli su napravljeni tako da imaju minimalan broj poligona i točaka sto znatno ubrzava crtanje modela na mobilnim uređajima. Zaobljeni bridovi napravljeni su pomoću prikladnog alata (engl. Bevel tool).

Konačan model interaktivne prostorije vidljiv je na sljedećoj slici.

Slika 38 Virtualno okruženje

KORISNIČKO SUCELJE

Korisničko sučelje za interakciju s aplikacijom izvedeno je koristeći opisani VR kontroler. Čitavo korisničko sučelje implementirano je da gravitira oko virtualne reprezentacije kontrolera zbog lakšeg korištenja.



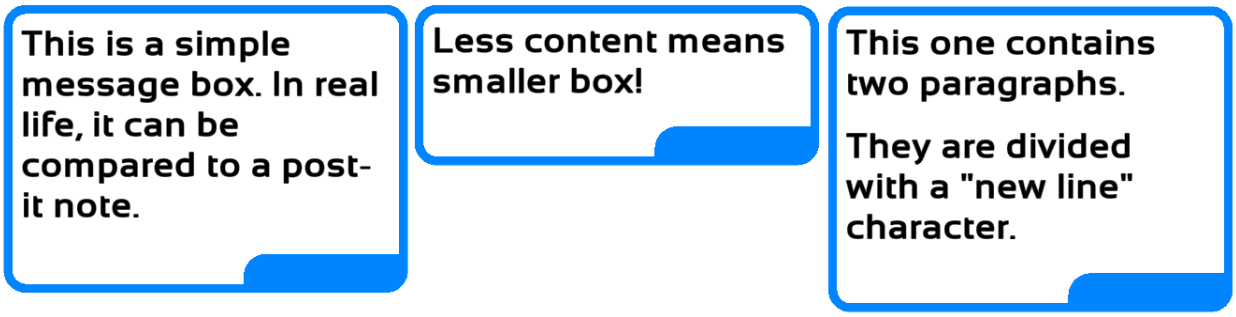
Slika 39 Korisničko sučelje

Korisničko sučelje se poziva tako da se istovremeno pritisnu oba tipkala na kontroleru. Ta akcija otvara glavni izbornik kojim se navigira koristeći pomake na upravljačkoj palici. Horizontalni pomaci okreću izbornik oko kontrolera i fokusiraju jednu od ikonica. Stavka se odabire pritiskom tipkala na poleđini kontrolera. Ta akcija također zatvara izbornik.

KOMPONENTE APLIKACIJE

Korisnik si interaktivne elemente može po želji rasporediti i postaviti po površinama zidova unutar virtualne prostorije. Ti elementi imaju međusobne interakcije i mogu se pretvarati u kompleksnije stavke prilikom korištenja.

Osnovni element aplikacije zove se “Jednostavna poruka” (engl. Simple message). Ta komponenta sluzi da pohranjuje jednostavan tekst i prikazuje ga na zidu. U ovoj verziji aplikacije dostupan je demonstracijski skup poruka, ali nove poruke je također moguće primiti pomoću UDP poruke ako se razvije dodatna aplikacija. Unos tekstualnih zapisa u VR aplikaciji je neprecizan i teško ostvariv a da se ne naruši korisničko iskustvo prilikom upravljanja aplikacijom.

