SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Filip Bradarić, Marko Cavalli, Ivona Krajačić, Matija Kunštek, Andrea Rebec, Jelena Tabak

# Radionica za djecu "Robofun"

Zagreb, 2020.

*Ovaj rad izrađen je u Laboratoriju za robotiku i inteligentne sustave upravljanja na Fakultetu elektrotehnike i računarstva pod vodstvom izv. prof. dr. sc. Matka Orsaga i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2019./2020.*

# SADRŽAJ

# 1. Uvod

Fakultet elektrotehnike i računarstva (FER), predvođen Laboratorijem za robotiku i inteligentne sustave upravljanja, u prosincu 2019. godine je pod geslom „Adventska groznica obuzela robote s FER-a“ organizirao humanitarnu akciju za Društvo za poboljšanje kvalitete života siromašne i nezbrinute djece *Mali zmaj*. Akcija je trajala tjedan dana, a svi zainteresirani posjetitelji su u tom razdoblju imali priliku provjeriti glazbena, edukativna i ugostiteljska umijeća FER-ovih profesora, studenata i robota. Sve je počelo koncertom FER-ovog *Roboband*-a, sastavljenog od robota-instrumentalista i FER-ovih zaposlenika te se nastavilo gostujućim predavanjima profesora američkog sveučilišta George Mason. Posljednjeg dana akcije posjetitelji su u predvorju FER-a uživali u kuhanom vinu i čokoladnim kolačima koje su im posluživali roboti, dok su paralelno djeca osnovnoškolskog i srednjoškolskog uzrasta imala priliku sudjelovati na interaktivnoj radionici *Robofun*.

Radionica *Robofun* je rezultat rada studenata diplomskog studija Fakulteta elektrotehnike i računarstva u suradnji sa studenticom diplomskog Studija dizajna pri Arhitektonskom fakultetu u Zagrebu . Glavni cilj radionice je bila popularizacija znanosti, u prvom redu robotike, kroz dva odvojena modula. Prvi modul u fokus stavlja upravljanje robotom, odnosno direktnu i inverznu kinematiku, na način prilagođen uzrastu djece. Kroz drugi modul predstavljena je *soft* (podatna) robotika.

*Soft* robotika je područje robotike koje se bavi konstrukcijom robota korištenjem podatnih materijala, a inspiraciju pronalazi u građi živih organizama. Naime, kao što Kim i sur. navode u [1], životinjama njihova građa olakšava snalaženje unutar složenog okruženja i ispitivanje istog, a uz efikasnije snalaženje unutar okruženja, razvoj

*soft* materijala i prikladnih aktuatora omogućuje i sigurniju interakciju čovjeka i robota. Osim fleksibilnosti i sigurnosti, autori u [1] kao prednosti *soft* robota navode i jeftinije i jednostavnije komponente i kontrolere te stavljaju poseban naglasak na primjenu ovakvih robota u medicini.

Razvoj *soft* robotike svakako ovisi i o razvoju *soft* senzora. Chorley i sur. su u [2] predstavili taktilni senzor inspiriran strukturom ljudskog vrha prsta. Na unutrašnjoj strani vanjske opne navedenog senzora nalaze se markeri koji se kamerom snimaju, pa se praćenjem pomaka markera može donositi zaključak o deformaciji opne, odnosno o interakciji senzora i okoline. Senzor predstavljen u [2] poslužio je kao inspiracija za konstrukciju *soft* upravljačkih komponenti korištenih u sklopu *Robofun* radionice. Komponente su izrađene korištenjem silikonskih kalupa i jestivih materijala od želatine. Shintake i sur. u [3] potiču korištenje ovakvih, biorazgradivih materijala i predstavljaju aktuator izrađen od želatine. Navode da je uspješnost njihovog aktuatora jednaka uspješnosti aktuatora iste konstrukcije izrađenog od standardnih materijala korištenih u *soft* robotici.

Komponente izrađene u sklopu ovog rada korištene su za upravljanje videoigrom koja je osmišljena i razvijena za potrebe *Robofun* radionice. U ovom radu je opisan proces pripreme radionice, kao i sama provedba iste.

U drugom poglavlju je predstavljen proces izrade robotskog manipulatora koji je korišten pri demonstraciji direktne i inverzne kinematike. Navedene su mogućnosti upravljanja manipulatorom, kao i neka postojeća ograničenja. Dizajn komponenata videoigre te sam izgled i funkcionalnosti iste su objašnjene u trećem poglavlju. U četvrtom poglavlju je opisan cjelokupan izgled sustava upravljanja videoigrom, kao i proces izrade korištenih *soft* komponenata. Provedba radionice opisana je u petom poglavlju.

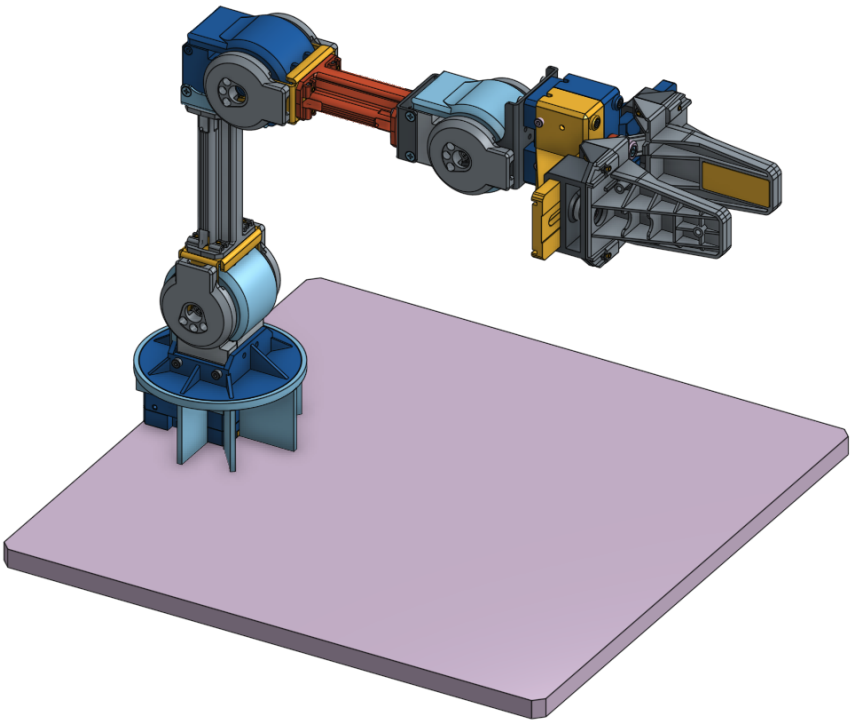
# 2. Robotski manipulator

Kako bi se složen teorijski dio upravljanja robotskim manipulatorom predstavio učenicima na jednostavan i zanimljiv način, održane su kratke praktične radionice, o kojima se više može pročitati u petom poglavlju. U sklopu praktičnih radionica učenici su imali priliku upoznati se s osnovnim i neophodnim postupcima potrebnim za upravljanje robotskim manipulatorima kroz tri kratka zadatka. Upravo zbog toga razvijen je robotski manipulator, koji iako je dimanzionalno malen, dovoljno je funkcionalan za obavljanje manje kompleksnih zadataka i za potvrđivanje opisanog teorijskog dijela. Korišteni robotski manipulator je prikazan na slici 2.1, a konstruiran je od strane tvrtke *Robotis* [4].

Tvrtka *Robotis* javno je objavila nacrte i 3D modelirane dijelove upravo u cilju popularizacije robotike među mlađim uzrastima, kao i u svrhu demonstracije osnovnih robotskih pojmova i algoritama. Nacrti, 3D modelirani dijelovi i upute dostupni su na [5]. Upravo zbog svojih namjena ovaj robotski manipulator bio je idealan izbor za kvalitetno i prije svega sigurno provođenje radionice.

## 2.1. Izrada robotskog manipulatora

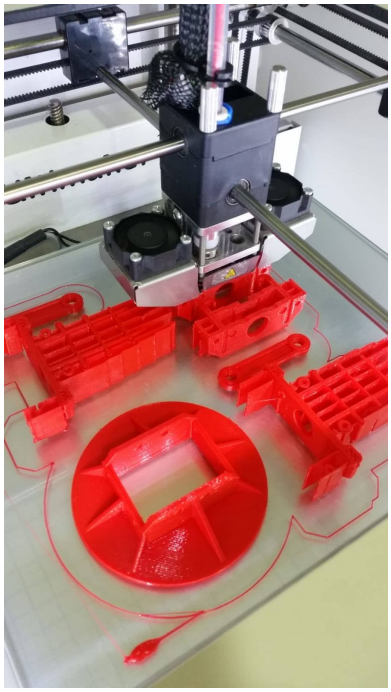
Iako zbog dostupnosti svih 3D modela nije bilo potrebno konstruirati robotski manipulator, bilo je potrebno spojiti sve dijelove i algoritme u jednu operativnu cjelinu. Na taj način se osiguralo kvalitetno održavanje praktičnog dijela radionice i u konačnici se ostvario cilj popularizacije robotike među mlađim naraštajima. Proces izrade robotskog manipulatora započinje izradom plastičnih dijelova robota. U sklopu izrade



**Slika 2.1:** Prikaz modeliranog robotskog manipulatora prema [5]

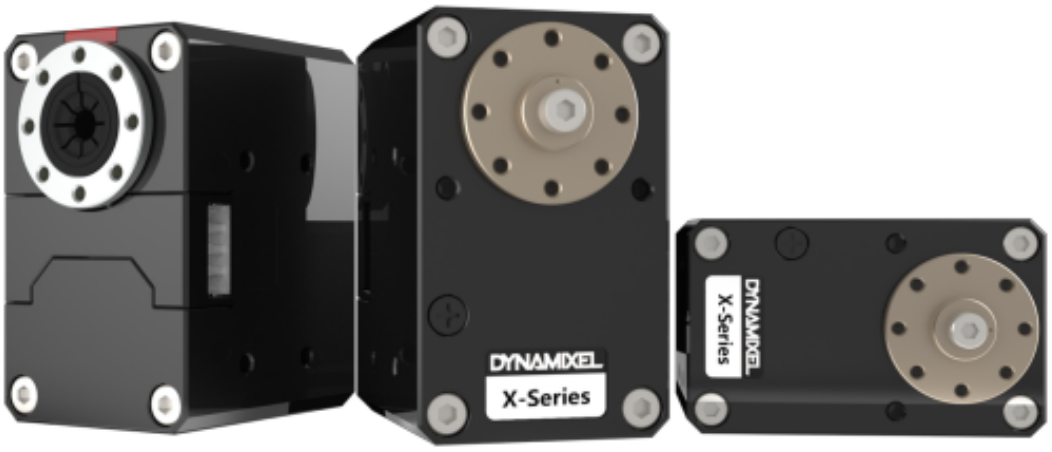
članaka, odnosno krutih dijelova robota, korišten je 3D printer. Korištenjem 3D printera na jednostavan, brz i kvalitetan način mogu se od plastike izraditi zadovoljavajući testni modeli, koji su u ovom slučaju dostatni za izradu realnog manipulatora. Proces izrade članaka pomoću 3D printera prikazan je na slici 2.2.

Nakon printanja članaka pomoću 3D printera, bilo je potrebno ukloniti sitne nepravilnosti i ispune. Kako članci robotskog manipulatora ne bi bili statični, korišteni su motori, koji ujedno predstavljaju i aktivne rotacijske zglobove. Za osiguravanje pozicijskog upravljanja robotskim manipulatorom korišteni su *Servo* motori. *Servo* motori su podvrsta električnih motora kojima je moguće upravljati po poziciji, odnosno po stupnju zakreta osovine motora u odnosu na standardno upravljanje po brzini. Prilikom odabira odgovarajućih *Servo* motora potrebno je voditi računa i o snazi samih motora. Pogrešnim odabirom snage motora, velik moment manipulatora na osovini motora može privremeno pregrijati motor ili u najgorem slučaju uništiti zupčanike unutar samog kućišta motora, čime održavanje praktične radionice ne bi bilo moguće. Zbog ovih zahtjeva te zbog konstrukcijskih karakteristika manipulatora, koji ima 4 stupnja



**Slika 2.2:** Proces izrade članaka pomoću 3D printera

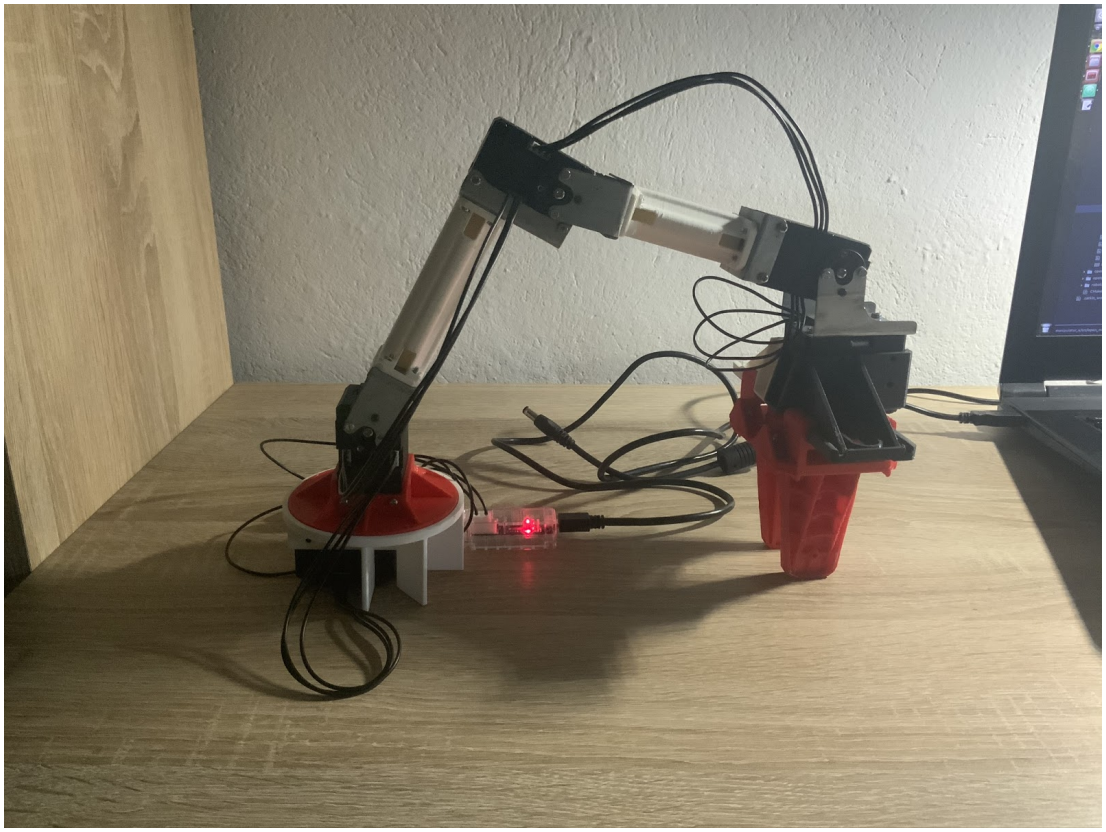
slobode i pomični vrh alata, korišteno je pet *Dynamixel XM430-W350-R* motora [6]. Korišteni *Dynamixel* motori prikazani su na slici 2.3.




**Slika 2.3:** *Dynamixel XM430-W350-R* motori [6]

Prije spajanja svih članaka i *Servo* motora u jednu funkcionalnu cjelinu, bilo je potrebno izraditi dijelove koji povezuju motore i članke. Zbog njihovog dimenzijama malog promjera, ove dijelove nije bilo moguće izraditi dovoljno čvrstima pomoću 3D printera. Upravo zbog toga su ovi dijelovi izrađeni od aluminija. Sklapanjem svih

dijelova dobiven je robotski manipulator prikazan na slici 2.4.



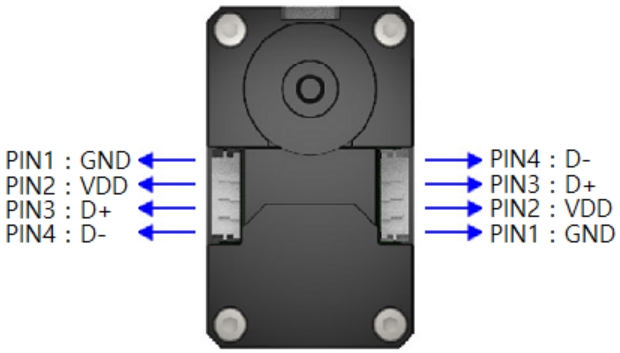
**Slika 2.4:** Korišteni robotski manipulator

## 2.2. Povezivanje robotskog manipulatora u funkcionalnu cjelinu

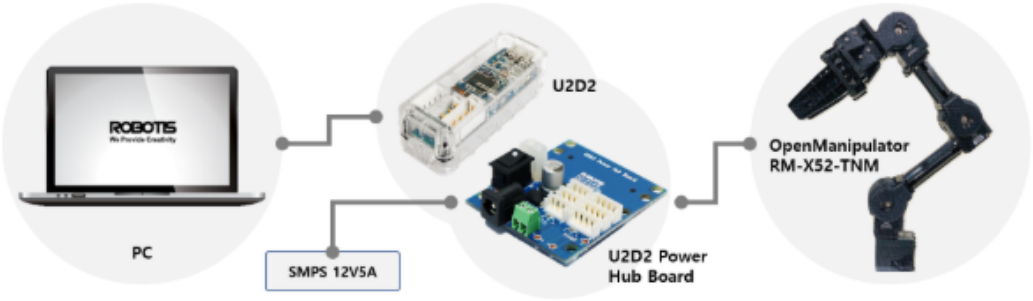
Za osiguravanje potpune mobilnosti i upravljivosti mobilnog robota korišteno je prijenosno računalo s instaliranom *Linux Ubuntu 16.04* distribucijom. Upravo je korištenjem *Linux* operacijskog sustava omogućeno povezivanje *Dynamixel* motora preko *U2D2* modula. Kako je na *U2D2* moguće spojiti samo jedan motor, ostala četiri motora spojena su u seriju s prvim motorom. Programski je svakom motoru pridodana odgovarajuća jedinstvena oznaka (ID) za raspoznavanje o kojem je motoru riječ. Postavljanjem različitih ID-eva za svaki motor osigurava se mogućnost zasebnog upravljanja. Kako prijenosno računalo ne pruža dovoljnu snagu za napajanje motora, potrebno

je koristiti zaseban izvor napajanja preko *U2D2 Power Hub* modula.

Serijsko povezivanje *Dynamixel* motora prikazano je na slici 2.5, dok je na slici 2.6 prikazano povezivanje motora s prijenosnim računalom.