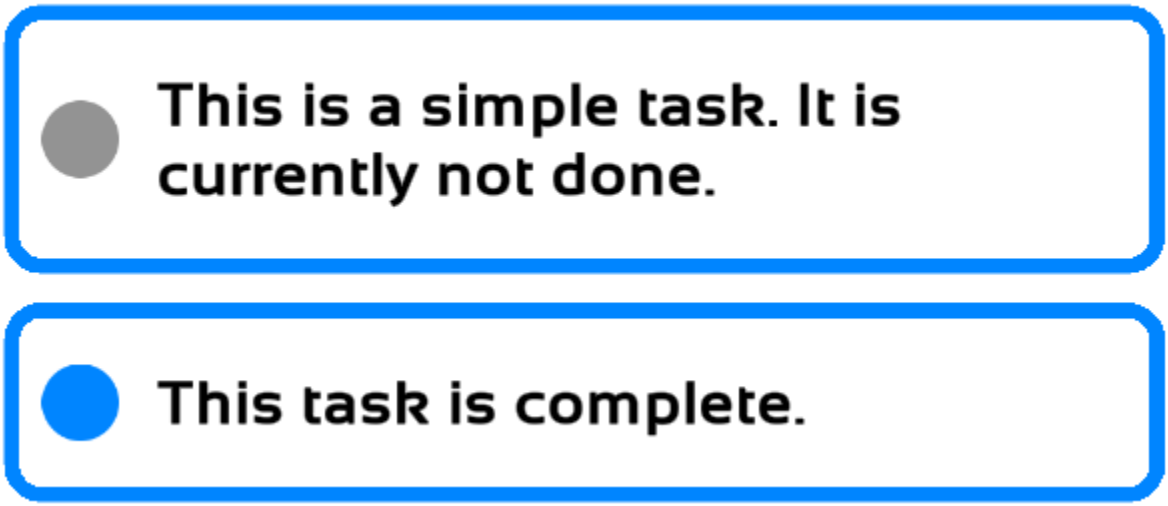


Slika 40 Jednostavna poruka

Veličina poruke se mijenja s obzirom na količinu teksta. Moguće je povezati dvije poruke u jednu tako da se kontrolerom jedna odvuče u drugu. Tada nastaje povezana poruka koja se ponaša isto kao i prije navedene poruke uz iznimku da je u njoj upisan sadržaj obje poruke odvojen novim redom. Plavi segment je tu iz estetskih razloga i sluzi da bi bila bolja vidljivost kojoj skupini pripada neka stavka.

Jednostavna poruka nadogradiva je u komponentu naziva “Zadatak” (engl. Task). Zadatak je interaktivni element koji nastaje odabirom alata iz alatnog izbornika i klikom na jednostavnu poruku. U tome trenutku jednostavna poruka pretvara se u zadatak.

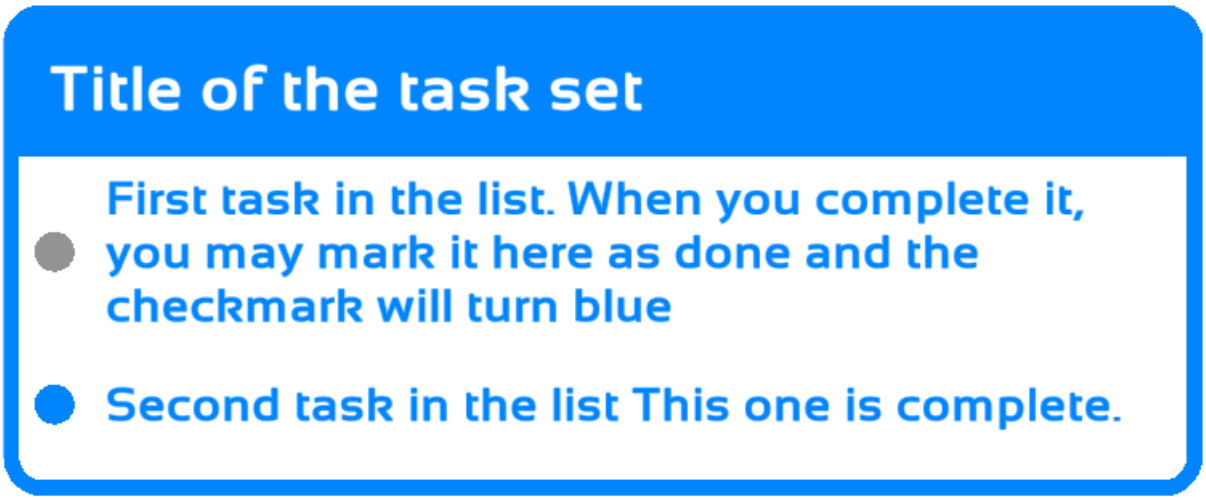
Interaktivna komponenta zadatka je kvačica koja indicira da li je zadatak obavljen ili nije. Dok je odabran alat za selekciju, moguće je mijenjati stanje te varijable.



Slika 41 Zadatak

Zadaci imaju istu interakciju s okolinom kao i jednostavne poruke. Mogu se postavljati na bilo koju interaktivnu plohu koristeći kontroler. Zadatke je moguće povezivati u listu zadataka, ili vratiti nazad u jednostavnu poruku. Ako je zadatak napravljen od povezane poruke, moguće ga je rastaviti na dva zadatka, bez da se stvori lista.

Lista zadataka stvara se na isti način kao sto se stvaraju povezane poruke. Korisnik treba odvuci jedan zadatak u drugi cime se spomenuta komponenta stvori.