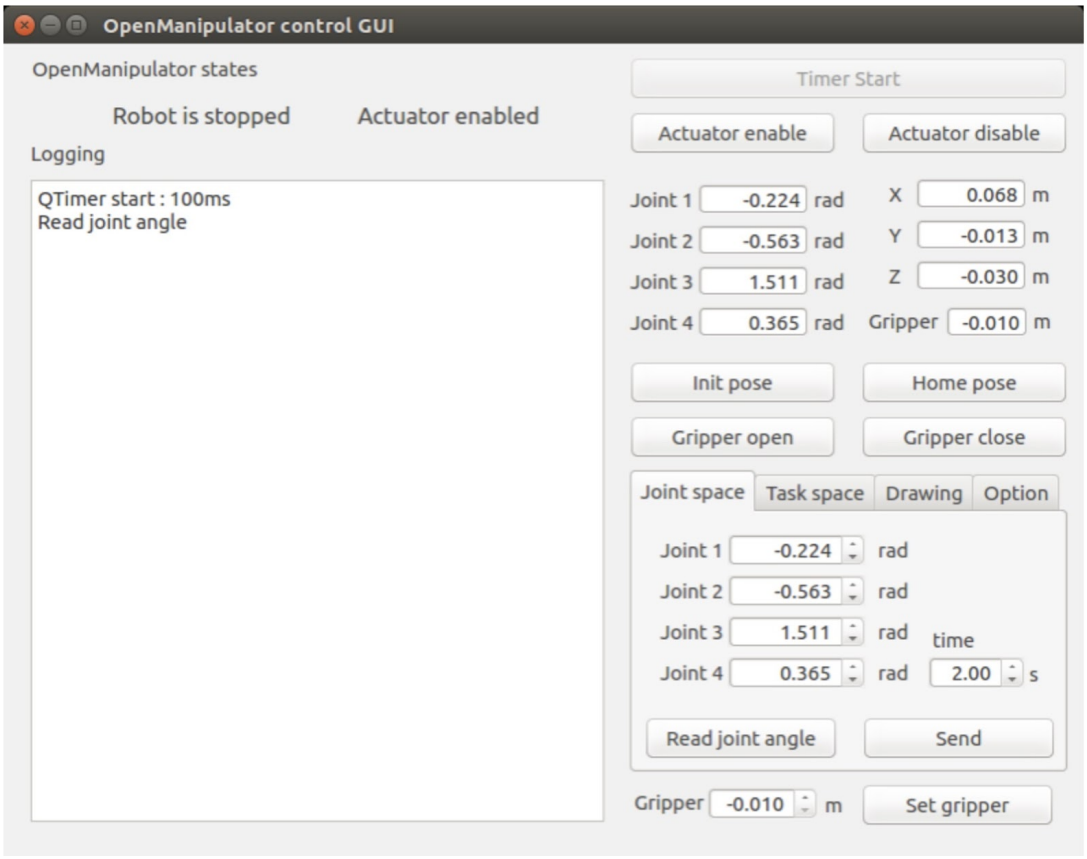



**Slika 2.5:** Serijsko povezivanje *Dynamixel* motora prema [6]




**Slika 2.6:** Povezivanje robotskog manipulatora s prijenosnim računalom prema [5]

Nakon povezivanja mobilnog robota s prijenosnim računalom i izvorom napajanja, radi lakšeg demonstriranja i upravljanja korišteno je grafičko sučelje povezano s robotskim manipulatorom. Povezivanje svih razina ostvareno je pomoću *ROS* (engl. *Robot Operating System*) *Kinetic* okruženja, a korišteno grafičko sučelje prikazano je na slici 2.7. Kao što se može vidjeti na slici, grafičko sučelje u svakom trenutku prikazuje poziciju svih zglobova u radijanima, kao i poziciju vrha alata u metrima u odnosu na bazni koordinatni sustav, odnosno koordinatni sustav koji se nalazi na mjestu spoja baze robota s podlogom. Uporabom grafičkog sučelja omogućeno je jednostavno upravljanje robotskim manipulatorom, u cilju što jednostavnijeg i bržeg izvršavanja zadataka opisanih u petom poglavlju.




**Slika 2.7:** Grafičko sučelje za upravljanje robotskim manipulatorom

## 2.3. Kinematika robotskog manipulatora

Prethodnim povezivanjem robotskog manipulatora u jednu funkcionalnu cjelinu osigurano je upravljanje robotskim manipulatorom samo pomoću promjene vrijednosti zakreta pojedinih zglobova. Iako se ovim načinom upravljanja može postići bilo koja točka unutar radnog prostora manipulatora, problem se javlja u kompleksnosti i nezgrapnosti samog postupka. Naime, ljudima je teško percipirati povezanost položaja i orijentacije vrha alata s vrijednostima zakreta pojedinih kutova. Razlog ovog problema leži u ljudskoj percepciji prostora. Drugim riječima ljudi prilikom pomicanja ruke ne razmišljaju o tome pod kojim kutom mora biti primjerice rame ili lakat da bi mogli uhvatiti određeni predmet. Ljudi promatraju okolni prostor intuitivno u kartezijskom koordinatnom sustavu u kojem svaki predmet ima točno određenu visinu, duljinu i širinu te se nalazi na točno određenoj udaljenosti. Upravo zbog toga razvijena je kinematika robotskog manipulatora koja u sklopu jednadžbi manipulatora daje odgovore na dva krucijalna pitanja. Odgovor na prvo pitanje poznat je pod nazivom direktna kinematika, a odnosi se na postupak određivanja točnog položaja i orijentacije vrha alata u odnosu na nepokretni koordinatni sustav pridružen bazi robota. Inverzna kinematika daje odgovor na drugo pitanje koje se odnosi na postupak određivanja vrijednosti vektora varijabli zglobova, odnosno zakrete svih zglobova uz poznatu željenu poziciju i orijentaciju vrha alata u baznom koordinatnom sustavu. U nastavku ovog poglavlja detaljnije su opisani postupci određivanja kinematike korištenog robotskog manipulatora.

### 2.3.1. Direktna kinematika

Prilikom određivanja rješenja direktne kinematike, odnosno određivanja osnovnih jednadžbi robotskog manipulatora potrebno je primijeniti osnovne trigonometrijske spoznaje. Zbog primjene osnovnih trigonometrijskih spoznaja, najčešće je potrebno uvesti velik broj pretpostavki koje u konačnici mogu rezultirati potpuno različitim, ali u suštini ispravnim rješenjima. Upravo s ciljem identifikacije univerzalnog rješenja

direktne kinematike definirana je Denavit-Hartenbergova metoda, detaljno opisana u [7]. Korištenjem Denavit-Hartenbergove metode uvode se konzistentne pretpostavke, a koje su neovisne o konfiguraciji promatranog robotskog manipulatora. Iako Denavit-Hartenbergova metoda sadrži 14 koraka, kompleksnost same metode može se smanjiti ponavljanjem jednostavnih transformacija koordinatnih sustava. Sam algoritam započinje povezivanjem svih zglobova robotskog manipulatora s pripadajućim koordinatnim sustavima. Točne pozicije i orijentacije koordinatnih sustava u 3D prostoru definirane su sljedećim pravilima:

1. Bazi robota pridruži se desno orijentirani koordinatni sustav $L_0$ tako da se os $z^0$ podudara s osi zgloba 1

2. Os $z^k$ postavi se u smjeru osi zgloba $z^{k+1}$, gdje je $k = 1, 2, 3, \ldots, n$ , n-broj zglobova nedostaje kontrasta u bojama.

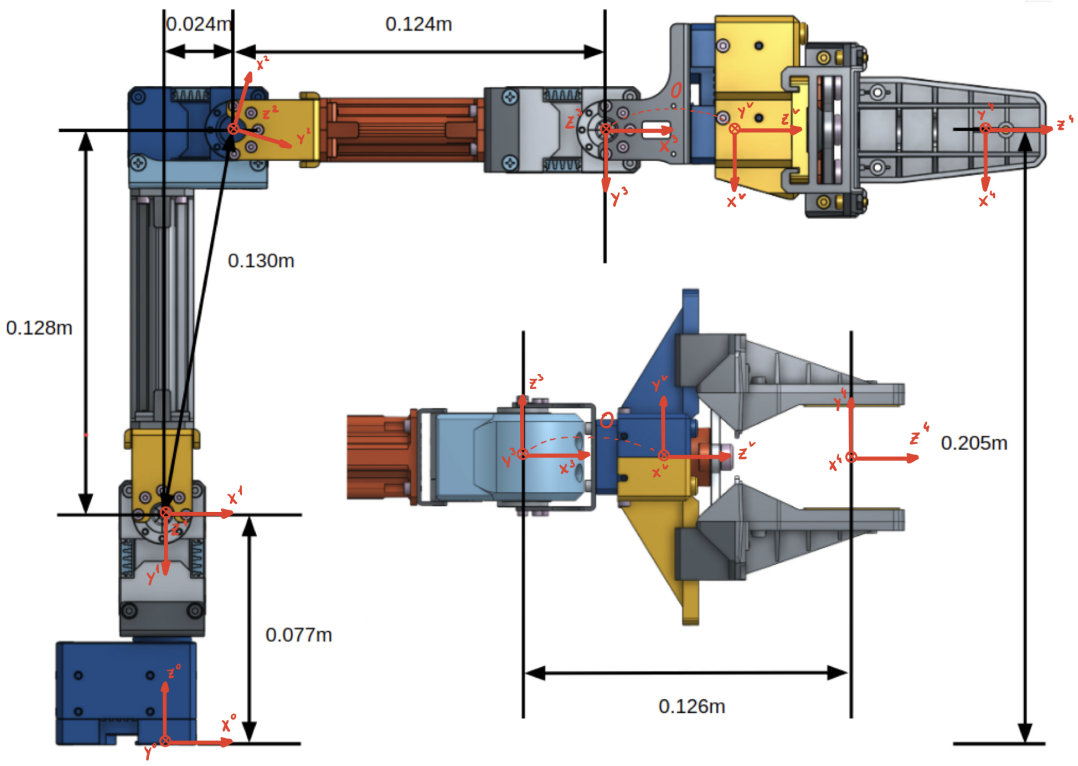
3. Ishodište koordinatnog sustava $L_k$ se postavi u presjecište osi $z^{k-1}$ i $z^k$. Ako se osi ne sijeku, tada se treba koristiti presjekom osi $z^k$ i zajedničke okomice na osi $z^{k-1}$ i $z^k$

4. Os $x^k$ postavi se tako da bude okomita na os $z^k$ i os $z^{k-1}$. Ako su te dvije osi paralelne, tada se postavi os $x^k$ tako da pokazuje smjer od osi $z^{k-1}$ prema osi $z^k$

5. Os $y^k$ postavi se tako da se dobije desno orijentirani koordinatni sustav $L_k$.

6. Ishodište koordinatnog sustava ln postavi se u vrh alata, zatim se vektoru približavanja alata pridruži os $z^n$, vektoru pomicanja os $y^n$ i okomitom vektoru os $x^n$.

Pridruživanjem koordinatnih sustava zglobovima manipulatora po prethodno navedenim pravilima dobiva se načelna shema robotskog manipulatora prikazana na slici 2.8.




**Slika 2.8:** Načelna shema robotskog manipulatora

Iako korišteni robotski manipulator ima samo četiri stupnja slobode uz pripadajući vrh alata, načelna shema robotskog manipulatora ima šest različitih koordinatnih sustava. Razlog ovog nesrazmjera leži u dodatnom virtualnom koordinatnom sustavu $L_v$. Dodatni virtualni koordinatni sustav ima istu poziciju kao i koordinatni sustav $L_3$, ali sadrži orijentaciju koja odgovara orijentaciji koordinatnog sustava pridruženom vrhu alata. Dodatnim koordinatnim sustavom ne mijenja se struktura robotskog manipulatora, ali se njegovom primjenom osigurava korištenje duljine $d_5$ u jednadžbi manipulatora.

Kao podrezultat korištenja Denavit-Harenebergove metode dobivaju se vrijednosti za četiri parametra koja su poznata pod nazivom DH parametri. Od dobivena četiri parametra dva parametra, $\theta$ i $d$, predstavljaju varijable zglobova, dok preostala dva parametra, $\alpha$ i $a$, predstavljaju varijable članaka. Sami iznosi DH parametara dobivaju se jednostavnim manipulacijama prethodno pridruženih koordinatnih sustava tako da

se u konačnici susjedni koordinatni sustavi poklapaju. Pomoću prve manipulacije, koja se odnosi na rotaciju koordinatnog sustava $L_{k-1}$ oko $z^{k-1}$ na način da se osi $x^{k-1}$ i $x^k$ poklapaju, dobiva se iznos parametra $\theta$. Nadalje, pomakom koordinatnog sustava $l_{k-1}$ po osi $z^{k-1}$ tako da osi $x^{k-1}$ i $x^k$ postanu kolinearne određuje se parametar $d$. Upravo korištenjem tih dviju prethodno navedenih manipulacija zglob $k$ se može u potpunosti opisati u $k-1$ koordinatnom sustavu. Trećom manipulacijom koordinatnog sustava $L_{k-1}$ oko osi $x^k$ da se osi $z^{k-1}$ i $z^k$ poklapaju, dobiva se parametar $\alpha$. Pomakom koordinatnog sustava $L_{k-1}$ po osi $x^k$ tako da osi $z^{k-1}$ i $z^k$ postanu kolinearne dobiva se i posljednji DH parametar (parametar $a$).

Određivanjem svih DH parametara uz korištenje prethodno opisanog postupka, može se razviti tablica 2.1. U tablici 2.1, upisane su i varijable $q_1$, $q_2$, $q_3$ i $q_4$ jer upravo one predstavljaju zakrete svih upravljivih zglobova u trenutku $t$.

| - | $\theta$ | $d$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| 1.os | $q_1[0]$ | $d_1 = 0.077$ | 0 | $\frac{-\pi}{2}$ |
| 2.os | $q_2[-1.3854]$ | 0 | $a_2 = 0.13$ | 0 |
| 3.os | $q_3[1.3854]$ | 0 | $a_3 = 0.124$ | 0 |
| 4.os | $q_4[\pi]$ | 0 | 0 | $\frac{\pi}{2}$ |
| 5.os | 0 | $d_5 = 0.126$ | 0 | 0 |

**Tablica 2.1:** Kinematički parametri manipulatora

Za određivanje jednadžbi manipulatora, potrebno je prethodno određene DH parametre uvrstiti u matricu homogene transformacije koja je prikazana jednadžbom 2.1. Matrica homogene transformacije preslikava pokretni koordinatni sustav u koordinate prethodnog sustava, sve dok je ispunjen uvjet prikazan jednadžbom 2.1. U slučaju da uvjet nije ispunjen tada se pomoću matrice homogene transformacije pokretni koordinatni sustav preslikava u koordinate nepokretnog, baznog koordinatnog sustava.

$$T_{k-1}^{k} = \begin{bmatrix} cos(\Theta_k) & -cos(\alpha_k) \cdot sin(\Theta_k) & sin(\alpha_k) \cdot sin(\Theta_k) & a_k \cdot cos(\Theta_k) \\ sin(\Theta_k) & cos(\alpha_k) \cdot cos(\Theta_k) & -sin(\alpha_k) \cdot cos(\Theta_k) & a_k \cdot cos(\Theta_k) \\ 0 & sin(\Theta_k) & cos(\Theta_k) & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

Uzastopnim množenjem definiranih matrica homogene transformacije dobiva se matrica složene homogene transformacije. Ona predstavlja matričnu jednadžbu manipulatora. Oblik matrične jednadžbe manipulatora prikazan je sljedećom jednadžbom:

$$T_0^n = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot ... \cdot T_{(n-2)}^{(n-1)} \cdot T_{(n-1)}^n \tag{2.2}$$

Uvrštavanjem prethodno određenih DH parametara u jednadžbu 2.1, a zatim i u jednadžbu 2.2 dobivena je matrična jednadžba manipulatora prikazana jednadžbom 2.5. Zbog jednostavnosti zapisa korišten je pojednostavljen zapis trigonometrijskih funkcija ($sinq_i = S_i$, $cosq_i = C_i$, $sin(q_i + q_j) = S_{ij}$, $cos(q_i + q_j) = C_{ij}$), a kutovi zakreta $q_2$ i $q_3$ zamijenjeni su pomaknutim kutovima $q_2'$ i $q_3'$, pri čemu vrijede jednadžbe 2.3 i 2.4.

$$q_2' = q_2 - 1.3854 \tag{2.3}$$

$$q_3' = q_3 + 1.3854 \tag{2.4}$$

$$T_0^5 = \begin{bmatrix} -S_{2'3'4} \cdot C_1 & -S_1 & C_{2'3'4} \cdot C_1 & C_1 \cdot (0.126 \cdot C_{2'3'4} + 0.124 \cdot C_{2'3'} + 0.13 \cdot C_2') \\ -S_{2'3'4} \cdot S_1 & C_1 & C_{2'3'4} \cdot S_1 & S_1 \cdot (0.126 \cdot C_{2'3'4} + 0.124 \cdot C_{2'3'} + 0.13 \cdot C_2') \\ -C_{2'3'4} & 0 & -S_{2'3'4} & 0.077 - 0.124 \cdot S_{2'3'} - 0.13 \cdot S_2' - 0.126 \cdot S_{2'3'4} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Prethodno definirana matrična jednadžba manipulatora može se zapisati i uz pomoć podmatrica $\boldsymbol{R}$, $\boldsymbol{p}$ i $v$ što prikazuje sljedeća jednadžba:

$$T_0^5 = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ v & 1 \end{bmatrix} \tag{2.6}$$

Pomoću podmatrice $\boldsymbol{R}$ s dimenzijama 3x3 definira se orijentacija vrha alata, dok se pomoću podmatrice $\boldsymbol{p}$ s dimenzijama 3x1 određuje položaj vrha alata u odnosu na nepokretni bazni koordinatni sustav manipulatora. Uz to definira se i nulmatrica $v$ dimenzija 1x3.

### 2.3.2. Inverzna kinematika

Definiranjem matrične jednadžbe manipulatora riješen je problem direktne kinematike te je postavljen temelj za rješavanje inverzne kinematike. Budući da inverzna kinematika predstavlja postupak određivanja vektora varijabli zglobova uz poznatu poziciju i orijentaciju vrha alata, potrebno je definirati nove varijable. Novo definirane varijable $w_1$, $w_2$ i $w_3$ predstavljaju željeni položaj vrha alata, dok njegovu željenu orijentaciju predstavljaju varijable $w_4$, $w_5$ i $w_6$ . Izraz novodefiniranih varijabli može se prikazati pomoću sljedećih jednadžbi:

$$w_1 = C_1 \cdot (0.126 \cdot C_{2'3'4} + 0.124 \cdot C_{2'3'} + 0.13 \cdot C_2') \tag{2.7}$$

$$w_2 = -S_1 \cdot (0.126 \cdot C_{2'3'4} + 0.124 \cdot C_{2'3'} + 0.13 \cdot C_2') \tag{2.8}$$

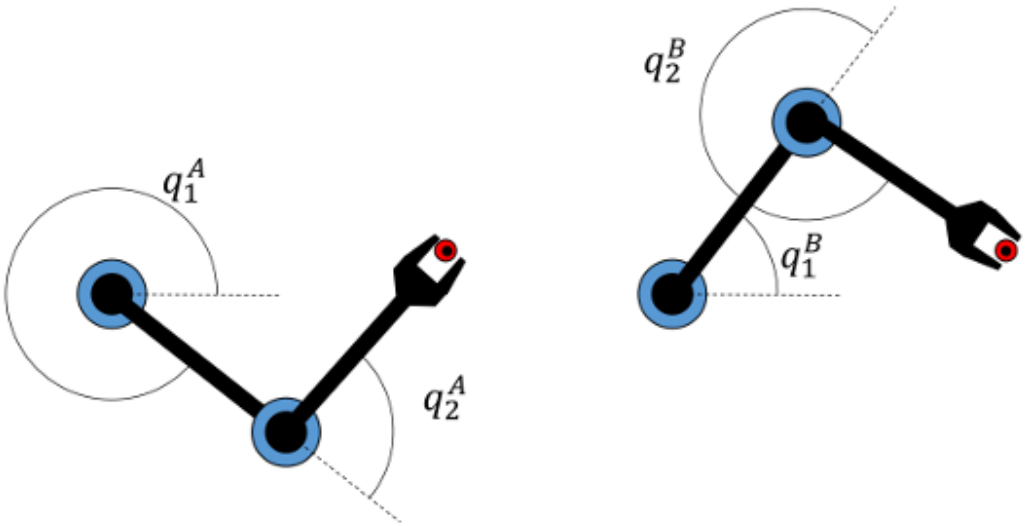
$$w_3 = 0.077 - 0.124 \cdot S_{2'3'} - 0.13 \cdot S_2' - 0.126 \cdot S_{2'3'4} \tag{2.9}$$

$$w_4 = C_{2'3'4} \cdot C_1 \tag{2.10}$$

$$w_5 = C_{2'3'4} \cdot S_1 \tag{2.11}$$

$$w_6 = -S_{2'3'4} \tag{2.12}$$

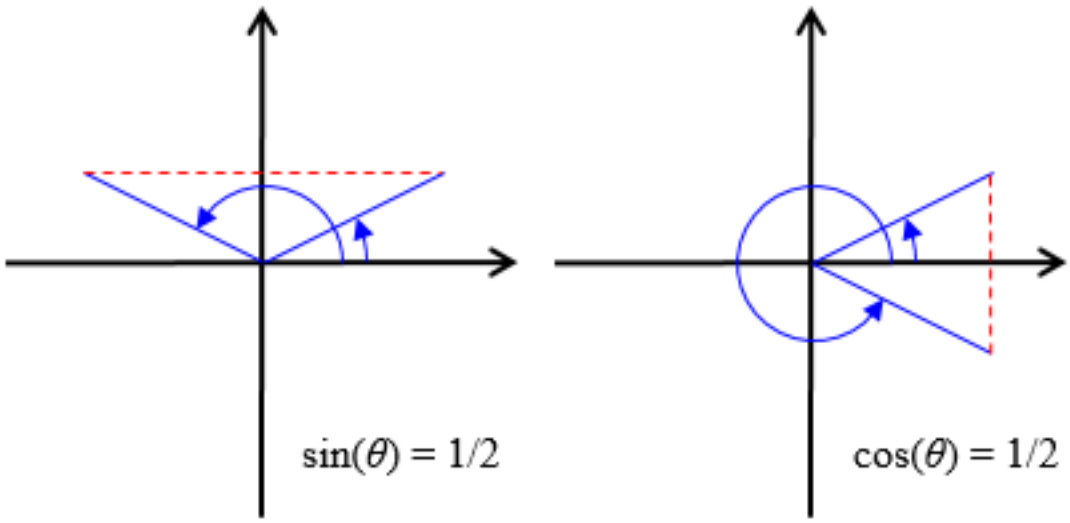
Raspisom prethodne jednadžbe dobiva se skup jednadžbi koje je potrebno riješiti po kutovima $q_1$, $q_2'$, $q_3'$ i $q_4$. Iako se korištenjem prethodno definiranog skupa jednadžbi može doći do rješenja inverzne kinematike, kompleksnost samog rješenja predstavlja veći problem. Upravo zbog kompleksnosti samog rješenja s jedne strane, te korištenja složenijih konstrukcija robotskih manipulatora s druge strane, razvijeni su algoritmi i programi pomoću koji se na iterativne načine može doći do rješenja inverzne kinematike. Pri tome neki od razvijenih algoritama su CCD (eng. Cyclic Coordinate Descent) [8] i Jakobijan metoda [9]. Prilikom odabira rješenja inverzne kinematike dolazi i do problema redundantnosti rješenja. Redundantnost rješenja se definira kao postojanje dva ili više rješenja inverzne kinematike pomoću koji se može ostvariti željena pozicija i orijentacija vrha alata. Redundantnost robotskog manipulatora može se najbolje predočiti slikom 2.9, gdje se postiže jednaka pozicija vrha alata uz različite izračunate vektore varijabli zglobova.




**Slika 2.9:** Redundandnost robotskog manipulatora prema [10]

Pojednostavljeni razlog postojanja redundantnosti rješenja leži u većem broju stupnjeva slobode robotskog manipulatora nego što je dimenzija prostora. Iako bi se ogra-

ničavanjem broja stupnjeva slobode mogao riješiti ovaj problem, nedostatak ovog postupka uočava se prilikom obavljanja kompleksnijih operacija koje iziskuju obilaženje prepreka koje se nalaze unutar radnog prostora manipulatora. No pravi razlog postojanja redundantnosti inverzne kinematike leži u redundantnosti trigonometrijskih funkcija koje se koriste za opis rotacijskih zglobova s pripadajućim člancima. Redundantnost trigonometrijskih funkcija najbolje se može prikazati pomoću slike 2.10.




**Slika 2.10:** Redundantnost trigonometrijskih funkcija prema [11]

Kao što je prikazano na slici 2.10, trigonometrijske funkcije mogu imati jednake vrijednosti uz različite kutove. Zbog toga je od iznimne važnosti da se tijekom rješavanja inverzne kinematike obuhvate sva moguća rješenja. Upravo zbog postojanja redundantnosti rješenja, prilikom odabira idealnog rješenja inverzne kinematike vrše se razne optimizacije po utrošku energije, prijeđenom putu, očuvanju brzine i akceleracije itd. Pri tome je važno osigurati izbjegavanje svih mogućih kolizija među susjednim člancima kao s okolnim predmetima u radnome prostoru robotskog manipulatora. Upravo se definiranjem kinematike robotskih manipulatora značajnije pojednostavljuje postupak upravljanja manipulatorima, što u konačnici i omogućava automatizaciju robotskih postrojenja. Primijenjeni algoritmi inverzne i direktne kinematike su implementirani unutar grafičkog sučelja što je objašnjeno u prethodnom podpoglavlju uz sliku 2.7.

# 3. Videoigra *Santa's Delivery Drone*

Zastupljenost videoigara u raznovrsnim provedbama projekata posljedica je suvremenih istraživanja koja ispituju njihov utjecaj na razvoj osnovnih i intelektualnih vještina kod djece. Griffiths u [12] navodi pozitivan utjecaj videoigara korištenih u edukacijske svrhe. Uočeno je povećanje vještina u timskom radu, razvija se sposobnost kritičkog promišljanja, poboljšano je razumijevanje edukativnog sadržaja, pamćenje, spacijalno (prostorno) snalaženje i percepcija te aktivno razumijevanje problema i rješavanje istog. S obzirom na navedene ciljeve radionice i uvjetovanost vremenskog trajanja iste te potrebu za edukativnom svrhom, razvijena je videoigra koja će zadovoljiti spomenute kriterije.

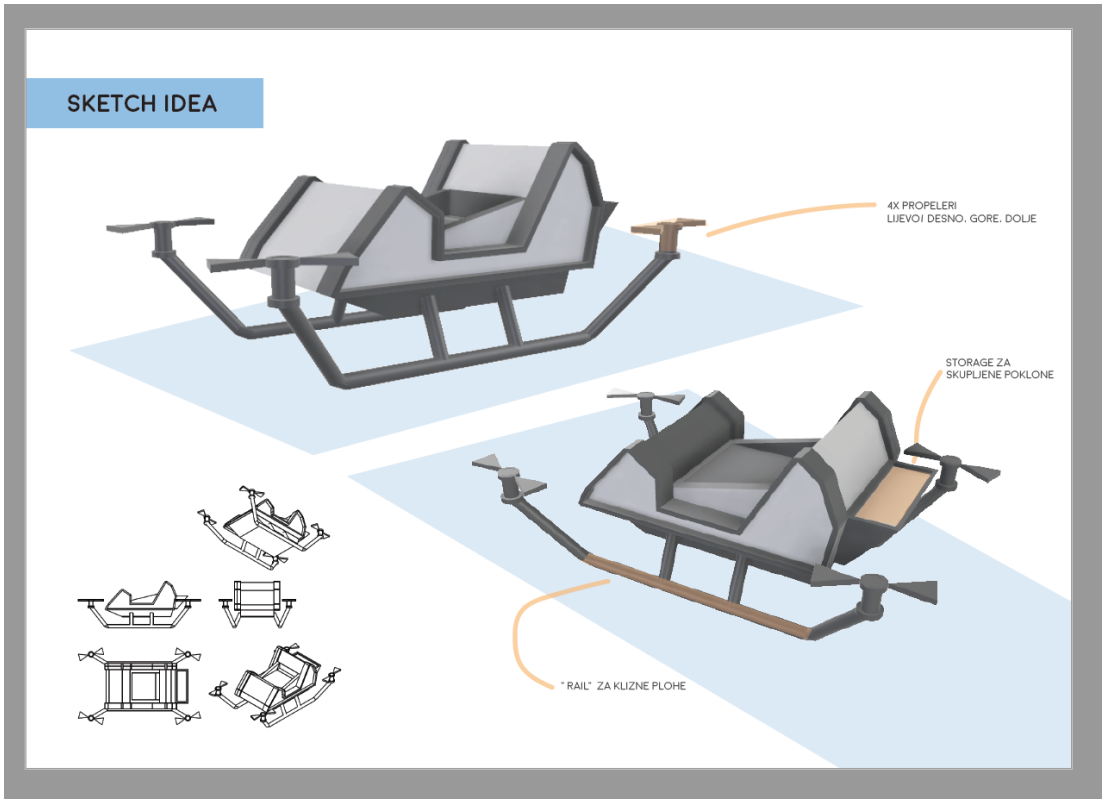
Humanitarna akcija održana u predbožićno razdoblje odredila je i smjer vizualnog oblikovanja i dizajn protagonista. Prepoznatljiv lik Djedice oblikovan je od jednostavnih geometrijskih volumena s naglašenim karakteristikama poput crvenog odijela, čizmica, snježne brade i kapice, te kao takav ne opterećuje igrača u razumijevanju samog sadržaja videoigre i daje jasni narativ i što se očekuje od korisnika, a to je prikupljanje poklona i dostava istih u dimnjake susjednih kućica. Dodatan naglasak na Laboratorij za robotiku i inteligentne sustave upravljanja kao organizatora radionice stavlja se kroz saonice Djedice koje su prepravljene u lebdeći dron kojim korisnik upravlja kroz upravljačke kontrole. U namjeri održavanja dinamike videoigre kroz zabavan i edukativan karakter te uključivanja što većeg broja igrača koji će isprobati videoigru, postavljeno je vremensko ograničenje prikupljanja poklona koje je jasno vidljivo na sučelju videoigre.

## 3.1. Razrada koncepta i izvedba Djedičinih saonica

Želja za provlačenjem ideje same robotike utjecala je na vizualno oblikovanje elemenata videoigre tj. saonica kojima upravlja Snježni Djedica. Glavna ideja koncepta bila je spojiti klasične saonice za snijeg sa suvremenim dronovima koji su danas sve više zastupljeni u privatnoj upotrebi. Promotrimo li saonice, njihova logika kretanja je klizanje na horizontalnoj površini zahvaljujući skijama koje su povezane sa samim tijelom saonica. Kod analize kretanja drona, prikazane slikom 3.1 vidljiva je potpuno drugačiju vrstu kretnje. Naime, zahvaljujući svojim lopaticama, dron je u mogućnosti ostvarivati različito kretanje u odnosu na više osi te zbog toga postaje znatno zanimljiviji kao ideja upravljanja u videoigri jer se omogućuje kretanje kroz prostor u više smjerova. Zamjena logike kretanja s horizontalne površine na lebdeće pomicanje zahtijevala je drugačije oblikovanje saonica te je bilo potrebno osmisliti način na koji djeluje mehanizam lopatica.


**Slika 3.1:** Istraživanje karakteristika kretanja saonica i drona

Pozicija lopatica u odnosu na trup saonica trebala je biti takva da lopatice imaju dovoljno prostora za rotaciju svojih lopatica. Broj lopatica je četiri, jer je na taj način omogućeno višeosno kretanje u zraku, a nastavci na kojima se nalaze lopatice nastavljaju se na donji dio trupa saonica kako bi se održala početna ideja klasičnih saonica Djeda mraza. Skica saonica prikazana je na slici 3.2.




**Slika 3.2:** Skica saonica za videoigru

Zadržavajući zajednički vizualni jezik svih elemenata videoigre, pojednostavljeno oblikovanje geometrijskih volumena, saonice su izrađene od više kubusa koji su kroz rezanje u prostoru poprimili željeni oblik. Skica i završni model, koji se može vidjeti na slici 3.3, su oblikovani u programu *Autodesk Fusion 360* [13]. Navedeni program upotrebljava se u izradi 3D modela predmeta i prostora. Besplatan je za korištenje za učenike i studente koji imaju želju naučiti i/ili nadograditi svoje znanje u trodimenzionalnom prikazu ideja, ali i u brzim preinakama unutar programa.

Izgled završnog modela saonica ishod je promišljanja o logici oblikovanja spoja drona i klasičnih snježnih saonica, ideji kretanja u zračnom prostoru u više smjerova,

**Slika 3.3:** Završni dizajn modela saonica

ispunjenju kriterija vizualne usklađenosti s modelom Djedice i okoline u kojoj se nalaze saonice i estetskom oblikovanju kroz jednostavnu prepoznatljivost funkcionalnosti i svrhe saonica. Uporaba boja na modelu vrlo je pojednostavljena - korištene su boja metala, crvena i tamnozelena, uz dodatak teksture stakla na sjedištu saonica kroz koje je vidljiv sam Djedica.

## 3.2. Oblikovanje lika Djedice

Nastavljajući se na oblikovanje saonica, koncept Djedice proizašao je iz jednostavnog slaganja volumena kubusa i sfere jednog na drugi te pomicanjem istih kako bi se dobile karakteristike Djedice. U izradi modela bilo je potrebno zadržati prepoznatljivost lika kroz njegove poznate odjevne predmete koji su postignuti različitim dodavanjem boja na prethodno spomenute kubuse. Bitno je prisjetiti se kako su polaznici radionice većinom djeca, pa se ovakvim oblikovanjem poticao interes onih koji imaju želju za vlastitim kreativnim izražavanjem u programima za 3D oblikovanje. Promicala se ideja da je počevši od jednostavnih modela moguće dodatnim učenjem graditi kompleksnije modele, ovisno o izraženom interesu. S druge strane, jednostavnim oblikovanjem Djedice održala se i doza humora u videoigri. Model je, kao i model

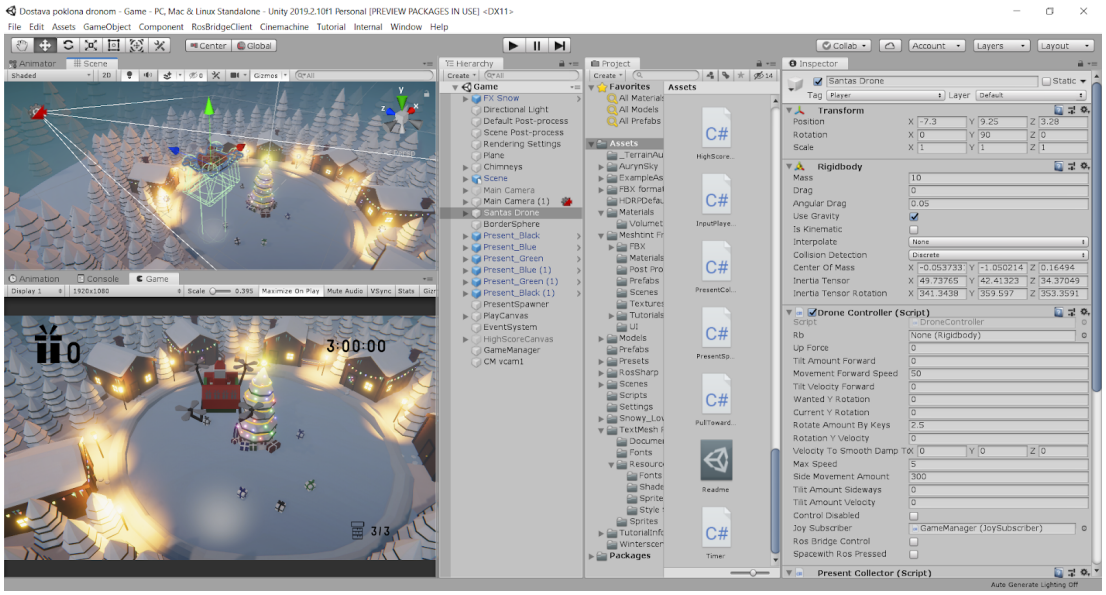
saonica, izrađen u programu *Autodesk Fusion 360*, a prikazan je slikom 3.4.

**Slika 3.4:** Završni model Djedice

## 3.3. *Unity*-pokretač igre

Igra je izrađena u *Unity*-ju, višeplatformskoj razvojnoj okolini za razvoj 2D i 3D igara. Logika igre pisana je u *C#* programskom jeziku. *Unity* dolazi s brojnim opcijama koje ubrzavaju proces izrade videoigara kao što su učitavanje modela, gotovi sjenčari, upravljanje materijalima, izrada animacija, izgradnja scene, sustava čestica itd. Velika prednost *Unity*-ja nad drugim pokretačima igara (engl. *Game engine*) je što ima jako veliku zajednicu korisnika i jako je lako doći do odgovora jer postoji velika baza postavljenih pitanja i pripadajućih odgovora. Izgled sučelja je prikazan na

slici 3.5. Igra je izrađena u verziji *Unity*-ja 2019.2.10f1. Korišten je novi HDRP način iscrtavanja kako bi se postigla što preciznija kalkulacija osvjetljenja.