Slika 42 Lista zadataka

ALATI ZA MANUPULACIJU INTERAKTIVNIK ELEMENATA

Pomoću kontrolera, moguće je pristupiti nizu alata koji se koriste za manipuliranje interaktivnim elementima u sceni.

Jednostavan alat za interakciju klikom. Koristi se kod interakcije sa zadacima ili listama zadataka kako bi se promijenio stanje zadatka.

Alat za pomicanje elemenata omogućuje korisniku da elemente razmješta po prostoru kako želi. Interakcija započinje pritiskom na tipkalo, a završava otpuštanjem tog tipkala. Koristeći ovaj alat, izvodi se povezivanje elemenata u veće skupine.

Koristeći ovaj alat, otvara se novi izbornik s preddefiniranim bojama koje se koriste kako bi korisnik označio pojedina stavke bojom (npr. Grupiranje po težini, projektu, osobi itd…)

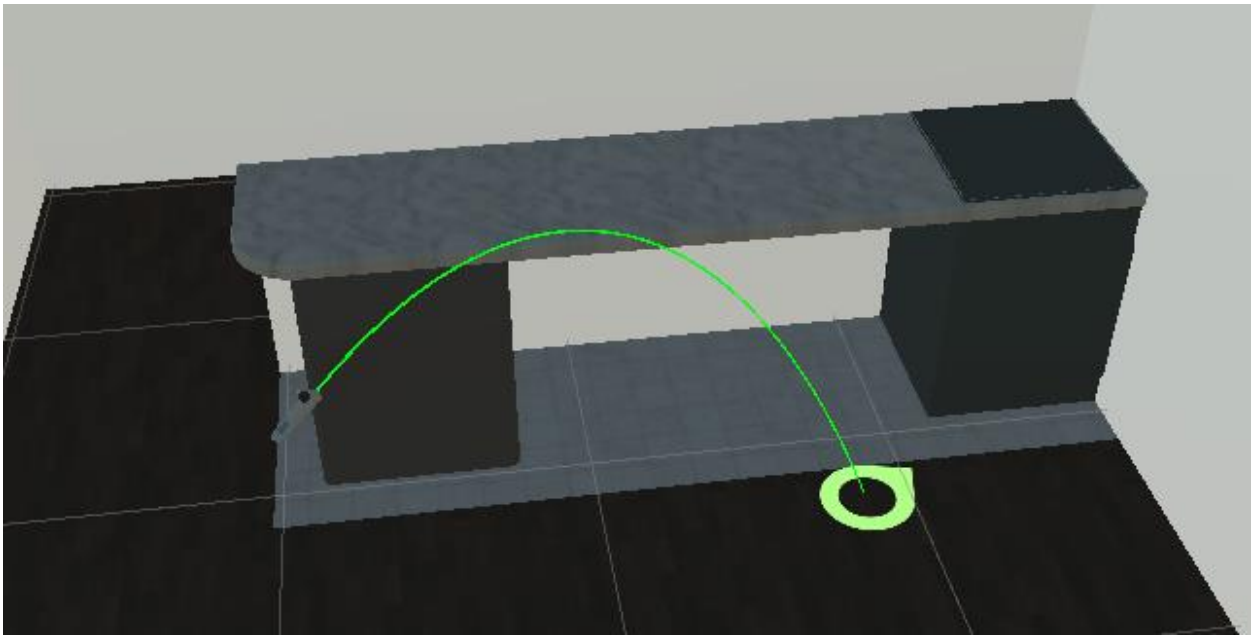
Ovim alatom korisnik postojeću komponentu razlama na njene osnovne komponente ako je to moguće. Ako se primjeni nad jednostavnom porukom načinjenom od nekoliko paragrafa, ta poruka će se razlomiti na odvojene paragrafe. Lista zadataka se rastavlja na pojedine zadatke.

Alat koji omogućuje nadogradnju komponente. Ako se primjeni na jednostavnu poruku, ona postaje zadatak.

Ovim alatom korisnik može poništiti postupak nadogradnje komponente. Ako se primjeni na zadatak, on će postati jednostavna poruka (zadržat će paragrafe ako su postojali)

NAVIGACIJA PO VIRUALNOM PROSTORU

Kretnja po virtualnom prostoru ostvarena je pomoću teleportacije. Teleportiranje je standardan način kretnje u statičnim VR igrama i temelji se na odabir pozicije na koju korisnik želi doći. Pod prostorije označen je kao posebna ploha po kojoj je moguća kretnja. Teleportiranje na zid nije moguće.