**Slika 3.5:** Prikaz Unity sučelja

## 3.4. Mehanika igre

Igra traje dvije minute unutar kojih je potrebno dostaviti što više poklona u dimnjake.

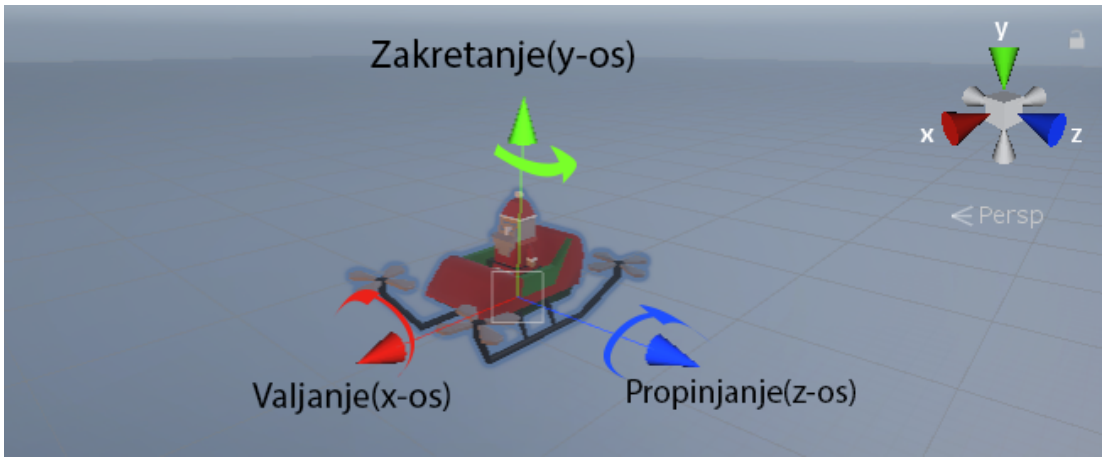
### 3.4.1. Kretanje

Igrač se može kretati unaprijed, unatrag te se može rotirati ulijevo i udesno. Kretanje se na tipkovnici kontrolira tipkama WASD i/ili strelicama. Ako se igra s gumbima od želatine svaki gumb je zadužen za jednu akciju. Postoji po jedan gumb za kretanje unaprijed, unatrag, rotaciju ulijevo i rotaciju udesno te jedan gumb za dostavu i prikupljanje poklona. Detalji izrade gumba su opisani u četvrtom poglavlju. Visina na kojoj saonice lete je fiksna kako bi se olakšalo upravljanje saonicama i uklonio jedan stupanj slobode kretanja zbog ograničenog broja gumba od želatine.

Prilikom kretanja unaprijed saonice se zarotiraju u pozitivnom smjeru oko svoje $z$-osi za kut $\Theta$ (propinjanje) te im se doda sila u smjeru lokalne $z$-osi. Prilikom kretanja unatrag saonice se zarotiraju u negativnom smjeru oko svoje $z$-osi za kut $\Theta$ te im se doda sila u smjeru lokalne $z$-osi. Prilikom rotacije saonice se rotiraju oko svoje $y$-osi za kut $\Psi$ (zakretanje). Osi rotacije su navedene u nastavku i prikazane slikom 3.6.

Osi rotacije saonica:

- $\Psi$ - kut zakretanja, nagib saonica oko okomite $y$-osi,
- $\Theta$ - kut propinjanja, nagib saonica oko poprečne $z$-osi,
- $\Phi$ - kut valjanja, nagib saonica oko uzdužne $x$-osi.




**Slika 3.6:** Lokalni koordinatni sustav saonica i osi rotacije

### 3.4.2. Skupljanje i dostava poklona

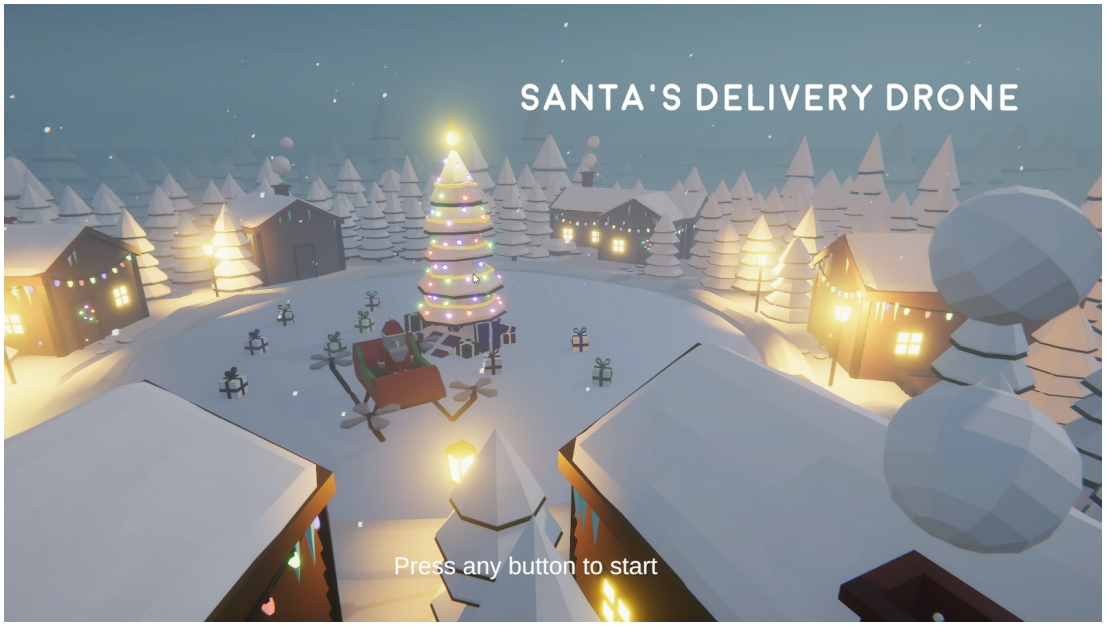
Igrač skuplja poklone tako da se pozicionira saonicama iznad poklona i pritisne gumb za skupljanje poklona. Igrač dostavlja poklone u dimnjake tako da se pozicionira iznad dimnjaka iz kojeg ne izlazi dim i pritisne gumb za dostavljanje poklona. Poklon se može skupiti i dostaviti samo ako se saonice nalaze dovoljno blizu dimnjaku/poklonu. Kako bi se igraču olakšalo zaključivanje je li unutar dosega u kojem može skupiti/dostaviti poklon, dodano je reflektorsko svjetlo na dno saonica koje baca svjetlost prema tlu.

## 3.5. Scene u igri

Igra se sastoji od četiri scene: početna scena, glavna scena, scena s prikazom ostvarenog rezultata i završna scena. Svaka od njih će biti detaljnije pojašnjena u nastavku.

### 3.5.1. Početna scena

Nakon pokretanja igre, igraču je prikazana scena malog snježnog sela s kućicama raspoređenim u krug oko središnjeg božićnog drvca, vidljiva na slici 3.7. Djedica sjedi u svojim letećim saonicama s lopaticama koji se lagano rotiraju. Pokloni su razbacani oko drvca i čekaju da budu dostavljeni u dimnjake okolnih kuća. Atmosfera je upotpunjena uličnim lampama, božićnim lampicama, svjetlima iz kuća te dimom koji polagano izlazi iz dimnjaka i snijegom koji pada. Na ekranu je također vidljiv naziv igre, *Santa's Delivery Drone*, i kratka poruka na engleskom jeziku, *Press any button to start*, koja govori igraču da pritisne bilo koji gumb kako bi započeo igru. Pritiskom gumba otvara se glavna scena.




**Slika 3.7:** Početna scena

### 3.5.2. Glavna scena

Kada igrač uđe u glavnu scenu, prikazanu na slici 3.8, djedica je već u zraku i može početi dostavljati poklone. Scena se sastoji od 2D elemenata grafičkog sučelja, 3D objekata, kamere, svjetla, efekata i skripti.

#### Graf scene

Slijedi opis elemenata u sceni.


**Slika 3.8:** Glavna scena

- *FX Snow* - čestični sustav snijega.
- *Directional Light* - glavno svjetlo u sceni.
- *Default* i *Scene Post-Process* - sloj koji radi obradu slike nakon što se svi objekti iscrtaju. Služi kako bi se postigla bolja atmosfera u igri i realniji prikaz. Korišteni su sljedeći efekti: TAA, automatska ekspozicija (engl. *auto exposure*), mapiranje tonova (engl. *tonemapping*), zamućivanje svjetlećih elemenata (engl. *bloom*), balans bijele boje (engl. *white balance*), efekt vinjete (engl. *vignette*), ambijentalno zaklanjanje i volumetrijska magla.
- *Plane* - led u obliku kruga u sredini mape ispod drvca.

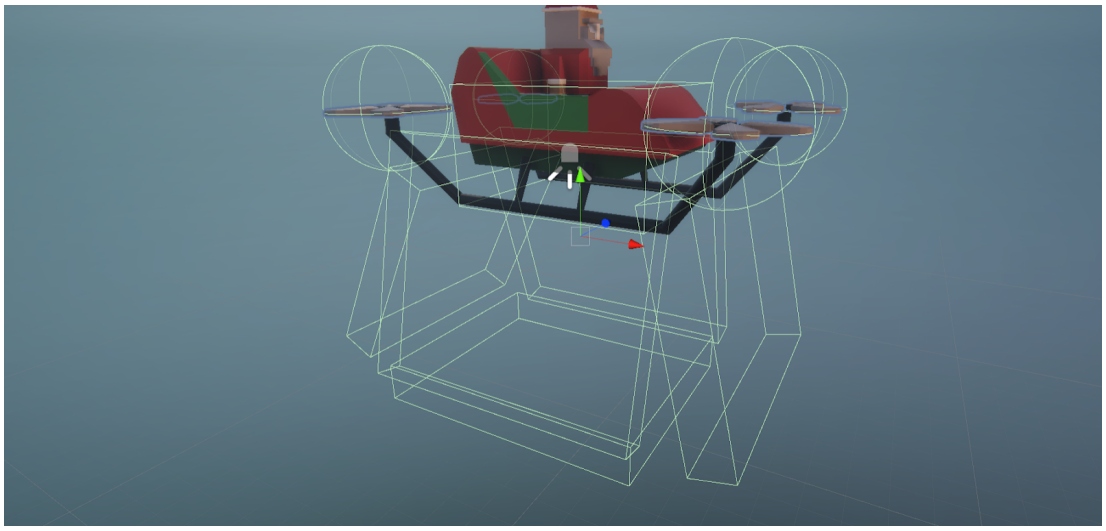
– *Chimneys* - set dimnjaka koji sadrži *ChimneyController.cs* skriptu i 7 dimnjaka svake kuće koji sadrže sudarač u obliku kugle, skriptu *Chimney.cs* i animaciju dima.

– *Scene* - sastoji se od glavnih 3D modela u sceni, preuzetih sa [14]: snijeg, lampe, kuće, božićno drvce, kamenje i stabla.

– *Main Camera (1)* - kamera koja prati saonice i sadrži *Cinemachine Brain* komponentu.

– *Santas Drone* - saonice izrađene od više dijelova, navedenih u nastavku.

  - Model saonica - model sadrži trup, lopatice i kavez za poklone. Saonice na sebi imaju četiri osnovna sudarača na svakoj lopatici kako bi se ograničilo kretanje unutar granica scene. Dodani su i sudarači kaveza, koji su u posebnom sloju te međudjeluju samo s poklonima. Kavez se sastoji od 6 sudarača posloženih tako da čine šuplji kvadar, kao što se može vidjeti na slici 3.9. Donji sudarač je otvoren za sve novopristigle poklone kako bi oni mogli ući u kavez, a čim uđu u kavez odmah im se sloj promjeni na sloj koji se sudara s kavezom kako poklon ne bi mogao pobjeći van jednom kada je pokupljen. Kada ih je potrebno dostaviti, pokloni ponovno mijenjaju svoj sloj kako se ne bi sudarali s kavezom i kako bi mogli izaći iz njega.
  - Model Djedice - sjedi u saonicama.
  - Četiri ikone poklona - predstavljaju broj trenutno pokupljenih poklona.
  - Sudarač - služi za detekciju poklona i dimnjaka. Saonice osim sudarača kaveza sadrže i jedan sudarač u obliku kapsule koji detektira sudar s poklonom ili dimnjakom te pokreće mijenjanje boje reflektorskog svjetla koje se nalazi ispod saonica. Sudarač je prikazan na slici 3.10.
  - Reflektorska svjetla:
    * Crveno svjetlo sugerira igraču da je iznad dimnjaka iz kojeg izlazi dim i da ako izbaci poklone neće dobiti bodove, a pokloni



**Slika 3.9:** Sudarači kaveza

**Slika 3.10:** Sudarač za detekciju poklona i dimnjaka

će izgorjeti. Primjer pozicije u kojoj se aktivira crveno svjetlo može se vidjeti na slici 3.11a.

* Zeleno svjetlo sugerira igraču da je iznad poklona i da ima dovoljno mjesta u saonicama, pa ga može pokupiti. Također označava trenutak kada je igrač iznad dimnjaka iz kojeg ne izlazi dim i ima poklone u saonicama koje je moguće dostaviti, kao što je slučaj na slici 3.11b.
* Bijelo svjetlo daje povratnu vizualnu informaciju o tome gdje je trenutno područje koje saonice obuhvaćaju i koliko je blizu dimnjaku ili poklonu, što je vidljivo sa slike 3.11c.

– *BorderSphere* - poluprozirna plava sfera koja predstavlja vizualnu i logičku granicu do koje se dron može kretati. Kugla koristi *MeshCollider* komponentu koja svakom poligonu kugle pridružuje sudarač. Kugli su normale obrnute u *Blender*-u kako bi *MeshCollider* komponenta na sferi mogla detektirati sudare drona s unutrašnjom stranom kugle.

– *Present_Black, Present_Blue, Present_Green* - početni pokloni koje igrač može pokupiti odmah na početku igre.

– *PresentSpawner* - sadrži skriptu *PresentSpawner.cs* koja kontrolira stvaranje poklona.

– *PlayCanvas* - grafičko sučelje koje sadrži sve 2D elemente poput brojača vremena, broja poklona i slobodnih dimnjaka. Sadrži i skriptu *CanvasManager.cs* koja kontrolira sve 2D elemente. Tipografija koja je korištena kroz sučelje videoigre u svim tekstualnim elementima je *Moon 2.0* i preuzeta je sa [15]. Izgled tipografije vidljiv je na slici 3.12.

  Za prikaz teksta u igri korišten je *TextMeshPro* dodatak koji omogućava prikaz vektorske grafike i teksta. Tako je osigurano da su elementi grafičkog korisničkog sučelja uvijek oštri bez obzira na izlaznu rezoluciju ekrana na kojemu se prikazuje videoigra. Svi grafički elementi u glavnoj sceni su vidljivi na slici


**(a)** Crveno svjetlo


**(b)** Zeleno svjetlo


**(c)** Bijelo svjetlo



**Slika 3.11:** Reflektorska svjetla


**Slika 3.12:** Tipografija *Moon 2.0*

3.13 te su popisani u nastavku.


**Slika 3.13:** Grafički elementi na ekranu

- Broj dostavljenih poklona - nalazi se u gornjem lijevom kutu ekrana, uz ikonu poklona.
- Vrijeme preostalo do kraja igre - nalazi se u gornjem desnom kutu ekrana. Kod preostalih 30 sekundi vrijeme poprimi žutu boju, a pri 10 sekundi crvenu kako bi igraču bilo lakše isplanirati preostalo vrijeme.
- Broj slobodnih dimnjaka - nalazi se u donjem desnom kutu ekrana uz ikonu dimnjaka i označava koliko je dimnjaka slobodno. Dimnjak je slobodan ako iz njega ne izlazi dim. Jednom kada se poklon dostavi u dimnjak, on postane zauzet i dim krene izlaziti iz njega. Kada se u

sva tri dimnjaka dostave pokloni, oslobode se nova tri dimnjaka. Tri dimnjaka se biraju nasumično između svih dimnjaka, a jedini dimnjak koji ne može opet biti izabran je onaj koji je zadnji postao zauzet.

- Broj trenutno skupljenih poklona - predstavljen je crnim ikonama poklona na stražnjoj strani saonica. Broj ikona crnih poklona označava trenutni broj poklona koji su skupljeni i spremni za dostavu, a broj ikona sivih poklona označava broj slobodnih mjesta za poklone.

– *EventSystem* - služi za slušanje ulaznih događaja koje korisnik generira.

– *HighScoreCanvas* - sadrži *HighScoreTable* skriptu koja kontrolira učitavanje, spremanje i prikaz najboljih rezultata na kraju igre.

– *GameManager* - sastoji se od *RosConnector.cs* i *JoySubscriber.cs* skripti koje komuniciraju s *ROS*-om i primaju ulazni signal s gumba.

– *Cm vcam1* - *Cinemachine* kamera koja prati saonice uz zaglađivanje kretanja. Zaglađivanje je ostvareno pomoću granica koje su na slici 3.14 predstavljene plavim pravokutnikom. Žuta točka je cilj koji kamera prati. Postavljena je na dno saonica i kada se saonice pomaknu, kamera lagano počinje dostizati konačnu poziciju drona kako bi se izbjegli nagli trzaji kamere.

### 3.5.3. Scena s prikazom ostvarenog rezultata

Po isteku vremena prikazuje se rezultat i traži se od igrača da unese svoje ime. Scena je prikazana na slici 3.15. Nakon unosa imena otvara se završna scena s prikazom najboljih rezultata.

### 3.5.4. Završna scena

U završnoj sceni prikazani su najbolji rezultati, kao što se može vidjeti na slici 3.16. Zadnji rezultat koji je igrač ostvario označen je sivim pravokutnikom. Kada igrač pritisne bilo koji gumb vraća se na početnu scenu.


**Slika 3.14:** Zaglađivanje kretanja


**Slika 3.15:** Prikaz rezultata i unos imena igrača


**Slika 3.16:** Najbolji rezultati

## 3.6. Komunikacija s *ROS*-om

Igrač upravlja dronom pritiskom gumba, a kako se informacije o stanju gumba nalaze na drugom računalu unutar *ROS* okruženja potrebno je prikupiti te informacije preko mreže. Igra je povezana s *ROS*-om preko *Rosbridgea* koji komunicira protokolom *WebSocket*. Korištena je *ROS#* biblioteka, čija je dokumentacija dostupna na [16]. Navedena biblioteka nudi jednostavne metode za povezivanje sa *WebSocket* serverom *ROS*-a i za pretplatu na poruke.