Slika 43 Teleportacija

Kako bi korisnik započeo teleportaciju, potrebno je pritisnuti tipkalo na kontrolnoj palici. Dok je tipkalo pritisnuto, iscrtavati ce se parabola kojoj je početak u vrhu kontrolera. Otpuštanjem tipkala korisnik potvrđuje lokaciju na koju se želi teleportirati.

Teleportacija nije moguća dok je otvoren izbornik na kontroleru.

ZAKLJUČAK

Sveukupan razvojni proces raspoređen kroz nekoliko mjeseci pokazao je kako se naizgled jednostavni problemi mogu razviti u kompleksne, bilo to zbog vanjski čimbenika ili nepredviđenih elemenata. Istraživački proces implementacije sklopovskog kontrolera otvorio je brojne mogućnosti i ideje za daljnje projekte, te me približio programskom jeziku kojeg nisam koristio toliko često. Brojne iteracije sklopovlja naučile su me kako pristupiti problemu na način da se nešto popravi, a ne da se piše ispočetka. Spomenuta tri sustava u radu su :

- VR kontroler
- Razvojno sučelje
- Demonstracijska aplikacija

Svaki pojedinačan sustav daje drugačiju perspektivu na sto treba obratiti pažnju, kako se nešto treba implementirati, i na što treba pripaziti ako bi ponudili nekome ovu implementaciju kao gotov proizvod kojeg bi ta osoba koristila za daljnji razvoj.

Prilikom razvoja VR kontrolera, puno vremena je utrošeno na istraživanje kako se ponašaju signali i smo može utjecati na njih. Opisane su moguće smetnje i kako ih izbjeći.

Kod dizajniranja razvojnog sučelja, glavna smjernica je bila “dizajnirati sučelje takvo da krajnji korisnik nema potrebe raditi izmjene na njemu”. Problem kod takvog pristupa je predvidjeti u koje bi se svrhe mogao koristiti kontroler, osim u ove po kojoj je dizajnirana demonstracijska aplikacija.

Demonstracijska aplikacija napravljena je kao skup komponenti koje bi demonstrirale rad svakog pojedinog elementa dostupnog u razvojnom sučelju, uz naglasak na vizualan izgled. Cilj je bio da aplikacija ne izgleda kao prototip nego da ima jednostavne, ali kvalitetne elemente kao sto su modeli za virtualni prostor, pregledno korisničko sučelje (engl. User Interface (UI)) i da pruža konzistentno korisničko iskustvo (engl. User Experience (UX)).

SAŽETAK

Diplomski rad istraživačkog tipa opisuje način na koji se prikupljaju signali dostupni iz pojedinih sklopova, njihovo korištenje i povezivanje u cjelinu. Kroz primjer je opisan način implementacije koda za razvojnu pločicu, čitanje podataka s inercijskog senzora i upravljačke palice, te korištenje tipkala sa spomenutom pločicom. Za sklopovski razvoj korišten je ESP32 senzor s ICM20948 Inercijskim senzorom. Demonstriran je način povezivanja sklopovlja s računalom pomoću UDP protokola.

Rad opisuje proces pisanja razvojnog okruženja i pozivanje elemenata koje kontroler pruža.

Demonstracijska aplikacija opisuje stvaranje jednostavne VR aplikacije za mobilne platforme, te korištenje implementiranog razvojnog sučelja u svrhu ostvarivanja željenih akcija.

SUMMARY

The research thesis describes the way in which the signals available from individual circuits are collected, their use and their connection. The example describes how to implement the code for the development board, read the data from the inertia sensor or the joystick, and the use of buttons with said board. An ESP32 microcontroller with an ICM20948 Inertia sensor was used for circuit development. The method of connecting hardware to a computer using the UDP protocol is demonstrated.

The paper describes the process of writing a development environment and invoking the elements provided by the controller.

The demonstration application describes the creation of a simple VR application for mobile platforms, and the use of the implemented development interface to achieve the desired actions.

POVEZNICE

[1] https://docs.unity3d.com/Manual/index.html

[2] https://docs.blender.org/manual/en/latest/

[3] https://www.autodesk.com/products/fusion-360/learn-support

[4] https://www.arduino.cc/en/main/software

[5] https://docs.microsoft.com/en-us/dotnet/csharp/

[6] https://code.visualstudio.com/

[7] https://platformio.org/

[8] https://github.com/ericalbers/ICM20948_DMP_Arduino

[9] https://github.com/Xinyuan-LilyGO/TTGO-T-Display

[10]https://en.wikipedia.org/wiki/Main_Page