Poruka koja se šalje iz *ROS*-a u igru sadrži niz brojki (nule i jedinice) kao *string*(niz simbola, brojeva i slova) tip podatka, npr. "01001". Svaki broj predstavlja jedan gumb. Nula označava da gumb nije pritisnut, a jedan da je pritisnut. Svaki gumb ima svoje mjesto u nizu: prvi broj označava gumb za kretanje unaprijed, drugi broj predstavlja gumb za kretanje unatrag, treći broj predstavlja gumb za rotaciju udesno, četvrti broj predstavlja gumb za rotaciju ulijevo, a peti broj predstavlja gumb sa skupljanje i dostavu poklona.

## 3.7. Skripte

U nastavku će biti predstavljene funkcije skripti koje su razvijene u procesu izrade videoigre.

**CanvasManager.cs**

Skripta kontrolira grafičke elemente na ekranu: broj dostavljenih poklona i broj trenutno skupljenih poklona. Zadužena je za promjenu boje poklona na saonicama - crna ikona označava skupljeni poklon, a siva ikona slobodno mjesto za poklon u saonicama.

**Chimney.cs**

Sadrži stanje dimnjaka (dimnjak s dimom illi dimnjak bez dima), njegov identifikator i referencu na dim. Pali i gasi svjetla na pripadajućoj kući signalizirajući igraču da u toj kući ljudi spavaju ako svjetla ne gore i ako iz dimnjaka ne izlazi dim te je moguće dostaviti poklon. Ako su svjetla upaljena i dim izlazi iz dimnjaka nije moguće dostaviti poklon (bit će uništen u slučaju da se ipak dostavi).

**ChimneyController.cs**

Kontrolira sve dimnjake. Kada se dimnjak pali, uključuje se dim i mijenja se stanje dimnjaka. Kada se dimnjak gasi, gasi se dim i mijenja se stanje dimnjaka. Skripta također ažurira grafički prikaz slobodnih dimnjaka na ekranu. Igrač na početku ima tri slobodna dimnjaka i kako dostavlja poklone u njih, dimnjaci se pale kako bi se omogućila dostava u isti dimnjak više puta zaredom. Kada dostavi poklone u sva tri dimnjaka, nasumično se biraju nova tri dimnjaka koja se oslobode. Pri tome se pazi da među ta tri dimnjaka ne bude onaj dimnjak u koji je zadnji ubačen poklon.

**CustomInput.cs**

Skripta objedinjuje kontrole za tipkovnicu, igraću palicu te silikonske gumbe. Uz pomoć ove skripte moguće je kontrolirati saonice sa svim ulaznim jedinicama u isto vrijeme.

**DroneController.cs**

Kod kretanja unaprijed i unatrag skripta rotira saonice oko *z*-osi i dodaje silu u smjeru lokalne *y*-osi, a kod rotacije rotira saonice oko lokalne *y*-osi. Čita ulaz s igraće palice, tipkovnice i *Rosbridgea* te objedinjuje ulaze u jedan. Pritiskom tipke *F1* letjelicu kontroliraju *ROS* poruke, a pritiskom tipke *F2* kontrolu preuzimaju tipkovnica i igraća palica.

**GameSceneUI.cs**

Skripta se nalazi u početnoj sceni i čita ulaz s tipkovnice. Ako je pritisnuta tipka *ESC*, izlazi iz igre, a ako je pritisnuta bilo koja druga tipka, učitava se glavna scena i počinje igra.

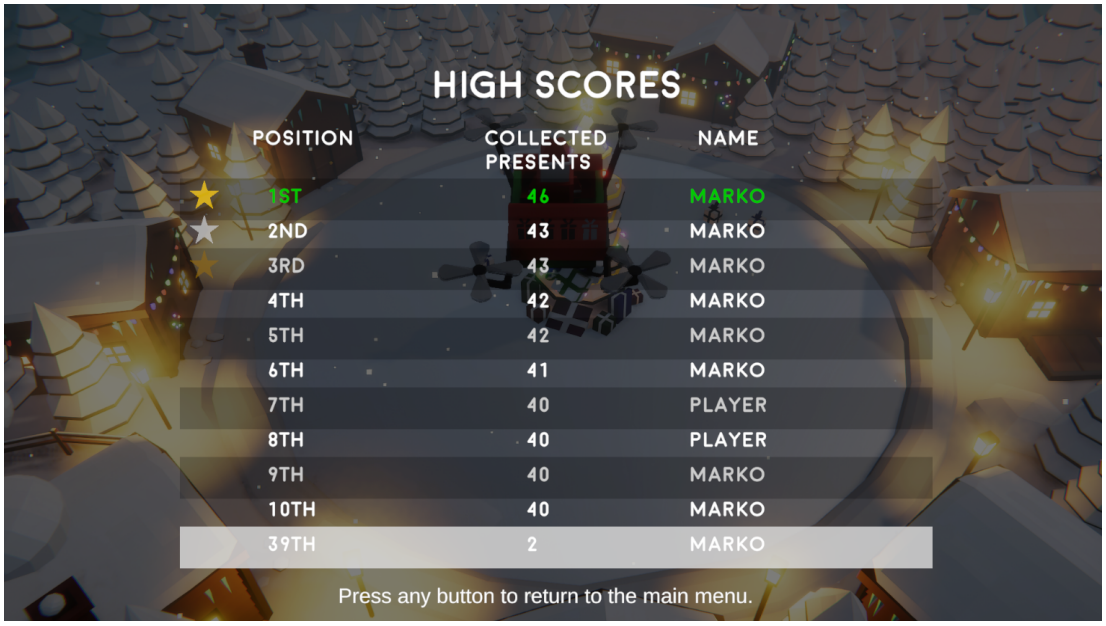
**HighScoreTable.cs**

Skripta sadrži tri razreda:

- *HighScoreEntry* - sadrži rezultat u obliku broja (*int*) dostavljenih poklona i imena igrača (*string*) koji je ostvario taj rezultat,
- *HighScores* - sadrži listu *HighScoreEntry* objekata,
- *HighScoreTable* - služi za zapisivanje rezultata u datoteku na kraju igre, za čitanje rezultata iz datoteke na završnoj sceni te prikaz tih pročitanih rezultata u obliku tablice. Lista rezultata se serijalizira i pretvori u *JSON* format te se zapisuje u datoteku. Pri čitanju datoteke koristi se deserijalizacija za pretvaranje *JSON* objekta u listu *HighScoreEntry* objekata. Datoteka se sprema u lokalnu datoteku na disku. Primjer izgleda datoteke:

```
{"highscoreEntryList":[
{"score":32,
"name":"Filip"},
{"score":29,
"name":"Ivan"},
{"score":41,
"name":"Nikola"}]}
```


Igračev red u tablici je dodatno označen sa sivim pozadinskim pravokutnikom. U slučaju da je rezultat među prva tri, kao na slici 3.15, sa strane se nalazi zlatna, srebrna ili brončana zvijezda. Kada igrač nije među najboljih 10 rezultata, kao na slici 3.17, njegov rezultat se prikazuje ispod svih rezultata s mjestom na kojem je završio. Kada igrač pritisne bilo koju tipku, igra se vraća na početni ekran.




**Slika 3.17:** Slučaj kada igrač nije među najboljih 10

**InputPlayerName.cs**

Skripta je pretplaćena na događaj završetka unošenja imena po završetku partije. Ako igrač ne unese ime nego samo pritisne tipku *Enter*, rezultatu će biti dodijeljeno generičko ime *Player*. Kada igrač završi s unosom imena, uključuje se tablica s najboljim rezultatima kako bi igrač mogao usporediti svoj rezultat s rezultatima ostalih igrača.

**JoySubscriber.cs**

Skripta koja implementira sučelje *UnitySubscriber<MessageTypes.Std.String>* i njegovu metodu *ReceiveMessage(MessageTypes.Std.String message)*. Klasa sadrži polje brojeva koje sadrži trenutno pritisnute gumbe (1 predstavlja pritisnuti gumb, 0 nepritisnuti gumb). Poruka se prima u obliku niza znakova (npr. ”01000”), koji se pretvara u niz brojeva. Skripta *DroneController.cs* potom čita te ulaze i kontrolira saonice.

**Loader.cs**

Statička klase koja sadrži enumerator tipa *Scene* koji ima dvije moguće vrijednosti, *Game* i *MainMenu*, te funkciju *Load(Scene)* koja učitava onu scenu koja je zadana u argumentu poziva. Služi za tranziciju iz početne scene u glavnu scenu na početku igre te iz glavne scene u početnu scenu na kraju igre. Klasa *GameSceneUI* na početku igre prati želi li igrač započeti igru i onda poziva *Load(Game)*, a klasa *HighScoreTable* na kraju igre prati želi li se igrač vratiti na početni ekran i tada poziva *Load(MainMenu)*.

**PresentCollector.cs**

Skripta se nalazi na saonicama i sadrži sudarač u obliku kapsule s kojim detektira sudare s poklonima i dimnjacima. Skripta kontrolira i reflektorsko svjetlo na dronu i mijenja mu boju ovisno o tome s kojim objektom je u sudaru. Kada su saonice iznad poklona i u saonicama još ima mjesta, pali se zeleno svjetlo. Ako je igrač u tom trenutku pritisnuo gumb za skupljanje poklona, poklonu se dodaje skripta *PullTowardsDrone.cs* i kao željena pozicija mu se postavlja pozicija saonica. *PullTowardsDrone.cs*

skripta tada dodaje silu u smjeru saonica te se poklon počinje kretati prema saonicama. Kada su saonice iznad dimnjaka iz kojeg ne izlazi dim i ako trenutno ima pokupljenih poklona, pali se zeleno svjetlo. Ako je igrač tada pritisnuo gumb za dostavu poklona, poklon se dostavlja tako da se poklonu i njegovoj skripti *PullTowardsDrone.cs* kao željena pozicija postavi lokacija dimnjaka i prebaci se u drugi sloj tako da može izaći iz kaveza. *PullTowardsDrone.cs* skripta tada dodaje silu u smjeru željene pozicije kako bi se poklon ubacio u dimnjak. Kada se poklon dostavi, povećava se rezultat za jedan, gasi se zeleno svjetlo te iz dimnjaka počinje izlaziti dim. Kada se saonice nalaze iznad dimnjaka iz kojeg izlazi dim i trenutno ima pokupljenih poklona, pali se crveno svjetlo kako bi se upozorilo igrača da ne smije dostaviti poklone jer će biti uništeni. Ako ipak dostavi poklone, oni će se uništiti i igrač neće dobiti bodove.

**PresentSpawner.cs**

Stvara poklone oko drvca u nasumičnom intervalu između 2.6 i 3 sekunde. Pokloni se stvaraju u krugu oko drvca, između 3 i 7 metara udaljeni od središta drvca, na visini od jednog metra. Postoje 3 vrste poklona: crni, plavi i zeleni, te se nasumično stvara jedan od njih. Pokloni su prikazani na slici 3.18.

**Slika 3.18:** Tri vrste poklona

**PullTowardsDrone.cs**

Skripta se nalazi na poklonu i dodaje silu poklonu u smjeru željene pozicije. Kada se sudari s dimnjakom, uništi se.

**RosConnector.cs**

Skripta iz *ROS#* biblioteke koja se nalazi u glavnoj sceni na *GameManager* objektu. Njoj je predana IP adresa računala na kojemu je pokrenut *ROS*. Skripta na startu igre pokušava 10 sekundi uspostaviti *WebSocket* konekciju stvaranjem *RosSocket* objekta. U slučaju da se konekcija nije uspostavila igra će i dalje raditi, samo bez mogućnosti kontrole od strane *ROS*-a i gumba od želatine. *RosSocket* objekt prima i šalje sve poruke.

**Timer.cs**

Skripta služi kao štoperica koja odbrojava vrijeme. Igrač ima dvije minute da dostavi što više poklona. Vrijeme se prikazuje u gornjem desnom dijelu ekrana. Vrijeme je prikazano u formatu MM:SS.mm, gdje MM označava broj minuta, SS broj sekundi a mm broj stotinki preostalih do kraja runde. Kada štoperica dođe do 30 sekundi, boja teksta se mijenja u žutu, a kada dođe do 10 sekundi, u crvenu kako bi igrač lakše isplanirao zadnji dio runde. Ako igrač pritisne tipku *ESC* izlazi se iz igre. Ako pritisne *F5* štoperica se pauzira, a ako pritisne *F5* još jednom štoperica nastavlja s izvođenjem. Pauziranje se koristilo prilikom objašnjavanja logike igre djeci i igranja s gumbima od želatine jer cilj nije bio da se s gumbima ostvare najbolji rezultati, nego da se demonstrira njihova funkcionalnost. Po isteku vremena letjelici se isključuju kontrole i gase svjetla. Prikazuje se ekran vidljiv na slici 3.15, gdje je prikazan ostvaren rezultat i od igrača se zahtijeva unos imena. Kada igrač završi s unosom, prikazuje se tablica s najboljim rezultatima i njegova pozicija na toj tablici.

## 3.8. Kontrole

Prilikom upravljanja videoigrom korištenjem tipkovnice, tipke imaju sljedeće funkcije:

- W ili strelica gore - kretanje unaprijed
- S ili strelica dolje - kretanje unatrag
- A ili strelica lijevo - rotacija ulijevo
- D ili strelica desno - rotacija udesno
- Razmaknica - skupljanje i dostava poklona
- F1 - upravljanje saonicama pomoću gumba od želatine
- F2 - upravljanje saonicama pomoću tipkovnice ili igraće palice
- F5 - pauziranje tajmera
- ESC - izlazak iz igre

Kontrole na igraćoj palici: lijeva gljiva služi za kretanje a gumb *X* za skupljanje i dostavu poklona.

## 3.9. Zaključak o uspješnosti igre

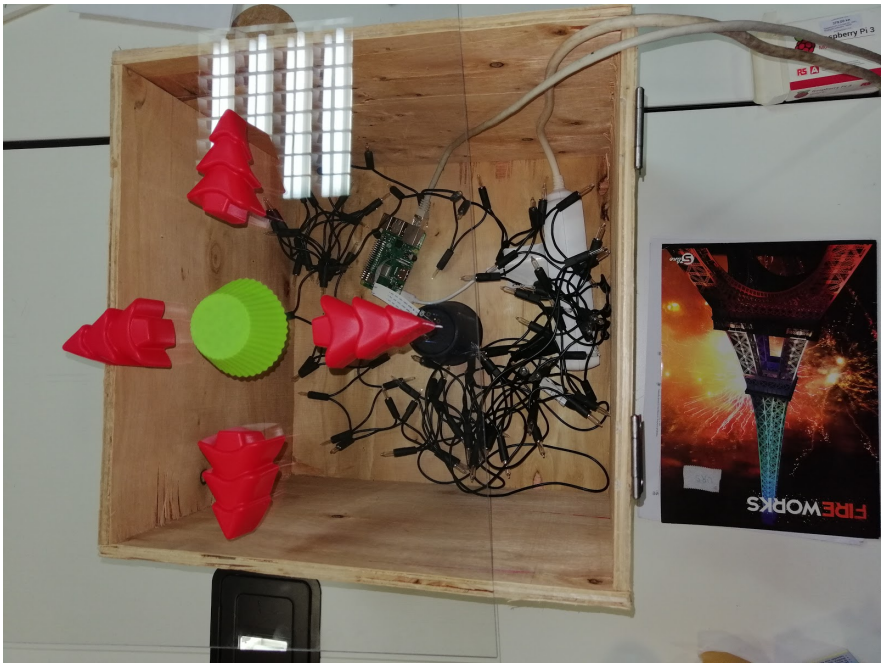
Igru su igrali igrači svih dobnih skupina, dolazili su studenti u pauzi između predavanja, profesori, asistenti, djeca svih dobnih skupina, čak i njihovi roditelji. Igru je odigralo preko 200 igrača, jedan dio na velikom televizoru s igraćom palicom, a drugi dio na manjem monitoru s gumbima od želatine. Tablica najboljih igrača je na kraju dana imala 236 rezultata pojedinačnih igara. Nekim igračima se toliko svidjelo da su dolazili probati i više puta. Ponekad bi došlo i do rivalstva među prijateljima, kao u slučaju dvojice prijatelja koji su se natjecali tko će imati veći rezultat, pa je u trenutku upisa rezultata jedan isključio drugome igru pritiskom na tipku *ESC* kako bi spriječio njegov upis u tablicu, budući da je ostvario bolji rezultat od njega samoga.

# 4. *Soft* robotika - izgled sustava i upravljanje videoigrom

Upravljačke kontrole za videoigru izrađene su od želatine. Inspiracija za njihovu izradu bio je projekt američkih učenika srednje škole Haverford School u Pennsylvaniji, dostupan na [17] temeljen na znanstvenom istraživanju [3]. Projekt se bavio razvijanjem biorazgradive *soft* robotike koja je kompatibilna s ljudskim tijelom. Detaljan proces izrade upravljačkih kontrola je opisan u potpoglavlju 4.1.

Sustav upravljanja smješten je unutar drvene kutije, čija je jedna stranica zamijenjena pleksiglasom. Unutar kutije nalazilo se *Raspberry Pi* računalo, model *B*, sa priključenom kamerom *Raspberry Pi Camera V2*. Izgled kutije može se vidjeti na slici 4.1. LED traka unutar kutije služila je kao izvor svjetlosti, a crni papir iskorišten je za sprječavanje pojave odsjaja, kao na slici 4.1b. Na gornju, prozirnu stranu, smješteno je pet gumba od želatine. Pojedini gumb odgovarao je naredbi naprijed/natrag/rotacija ulijevo/rotacija udesno te skupljanje i dostava poklona. U skladu s božićnim duhom i vizualnim identitetom videoigre, vanjski izgled drvene kutije zamišljen je kao božićni poklon, što se može vidjeti na slici 4.1c.

Na *Raspberry Pi*-ju instalirani su *Ubuntu Mate* i *ROS*. Komunikacija između *Raspberry Pi*-ja i računala odvijala se preko *ROS Master*-a, što je detaljnije objašnjeno u potpoglavlju 4.2. i 4.3, gdje je opisan postupak obrade slike s kamere i arhitektura sustava. Naime, korištenjem slike s kamere detektiran je pritisak gumba, što je omogućilo bežično upravljanje videoigrom.




**(a)** Kutija u izradi

**(b)** LED traka i crni papir unutar kutije

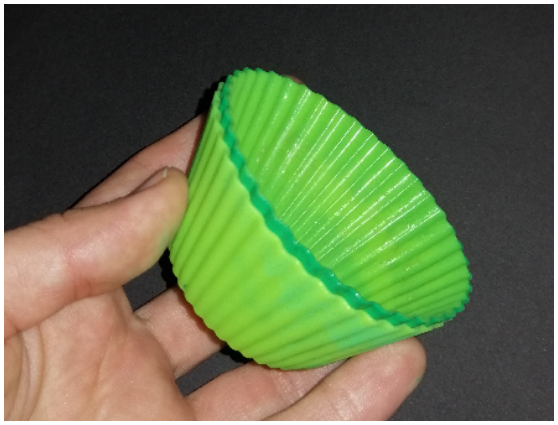
**(c)** Vanjski izgled kutije

**Slika 4.1:** Kutija korištena na radionici

## 4.1. Gumbi od želatine

Zbog svoje podatnosti, elastičnosti i jednostavnog rukovanja, gumbi su izrađeni od silikonskih kalupa, koji se mogu pronaći u svakoj trgovini mješovite robe, a kalupi su naknadno ispunjeni kuhanom želatinom u listićima. Korišteni materijali su prikazani na slici 4.2.


**(a)** Želatina u listićima

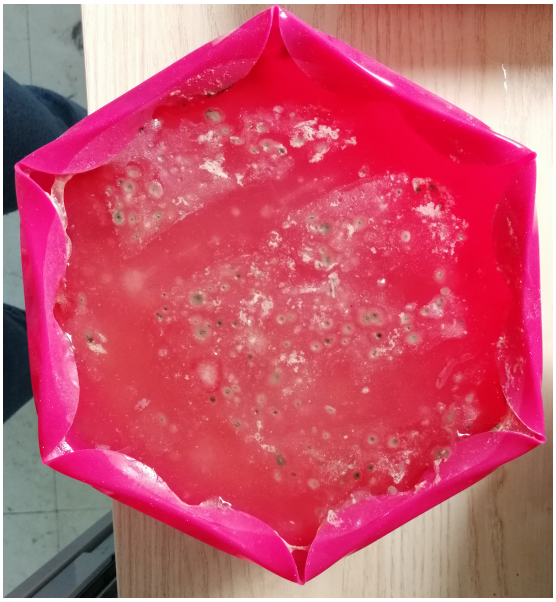


**(b)** Kalupi

**Slika 4.2:** Materijali korišteni pri izradi gumba

Prije svega, bilo je potrebno pronaći odgovarajući način pripreme i hlađenja želatine, što se ponajviše odnosi na pronalazak odgovarajućeg omjera vode i mase želatine u listićima, kako bi gumb bio dovoljno tvrd da izdrži pritiskanje prilikom upravljanja videoigrom, a istovremeno dovoljno mekan da računalni algoritam odgovoran za detekciju pritisnutog ili nepritisnutog gumba može lako detektirati promjenu. Nakon nekoliko neuspjelih pokušaja stvrdnjavanja želatine, poput onog prikazanog na slici 4.3, pronađen je odgovarajući omjer od 28,8 *g* (*cca* 12 listića želatine) na 100 *ml* vode.

Potrebna količina vode za punjenjenje pet silikonskih kalupa pomiješana je s listićima želatine prethodno namočenim u hladnoj vodi te je kuhana dok nije dobivena homogena smjesa. Tekućoj želatini je bilo potrebno, ovisno o temperaturi okoline,



**Slika 4.3:** Neuspjela želatina

otprilike tri sata za učvrščivanje. Nakon što se želatina ohladila, bilo je potrebno odstraniti nečistoće koje su se nakupile na površini te želatinu ponovno otopiti kako bi se dobila što prozirnija i homogenija smjesa. Cijeli postupak je prikazan na slici 4.4.

Nakon ponovnog topljenja, čista smjesa je uljevena u silikonske kalupe gdje je nakon otprilike tri sata hlađenja bila spremna za upotrebu. Različiti uzorci želatine prikazani su na slici 4.5. Konačni izgled gumba je prikazan na slici 4.6. Gumb je sastavljen od dva zelena kalupa, crnih markera i želatine. Crni markeri na unutrašnjosti kalupa napravljeni su uz pomoć posebne crne boje koja je namijenjena za premaz preko silikonskog materijala. Nakon što se boja osušila, u kalupe je ulivena tekuća želatina te je ostavljena na hlađenje. Markeri se u svim gumbima nalaze na jednakom razmaku kako bi se osigurala jednaka detekcija pritiska svih gumba.

**(a)** Kuhana i tekuća želatina

**(b)** Odstranjivanje nečistoća

**(c)** Čisti komadići želatine

**Slika 4.4:** Priprema želatine

**Slika 4.5:** Različiti uzorci ohlađene želatine u kalupima

**Slika 4.6:** Gumb s crnim markerima

## 4.2. Upravljanje videoigrom korištenjem izrađenih gumba

Cilj igre *Santa's Delivery Drone* je dostaviti što veći broj poklona u odgovarajuće kućice u određenom vremenu. Kako bi to bilo moguće ostvariti, potrebno je upravljati letjelicom, tj. njenim gibanjem u horizontalnoj ravnini, te, dodatno, prikupljati i dostavljati poklone, što podrazumijeva uporabu neke vrste kontrolera. U tu su svrhu korišteni prethodno opisani gumbi izrađeni od želatine. Kako bi se ostvarilo slobodno gibanje letjelice u horizontalnoj ravnini, a istovremeno zadržala jednostavnost upravljanja, odabrano je upravljanje korištenjem četiri gumba. Dva gumba predstavljaju gibanje letjelice naprijed, odnosno natrag u odnosu na kut gledanja kamere, dok dva gumba predstavljaju rotaciju letjelice oko vlastite osi. Posljednji, peti, gumb korišten je za prikupljanje i dostavljanje poklona.

Identično izrađeni gumbi, s po pet označenih markera unutar kalupa, raspoređeni su na površinu izrađenu od pleksiglasa koja se nalazi na gornjoj strani kutije. Dodatno je, ispod pleksiglasa, postavljena crna podloga s izrezanim rupama na mjestima gdje se nalaze gumbi s ciljem izolacije nevažnih informacija, odnosno s ciljem detektiranja i prepoznavanja samo mjesta na kojima se gumbi nalaze, kao što je opisano u uvodnom dijelu poglavlja. Unutar kutije smještena je kamera okrenuta prema gornjoj strani kutije na kojoj se nalaze gumbi. Kamera u realnom vremenu snima trenutno stanje gumba (jesu li pritisnuti) te informaciju prosljeđuje logici igre.

S obzirom da je kamera povezana s *Raspberry Pi* pločicom, koja je dodatno povezana *Ethernet* kabelom na računalo na kojem se izvršavaju algoritmi obrade slike, odabrana je rezolucija kamere koja daje odgovarajuće informacije, uz ne pretjerano velik broj piksela kako ne bi dolazilo do velikog kašnjenja u komunikaciji, a iznosi 1280x960. Računalo na kojem se izvršavaju algoritmi obrade slike dobiva oko 20 slika po sekundi te po završetku obrade prosljeđuje informacije o pritisku gumba igri, također frekvencijom od 20 *Hz*.

Prilikom pokretanja igre, osnovna pretpostavka je da korisnik ne drži pritisnut niti jedan gumb, budući da se inicijalno snimljena slika uzima kao referentna. Na temelju razlike referentne i trenutne slike moguće je odrediti koji je gumb trenutno pritisnut.

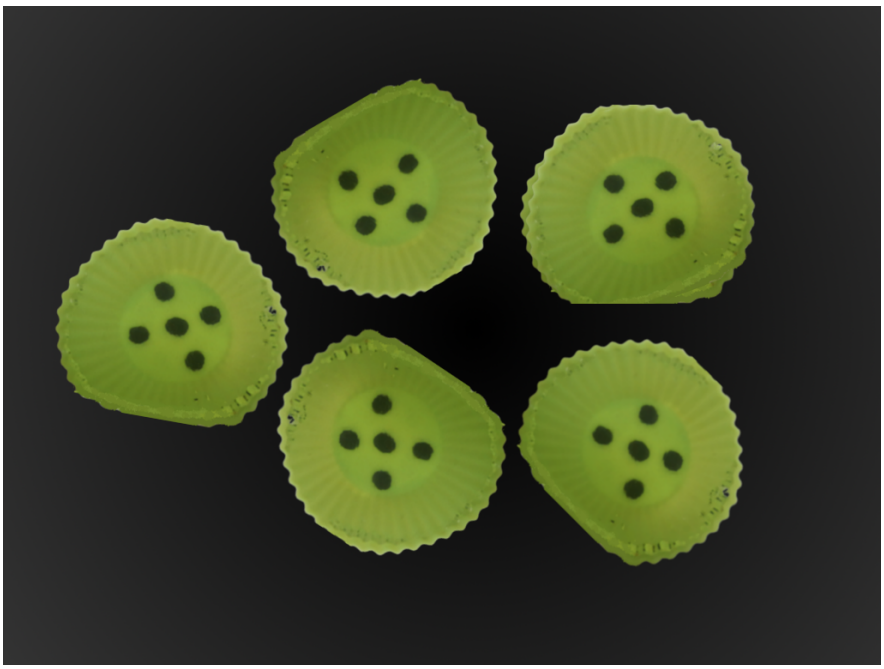
Pri pokretanju programa inicijaliziraju se potrebni *ROS* čvorovi i poruke potrebne za dohvaćanje podataka s kamere i prosljeđivanje informacija o statusu gumba igri. U tu se svrhu koristi tzv. pretplatnik (engl. *subscriber*) koji dobiva slike s kamere i tzv. objavljivač (engl. *publisher*) koji obrađene podatke prosljeđuje igri u obliku niza od pet brojeva, gdje svaki broj predstavlja jedan gumb, a može imati vrijednost 0 u slučaju ako gumb nije pritisnut ili 1 ukoliko je gumb pritisnut.

Tijekom dužeg vremena igranja igre dolazi do sve većeg pomaka gumba po pleksiglasu, što može rezultirati netočnim informacijama o pritisku gumba. Naime, razlika između početne i trenutne slike može biti posljedica spomenutog pomaka gumba, a ne pritiska gumba. Zbog toga je dodatno uveden parametar naziva *ready* čiju je vrijednost jednostavno postaviti tijekom izvršavanja program. Parametar *ready* omogućava resetiranje i ponovno postavljanje inicijalne slike za usporedbu. Prilikom pokretanja, vrijednost parametra je 0, što rezultira spremanjem prve (inicijalne) slike za buduću usporedbu i postavljanjem vrijednosti parametra na 1, sve do njegove manualne promjene u 0, kada se inicijalna slika resetira i ponovno postavlja.

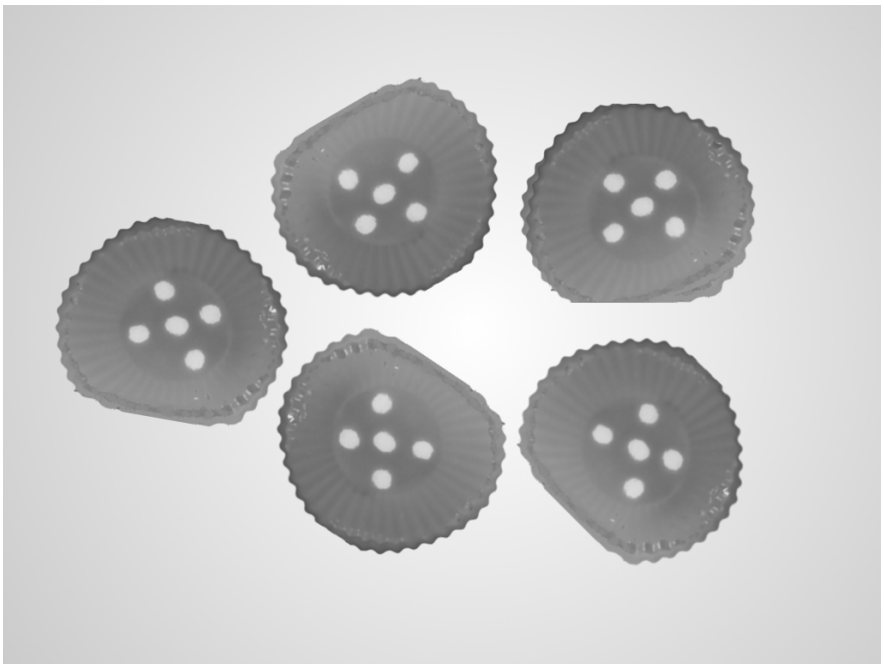
Budući da slike kao takve sadrže velik broj informacija, od kojih je velika većina nepotrebna, potrebno je iz njih izvući samo bitne informacije. U tu se svrhu koriste brojni procesi i algoritmi obrade slika, čija je implementacija iznimno olakšana uporabom *OpenCV* proširenja [18]. Riječ je o visoko optimiziranom i iznimno velikom skupu programskih funkcija korištenih za obradu slike, od kojih su neke korištene u procesu izvlačenja informacije o pritisku gumba.

Kamera zadužena za snimanje stanja gumba snima sliku u klasičnom formatu slike u boji. Takva se slika sastoji od tri kanala, odnosno svaki piksel slike se sastoji od tri brojčane vrijednosti u intervalu od 0 do 255, koje prikazuju intenzitet crvene, zelene i plave boje. Primjer takve slike je slika 4.7.

Budući da boja nije informacija koja je važna za određivanje prikazuje li pojedina slika pritisnut gumb ili ne, potrebno ju je filtrirati. To se ostvaruje na način da se sliku pretvori u crno-bijelu sliku, tj. sliku s 1 kanalom s vrijednostima od 0 do 255 koje predstavljaju intenzitet crne boje. Rezultat takvog procesa prikazan je slikom 4.8.



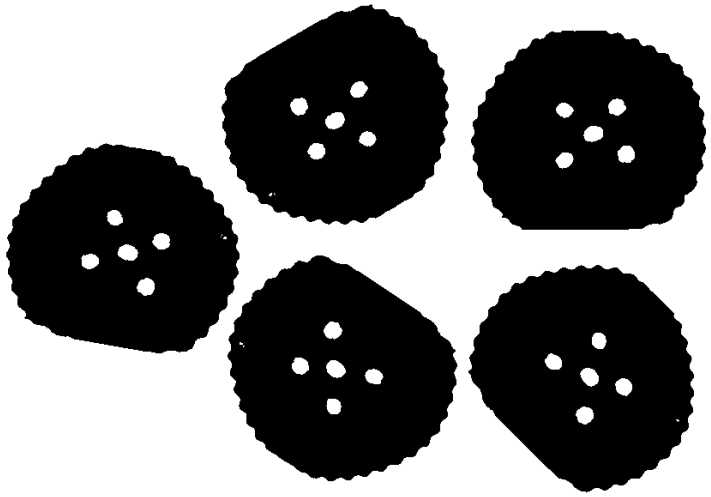
**Slika 4.7:** Neobrađena slika s kamere



**Slika 4.8:** Slika s kamere pretvorena u crno-bijelu

Eksperimentalno je utvrđeno da samo pretvaranje slike u crno-bijelu nije dovoljno za određivanje položaja pojedinih markera. Iz tog je razloga iz slike bilo potrebno izvući dodatne informacije o mjestima na kojima se nalaze gumbi, odnosno markeri u njima. Budući da su korišteni gumbi zelene boje, dok je sva okolna pozadina bila u različitim nijansama crne, sive i nekoliko žutih točaka uzrokovanih osvjetljenjem, iz slike su izlučena mjesta gdje su iznosi piksela zelene boje iznad određenog praga. To je ostvareno pretvorbom slike s kamere u trokanalnu HSV (engl. *hue, saturation, value* - ton, zasićenje, svjetlina) sliku za koju je poznato te je dodatno eksperimentalno određeno da se vrijednosti zelene boje od interesa nalaze u rasponu od (80, 0, 0) do (1, 255, 255). Na taj je način dobivena maska, tj. slika na kojoj su određeni pikseli različiti od 0 i predstavljaju mjesta zelene boje. Takva maska primijenjena je na crno-bijelu sliku s ciljem još preciznijeg izdvajanja i naglašavanja mjesta na kojima se nalaze gumbi. Dodatno je primijenjen algoritam praga koji funkcionira na način da piksele čije su vrijednosti iznad određene razine postavi na maksimalnu vrijednost, dok one ispod razine postavlja na nulu. Eksperimentalno je određen prag od 145, čime je dobivena slika 4.9. Sa slike je jasno vidljivo da su uspješno izolirani gumbi, a osim toga je napravljena jasna razlika između markera u gumbima i samih gumba, koji su na prvoj slici tamno zeleni, odnosno zeleni.

U tako dobivenoj slici, kao posljedica pritiskanja gumba i promjena u osvjetljenju, dolazi do raznih neželjenih pojava. Jedna od njih je promjena iznosa pojedinih piksela, što može dovesti do pogrešnog detektiranja pritiska gumba. Iz tog je razloga nad slikom potrebno provesti dodatna poboljšanja s ciljem što veće robusnosti. Konačni cilj takvog postupka je dobiti sliku na kojoj su samo točke koje predstavljaju markere pojedinih gumba bijele boje, dok su svi ostali dijelovi slike crni, naročito u slučaju pritiska gumba ili promjene u osvjetljenju. To je ostvareno na način da je nad slikom prvo provede algoritam ispune (engl. *floodfill*) čija je svrha obojati piksele iste ili slične vrijednosti u određenoj okolini u istu boju kako bi se dobila što glađa slika. Proces je proveden korištenjem programske funkcije unutar *OpenCV*-a, pod nazivom *morphologyEx*, koja primjenom matematičke morfologije ostvaruje izglađivanje slike,



**Slika 4.9:** Crno-bijela slika nakon primjene maske za izdvajanje zelene boje

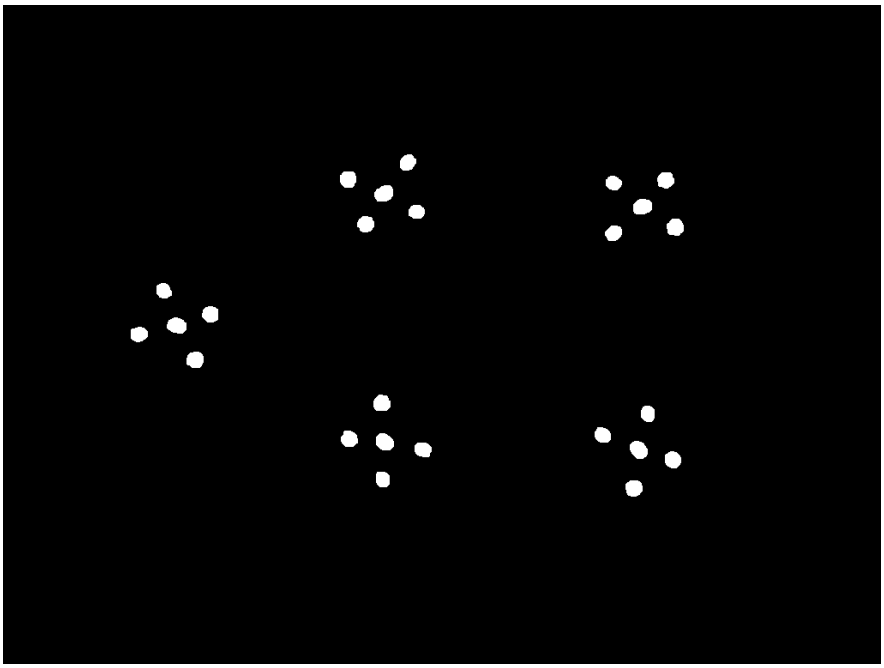
prikazano slikom 4.10.

Konačni rezultat slijedne primjene opisanih algoritama obrade slike s ciljem ekstrakcije piksela (područja) od interesa prikazan je slikom 4.11.

Rezultat je crno-bijela slika gdje su svi pikseli od interesa, odnosno mjesta na kojima se nalaze markeri svakog pojedinog gumba, bijele boje. Takva slika, u odnosu na inicijalnu, i dalje je predstavljena velikim brojem piksela nad kojima je iznimno teško raditi usporedbu i detektirati pritisak pojedinog gumba. Najrobusnije i najpreciznije rješenje dobiveno je korištenjem centroida svakog markera za čiji je izračun potrebno koristiti momente. Moment slike [19] predstavlja određeni otežani prosjek ili funkciju intenziteta piksela slike uobičajeno odabranih tako da imaju neko svojstvo ili interpretaciju. Moment se ne odnosi isključivo na cijelu sliku, već se može koristiti i za izračun određenih područja od interesa unutar slike.

Za određivanje centroida svakog markera, koji će se kasnije koristiti za usporedbu i detekciju pritiska gumba kojemu marker pripada, prvo je potrebno detektirati markere. U sklopu *OpenCV*-a postoji funkcija za detekciju kontura [19]. Konture predstavljaju područja u slici u kojima dolazi do nagle promjene intenziteta osvijetljenosti, odnosno

**Slika 4.10:** Primjena matematičke morfologije



**Slika 4.11:** Slika nakon obrade

nagle promjene vrijednosti piksela slike. Takva područja u ovom slučaju predstavljaju sami rubovi markera, budući da u tim pikselima dolazi do nagle promjene vrijednosti piksela iz bijele u crnu, odnosno brojčano iz 255 u 0. Kako prilikom pritiska gumba može doći do pojave novih bijelih područja koja ne predstavljaju markere, a mogu se pojaviti zbog nesavršenosti algoritma obrade i osjetljivosti na razne artefakte unutar želatine od koje su gumbi napravljeni, potrebno je uvesti dodatnu mjeru s ciljem povećanja robusnosti. To je ostvareno na način da se prilikom detekcije kontura na slici u obzir uzimaju samo konture čija se površina u pikselima (područje koje kontura obuhvaća) nalazi u rasponu od 150 do 900 piksela. Taj je iznos eksperimentalno određen ispitivanjem različitih pritisaka gumba prilikom kojih uslijed pritiskanja želatine, tj. gumba, dolazi do smanjivanja površina markera koje kamera vidi. Iznos je i dalje dovoljno velik te osigurava izbjegavanje detekcije neželjenih piksela koji se mogu pojaviti. Za svaku od kontura zatim se izračunava njezin centroid:

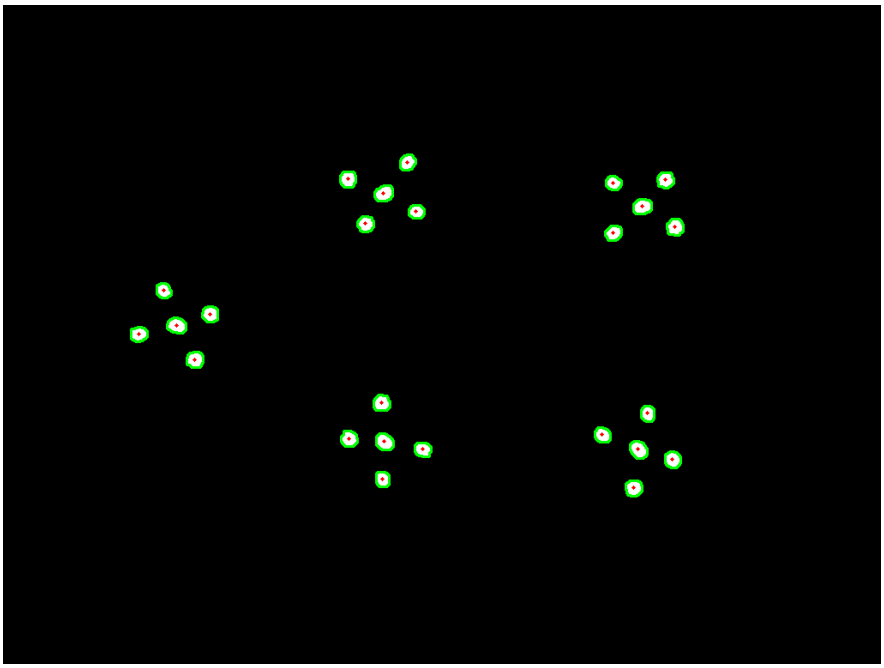
$$\{\overline{x}, \overline{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\} \tag{4.1}$$

gdje $M_{10}$, $M_{01}$ i $M_{00}$ predstavljaju vrijednosti momenta konture [19]. Momenti karakteriziraju oblik područja. Ako $(x, y)$ predstavlja redatk i stupac piksela s težinom 1 unutar željenog područja, moment tog područja se definira na sljedeći način:

$$M_{kj} = \sum x^k y^j \tag{4.2}$$

Na slici 4.12 zelenom bojom označene su detektirane konture, dok su crvenom bojom prikazani izračunati centroidi svake. Na taj su način iz slike dobivene informacije o pozicijama markera u koordinatnom sustavu slike, predstavljene u obliku koordinata (x, y), koje će se koristiti za detekciju pritiska pojedinog gumba. Cijeli postupak prikazan je i pseudokodom koji je dan u nastavku.

```
transformacija RGB slike u crno-bijelu
transformacija crno-bijele slike u HSV sliku
eksperimatalno određivanje maske
primjena maske i algoritma praga na crno-bijelu sliku
primjena algoritma ispune na crno-bijelu sliku
izračunaj konture iz slike
za svaku konturu c:
    izračunaj momente M_00, M_10, M_01
    ako je M_00 != 0 i 150 < površina(c) < 900:
        c_X = M_10/M_00
        c_Y = M_01/M_00
        dodaj (c_X, c_Y) u polje centroida
vrati centroide
```



**Slika 4.12:** Obrađena slika s prikazanim konturama markera i pripadnim centroidima

Kao što je već rečeno, prilikom pokretanja programa, pretpostavka je da niti jedan gumb nije pritisnut, te se za prvu sliku spremaju inicijalne pozicije markera. S obzirom da se za upravljanje koristi pet gumba, potrebno je odrediti koji marker pripada kojem

gumbu. Kamera i gumbi uvijek se nalaze na identičnoj poziciji što pojednostavljuje pridruživanje markera njihovim gumbima. Eksperimentalno je utvrđeno da se markeri koji pripadaju gumbu za prikupljanje/dostavljanje darova nalaze na pozicijama čija je vrijednost *x*-koordinate manja od 310 u koordinatnom sustavu slike. Za gumb desno određeno je da se markeri nalaze na vrijednostima većim od 700 za *x*-koordinatu te većim od 400 za *y*-koordinatu markera, za gumb lijevo vrijednosti *x*-koordinate manje su od 650, a *y*-koordinate manje od 370, za gumb gore vrijednosti *x*-koordinate veće su od 650, a vrijednosti *y*-koordinate manje su od 370, dok su za gumba dolje vrijednosti *x*-koordinate manje od 650, a *y*-koordinate veće od 400. Na taj su način svakom gumbu pridružene inicijalne pozicije njegovih markera. Isti se postupak ponavlja za svaku novopristiglu sliku za koju se određuju pozicije svih pet markera svakog gumba.

U trenutku pristizanja nove slike započinje algoritam obrade koji iz pristigle slike opisanim postupkom izlučuje informacije o pozicijama markera svakog gumba. Kako bi se odredilo je li pojedini gumb pritisnut ili ne, vrši se usporedba trenutnih pozicija markera i pozicija markera inicijalne slike (kada niti jedan gumb nije bio pritisnut). Algoritam detekcije pritiska funkcionira na sljedeći način. Za svaki gumb, prolazi se kroz sve njegove markere. Za svaki se marker trenutne slike pronalazi najbliži marker inicijalne slike po *Manhattan* udaljenosti. Prema [20], *Manhattan* udaljenost, označena sa $d$, za dvije točke čije su koordinate $(x_1, y_1)$, odnosno $(x_2, y_2)$ može se izračunati prema sljedećoj formuli:

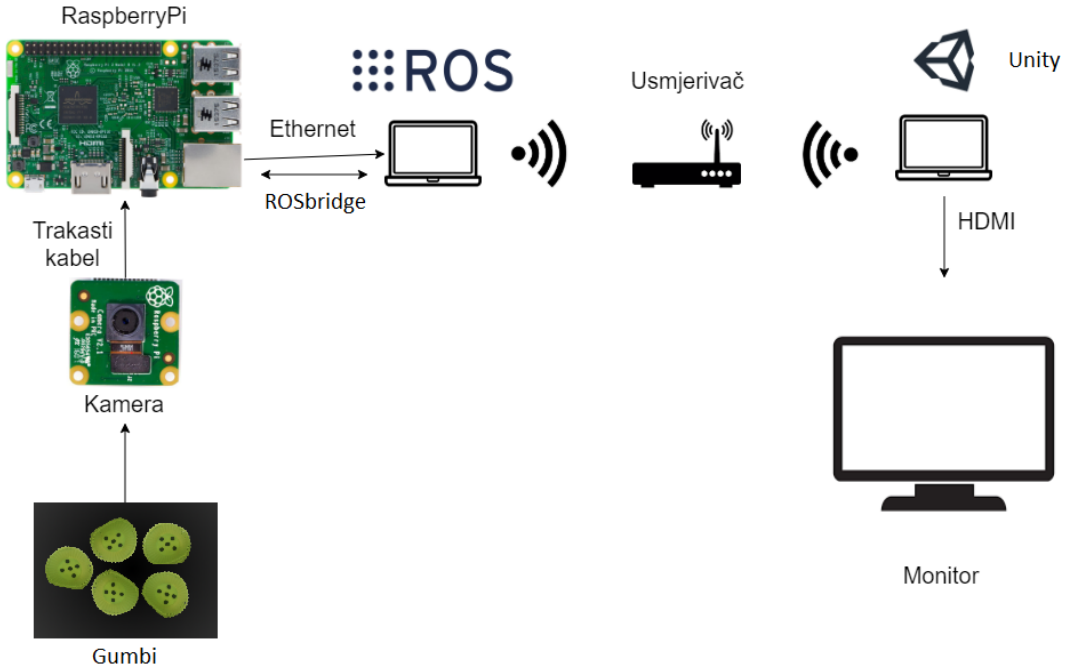
$$d = |x_1 - x_2| + |y_1 - y2| \tag{4.3}$$

Postupak se ponavlja za sve markere određenog gumba te se zbrajaju sve udaljenosti svakog markera trenutne u odnosu na inicijalnu sliku, nakon čega se izvršava sljedeći skup provjera. Ukoliko je zbroj udaljenosti svih markera do njihovih najbližih susjeda inicijalne slike veći od 50, tada se gumbu za koji se vrši obrada pridružuje vrijednost 1, odnosno pritisnuta, a u suprotnom se pridružuje vrijednost 0, odnosno nije pritisnut. Postupak se ponavlja za svaki gumb te se na taj način dobiva 5 brojeva s vrijednostima 0 ili 1 koji se putem *ROS* objavljivača prosljeđuju igri koja zatim

obavlja željenu akciju. Kao što je rečeno, skup algoritama obrade slike i detekcije pritiska svakog gumba izvršava se u realnom vremenu po primitku nove slike s ciljem maksimalne responzivnosti i smanjenja kašnjenja tijekom igranja igre. Uslijed stalnih pritisaka gumba, postepeno dolazi do sve većeg pomaka pozicija markera u odnosu na inicijalnu sliku koji je uzrokovan nedovoljno kvalitetnim materijalom korištenim za izradu gumba (želatina), što rezultira detekcijom pritiska gumba koji u tom trenutku možda nije pritisnut, jer je suma pomaka prešla vrijednost detekcije. U tom je slučaju potrebno pustiti sve gumbe i ponovno postaviti inicijalnu sliku i inicijalne pozicije markera promjenom iznosa zastavice *ready* u 0 nakon čega je moguće normalno nastaviti igranje igre.

## 4.3. Arhitektura sustava

U nastavku će biti iznesen sažet izgled ukupne arhitekture sustava, prikazane na slici 4.13.




**Slika 4.13:** Arhitektura sustava

Kada igrač pritisne silikonski gumb kamera koja snima u 20 sličica u sekundi kre-

ira novu sliku koja se šalje preko trakastog kabela u *Raspberry Pi* koji preko *Ethernet* kabela šalje sliku u *ROS*. *ROS* sadrži čvorove koji čitaju sliku, obrade je tako da izdvoje gumbe i pozicije markera na njima i usporedbe pozicije markera sa početnim pozicijama markera u stacionarnom stanju. Ako su markeri na gumbima pomaknuti za određeni prag, smatra se da je gumb pritisnut. *ROS* preko *Rosbridgea*, prema dokumentaciji dostupnoj na [21], kreira *WebSocket* server koji omogućuje da se druga računala spoje na *ROS* i pretplate na željene poruke. Računalo na kojem se izvodi igra se preko bežične mreže spoja na *WebSocket* server i pretplaćuje na poruku o stanju gumba. Čim se obradi slika i izračunaju nova stanja gumba šalje se poruka u videoigru. Kada videoigra primi poruku pretvori je u kontrole i primijeni silu na dron i pokupi ili dostavi poklon u slučaju da je taj gumb pritisnut. U tom trenutku se akcija igrača reflektira se na vanjskom monitoru. Cijela povratna veza od pritiska gumba do povratne informacije na ekranu traje oko 50 *ms*.

# 5. *Robofun* radionica

Radionica je održana 19. prosinca 2019. godine u auli FER-a. Na slici 5.1 su voditelji radionice na mjestu održavanja iste. Radionica je prvenstveno bila namijenjena učenicima osmog razreda Osnovne škole Tina Ujevića i učenicima prvog razreda XV. Gimnazije u Zagrebu u sklopu nastave iz informatike. Za njih su termini održavanja radionice bili unaprijed rezervirani, ali svi pripremljeni sadržaji demonstrirani su i ostalim zainteresiranim posjetiteljima. Rezervirani termini bili su u trajanju od jednog školskog sata, a unutar jednog termina radionicu je pohađao jedan razred. Po dolasku, učenici su podijeljeni u dvije skupine. Jedna skupina se teorijski i praktično upoznavala s upravljanjem robotskim manipulatorom, dok je druga skupina izrađivala komponente za upravljanje videoigrom, a zatim i u praksi provjeravala njihovu funkcionalnost. Nakon 20-ak minuta, grupe su zamijenile mjesta.

**Slika 5.1:** Voditelji *Robofun* radionice

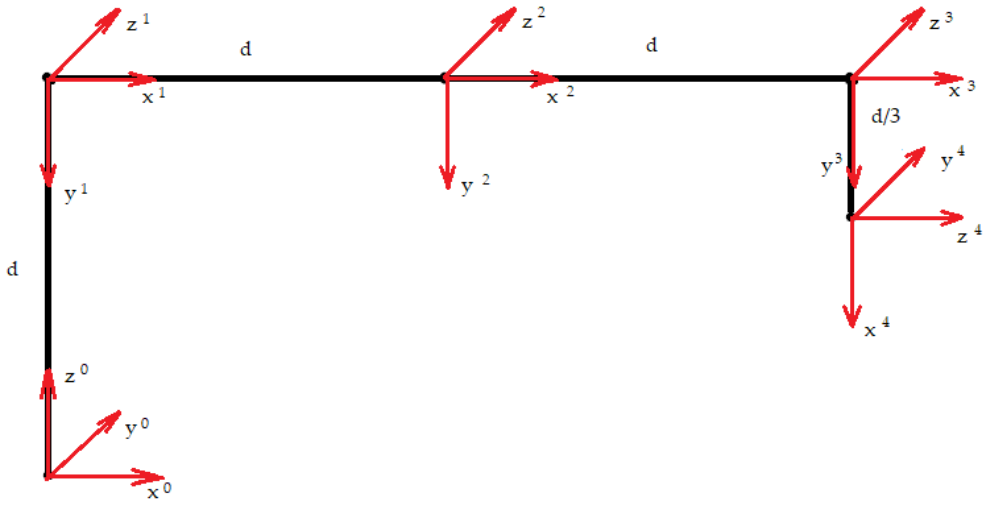
## 5.1. Upravljanje robotom

Učenicima je na samom početku predstavljen robotski manipulator, čija je izrada opisana u drugom poglavlju. Od njih je zatraženo da nabroje komponente robotskog manipulatora, odnosno materijale od kojih je izrađen. Učenici su s lakoćom zaključili da je plastika glavni građevni materijal predstavljenog manipulatora, pa je iskorištena prilika da im se objasni proces 3D printanja, što je olakšano činjenicom da se u blizini nalazio 3D printer koji je za to vrijeme izrađivao božićne ukrase. Pri nabrajanju komponenata, učenici su spomenuli motore, žice te ostale plastične dijelove. U tom trenutku je uveden naziv članak te je navedena analogija s ljudskom rukom, prema kojoj članke manipulatora nazivamo podlakticom, nadlakticom i sl. To je navelo učenike na zaključak da članke spajaju zglobovi (rame, lakat itd.) te su uspješno uočili da manipulator ima šaku, koja mu pri izvršavanju zadaća služi kao alat. Demonstrirana su i postojeća ograničenja robotskog manipulatora te je istaknuto da, kao što ni svi ljudski zglobovi nemaju puni opseg pokreta (360°), ni zglobovi robotskog manipulatora najčešće nemaju puni opseg. Takvo ograničenje se naziva unutarnjim, a ograničenje do kojeg dolazi uslijed prepreka prisutnih u okolini (učenici su prepoznali da bi to u našem slučaju mogalo biti osobno računalo na koji je manipulator spojen ili netko od sudionika u okolini manipulatora) naziva se vanjskim ograničenjem.

Nakon kratkog teorijskog uvoda, učenicima su podijeljeni jednostavni modeli manipulatora izrađeni od slamčica, prikazani na slici 5.3, na kojima je provjeravano njihovo shvaćanje objašnjenih pojmova te dodatno utvrđivano stečeno znanje. Sudionici su ispravno prepoznali da se korišteni model sastoji od tri zgloba i četiri članka. U nastavku će biti razrađena kinematika spomenutog slamka-robota, s ciljem omogućavanja usporedbe robota izrađenog od slamčica i stvarnog robotskog manipulatora, nakon čega će biti opisan praktični dio radionice.

### 5.1.1. Kinematika slamka-robota

U drugom poglavlju je ukratko objašnjena Denavit-Hartenbergova metoda te je korištenjem navedene metode dano rješenje direktne kinematike za stvarnog robota. Ista metoda korištena je i prilikom proračuna direktne kinematike za slamka-robota. Na slici 5.2 prikazana je shema slamka-robota sa označenim pripadajućim koordinatnim sustavima, a DH parametri su navedeni u tablici 5.1. Pri tome treba imati na umu da je baza slamka-robota nepomična, odnosno pričvršćena za ploču, a varijable $q_2$, $q_3$ i $q_4$ predstavljaju zakrete preostalih zglobova u trenutku $t$. Duljina slamke je na slici i u tablici označena varijablom $d$. Promatranjem sheme slamka-robota može se zaključiti da slamka-robot predstavlja planarni RRR manipulator.




**Slika 5.2:** Shema slamka-robota

Uvrštavanjem navedenih DH parametara [22] u matrice homogene transformacije i uzastopnim množenjem spomenutih matrica na način opisan u drugom poglavlju dobivena je matrica složene homogene transformacije prikazana jednadžbom 5.1.

| - | $\theta$ | $d$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| 1.os | 0 | $d$ | 0 | $\frac{-\pi}{2}$ |
| 2.os | $q_2[0]$ | 0 | $d$ | 0 |
| 3.os | $q_3[0]$ | 0 | $d$ | 0 |
| 4.os | $q_4[\frac{\pi}{2}]$ | 0 | $\frac{d}{3}$ | $\frac{\pi}{2}$ |

**Tablica 5.1:** Kinematički parametri slamka-robota

$$T_0^4 = \begin{bmatrix} -S_{234} & 0 & C_{234} & d \cdot (C_{23} + C_2 - \frac{1}{3} \cdot S_{234}) \\ 0 & 1 & 0 & 0 \\ -C_{234} & 0 & -S_{234} & d \cdot (1 - (S_{23} + S_2 + \frac{1}{3} \cdot C_{234})) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

Kao i za stvarnog robota, i za slamka-robota je iz navedene matrice moguće izvući varijable koje se koriste pri proračunu inverzne kinematike. One su navedene u nastavku.

$$w_1 = d \cdot (C_{23} + C_2 - \frac{1}{3} \cdot S_{234}) \tag{5.2}$$

$$w_2 = 0 \tag{5.3}$$

$$w_3 = d \cdot (1 - (S_{23} + S_2 + \frac{1}{3} \cdot C_{234})) \tag{5.4}$$

$$w_4 = C_{234} \tag{5.5}$$

$$w_5 = 0 \tag{5.6}$$

$$w_6 = -S_{234} \tag{5.7}$$

Iako slamka-robot ima samo tri stupnja slobode i nije kinematički redundantan, odnosno broj stupnjeva slobode je manji od dimenzije prostora, ipak postoji više mogućih rješenja inverznog kinematičkog problema. Takva redundantnost se prema [7] naziva funkcionalna redundantnost. U sklopu praktičnog zadatka demonstrirano je korištenje redundantnosti u svrhu izbjegavanja prepreka, a komentiran je i kriterij izbora optimalnog rješenja u slučaju postojanja višestrukih mogućnosti pozicioniranja robota.

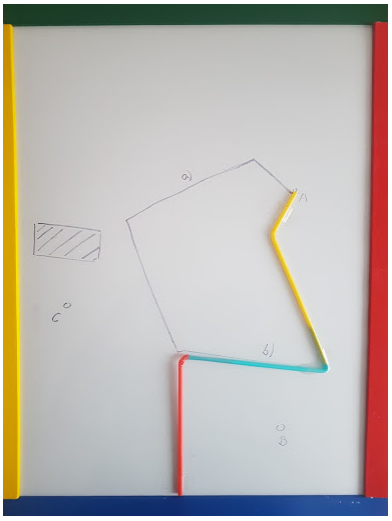
### 5.1.2. Praktični zadatak

U praktičnom dijelu radionice koji je uključivao rad sa slamka-robotom učenici su trebali zamisliti da se manipulator nalazi u tvornici automobilskih dijelova. Manipulator je pričvršćen za podlogu i treba doći u točku A, gdje preuzima objekt, koji zatim nosi u točku B na brušenje te konačno u točku C na lakiranje. Model od slamčica je radi demonstracije zadatka bio pričvršćen na ploču, na kojoj su označene spomenute točke, kao što je prikazano na slici 5.3a.

Prvi dobrovoljac imao je zadatak postaviti zglobove modela tako da vrh alata manipulatore postavi u točku A, kao što je prikazano na slici 5.3b. Zatim je drugi dobrovoljac pokušao pronaći drugi položaj za ostvarivanje istog cilja, što na primjer može biti položaj sa slike 5.3c. Oba rješenja označena su na ploči te su sudionici glasovanjem birali položaj koji je prema njihovom mišljenju bolji. Iako nisu znali dati razloge svog izbora, intuitivno su napravili točan odabir - položaj na ploči označen kao a). Učenicima je pojašnjeno da pri izboru treba težiti za tim da ukupan pomak svih zglobova bude čim je moguće manji. Zatim je uvedena pretpostavka da se motor koji pokreće prvi zglob pokvario i da je prvi zglob, označen na slici 5.3d, zaglavio u određenom položaju. Zadatak dobrovoljca je bio u takvoj situaciji dovesti vrh manipulatora u točku B. Nemogućnost ostvarivanja tog zadatka su učenici prepoznali kao praktičan primjer unutarnjih ograničenja. Vanjska ograničenja simulirana su preprekom, koja je na ploči prikazana kao pravokutnik crne boje, a koju je trebalo zaobići kako bi vrh manipulator dosegao točku C. Slika 5.3e prikazuje pogrešno rješenje, a slika 5.3f ispravno. Učenici su aktivno sudjelovali u rješavanju zadatka i spremno se javljali za demonstriranje rje-

**(a)** Početni položaj modela

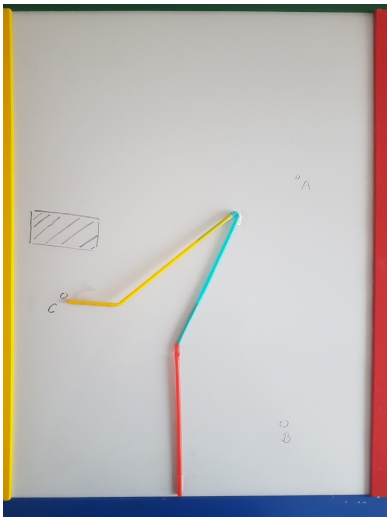
**(b)** Model u točki A



**(c)** Druga mogućnost dolaska u točku A

**(d)** Pokušaj dolaska u točku B

**(e)** Neispravan dolazak u točku C

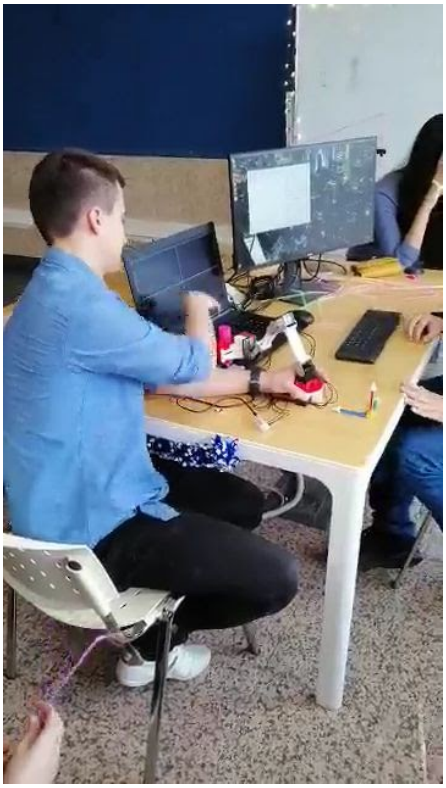


**(f)** Ispravan dolazak u točku C

**Slika 5.3:** Zadatak s modelom robotskog manipulatora

šenja na ploči. Također su ugodno iznenadili voditelje svojim logičkim zaključivanjem i zanimljivim pitanjima.

Za kraj, svaki od sudionika je dobio priliku upravljati stvarnim robotskim manipulatorom. Zadatak je bio koristiti manipulator za prijenos objekta od točke A do točke B, a bilo ga je moguće izvršiti na tri načina – upravljanjem vrhom alata manipulatora u kartezijskom prostoru, upravljanjem svakim pojedinim zglobom u prostoru zglobova i direktnim unošenjem varijabli svakog zgloba. Mogućnosti upravljanja manipulatorom su detaljnije opisane u drugom poglavlju. Učenicima je pri rukovanju s manipulatorom pomagao student, a sama izvedba je prikazana na slici 5.4.



**Slika 5.4:** Upravljanje robotskim manipulatorom na radionici

Kako bi učenici testirali tek objašnjene pojmove vezane za upravljanje robotskim manipulatorom, definirana su tri već spomenuta kratka zadatka. Kako je svaki idući zadatak bio nešto kompleksniji od prethodnog, polako su se objašnjavali postupci kroz koje algoritam prolazi prilikom izvršavanja osnovnih operacija, kao i razlozi postojanja fizičkih ograničenja u smislu limitiranog radnog područja i nemogućnosti preklapanja

članaka. Iako su svi zadaci bili temeljeni na jednostavnom pomaku predmeta iz jedne točke u prostoru u drugu, kompleksnost zadataka se ostvarila s načinom upravljanja robotskog manipulatora. U prvom zadatku se robotskim manipulatorom upravljalo u kartezijskom koordinatnom sustavu, odnosno u XYZ koordinatnom sustavu. Za vrijeme ovog zadatka učenici su pomoću grafičkog sučelja unosili vrijednosti za poziciju vrha alata u odnosu na X, Y i Z osi, dok je algoritam izračunavao zakret svakog motora u cilju postizanja željenih vrijednosti. U drugom zadatku učenici su morali ponoviti istu putanju robotskog manipulatora, ali se pri tome upravljalo svakim zglobom zasebno. U trećem zadatku učenici su ponovno rješavali isti problem, ali su pri tome vrijednosti zakreta zglobova morali direktno unositi u sustav. Upravo ovim jednostavnim zadacima učenicima je objašnjeno kroz koje sve postupke algoritam robotskog manipulatora mora proći kako bi se obavile ljudima jednostavne operacije.

## 5.2. *Soft* robotika

Po dolasku svake grupe, djeca su se najprije na interaktivan način upoznala s pojedinim komponentama *soft* dijela radionice. Prezentacija komponenti sustava prikazana je na slici 5.5. Susreli su se s upravljačkom konzolom u obliku drvene božićne kutije, uz koju su se nalazila dva monitora. Na jednom monitoru se odvijala videoigra, a na drugom su se u stvarnom vremenu mogli pratiti pomaci markera unutar gumba. S voditeljima su razgovarali o različitim načinima upravljanja videoigrama te o tome kako se konvencionalni načini upravljanja (igraća palica, tipkovnica, miš) razlikuje od predstavljenog sustava unutar drvene kutije. Objašnjen im je postupak dobivanja želatine i proces detekcije slike s kamere, a sve na njima prihvatljivoj razini razumijevanja.

**Slika 5.5:** Prezentacija načina rada videoigre djeci

Gornja ploha kutije, napravljena od pleksiglasa, mogla se podići, kao na slici 5.6, čime je bio omogućen uvid u unutrašnjost kutije u bilo kojem trenutku. Podizanjem poklopca kutije i uvidom djece da unutar kutije ne postoji ožičenje koje povezuju

gumbe sa računalom, uspješno je postignut efekt iznenađenja.

**Slika 5.6:** Podizanje poklopca kutije

Nakon što su voditelji završili izlaganje, djeca su mogla samostalno upravljati videoigrom sa silikonskim gumbima. Iako nisu naviknula na sporiji odziv prilikom stiskanja upravljačkih kontrola (gumba), bilo im je jako zanimljivo što su kontrole mekane, što mogu na ekranu pratiti pomak markera uzrokovan njihovim dodirom i što su gumbi bežično povezani sa ostatkom sustava.

Osim upravljanja sa gumbima od želatine, za one entuzijastične igrače bilo je omogućeno i igranje na velikom televizoru s igraćom palicom, uz mogućnost unosa imena igrača na kraju igre i prikaz tablice najboljih rezultata, što se vidi na slici 5.7.

Za mlađu djecu koja su dolazila neorganizirano na radionicu, omogućen je i radni stol sa indukcijskom pločom na kojoj su djeca uz nadzor voditelja sama mogla otopiti

unaprijed pripremljenu tvrdu želatinu kako bi dodatno stekla dojam o procesu stvaranja korištenih gumba.

**Slika 5.7:** Igra sa igraćom palicom

# 6. Zaključak

U radu je predstavljen razvoj i provedba *Robofun* radionice za djecu osnovnoškolskog i srednjoškolskog uzrasta. Preduvjeti za uspješnu provedbu radionice uključivali su razvoj videoigre, dizajn njenih komponenata, spajanje videoigre sa hardverskim dijelom sustava te konstruiranje robotskog manipulatora i implementaciju njegove upravljačke logike. Cilj cijelog procesa bio je predstaviti djeci robotiku na njima blizak i interaktivan način i pokazati im koliko znanost u praksi može biti zabavna.

Prema povratnoj informaciji dobivenoj od učenika, od kojih su neki iskazali želju da u skoroj budućnosti postanu studenti FER-a, cilj radionice je postignut. Učenici su prepoznali složenost problema upravljanja robotskim manipulatorom, ali su u tom izazovu uočili i brojne mogućnosti. Otkriveno im je jedno novo, njima nepoznato područje robotike - *soft* robotika, o čijem razvoju su razgovarali s voditeljima radionice. Prezentirano im je na koji način se *soft* komponente mogu koristiti i u industriji videoigara, a iznenadila ih je informacija da je takve komponente moguće izraditi od jestivih materijala. Imali su priliku zaigrati videoigru korištenjem inovativnog upravljačkog sustava, razvijenog na FER-u, ali i korištenjem klasične upravljačke palice, što je radionicu učinilo dodatno opuštenom i zabavnom.

Potaknuti uspjehom provedene radionice, autori se nadaju da će ista, u sličnom ili novom formatu, biti ponovljena kada jedini prisutni virus bude adventska groznica koja će još jednom obuzeti robote s FER-a.

# LITERATURA


[1] Sangbae Kim, Cecilia Laschi, and Barry Trimmer. Soft robotics: a bioinspired evolution in robotics. *Trends in Biotechnology*, 31(5):287 – 294, 2013.

[2] C. Chorley, C. Melhuish, T. Pipe, and J. Rossiter. Development of a tactile sensor based on biologically inspired edge encoding. *Advanced Robotics, ICAR*, 2009.

[3] J. Shintake, H. Sonar, E. Piskarev, J. Paik, and D. Floreano. Soft pneumatic gelatin actuator for edible robotics. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[4] *ROBOTIS STORE*. http://www.robotis.us/.

[5] *ROBOTIS e-Manual*. https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/. Pristupano 7.8.2020.

[6] *ROBOTIS Dynamixel XM430-W350*. https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/. Pristupano 21.8.2020.

[7] Zdenko Kovačić, Stjepan Bogdan, and Vesna Krajči. *Osnove Robotike*. Graphis, 2002.

[8] Donald L. Pieper. *The kinematics of manipulators under computer control*. PhD thesis, 1968.

[9] David E. Orin and William W. Schrader. Efficient computation of the jacobian for robot manipulators. *The International Journal of Robotics Research*, 1984.

[10] Kris Hauser. *Robotic Systems (draft)*. http://motion.cs.illinois.edu/RoboticSystems/. Pristupano 7.8.2020.

[11] Dave Peterson. *Trigonometric Equations: An Overview*. https://www.themathdoctors.org/trigonometric-equations-an-overview/. Pristupano 7.8.2020.

[12] Mark Griffiths. The educational benefits of videogames. *Education and Health*, 20:47–51, 01 2002.

[13] *Autodesk Fusion for educational purpose*. https://www.autodesk.com/products/fusion-360/overview.

[14] *Christmas Low-poly Scene*. https://sketchfab.com/3d-models/christmas-low-poly-scene-7984bd6024db4b89928ddeb625eebac0. Pristupano 4.6.2020.

[15] *Moon 2.0*. https://www.fontmirror.com/moon-2-0. Pristupano 4.6.2020.

[16] *ROS#*. https://github.com/siemens/ros-sharp. Pristupano 4.6.2020.

[17] *Edible actuators*. https://softroboticstoolkit.com/edible-actuators. Pristupano 4.6.2020.

[18] Khvedchenia Ievgen Daniel Lélis Baggio, Shervin Emami. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing Ltd., 2012.

[19] Paulo de Carvalho Lourena Karin de Medeiros Rocha, Luiz Velho. Image moments-based structuring and tracking of objects. *15th Brazilian Symposium on Computer Graphics and Image Processing*, 2002.

[20] Robert Nisbet, John Elder, and Gary Miner. *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press, 2009.

[21] *ROSbridge*. http://wiki.ros.org/rosbridge_suite. Pristupano 7.8.2020.

[22] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

## SAŽETAK

**Filip Bradarić, Marko Cavalli, Ivona Krajačić, Matija Kunštek, Andrea Rebec, Jelena Tabak**

**Radionica za djecu "Robofun"**

U sklopu humanitarne akcije “Adventska groznica obuzela robote na FER-u” za “Društvo za poboljšanje kvalitete života siromašne i nezbrinute djece *Mali zmaj*", na Fakultetu elektrotehnike i računarstva u prosincu 2019. godine organizirana je radionica za djecu osnovnoškolskog i srednjoškolskog uzrasta nazvana *Robofun*. Interaktivnu radionicu pripremili su studenti pete godine FER-a u suradnji sa studenticom Arhitektonskog fakulteta. U sklopu radionice razvijena je videoigra te su dizajnirane njene komponente koje su spojene je sa hardverskim dijelom sustava koji uključuje upravljačke gumbe izrađene od želatine. Konstruiran je i robotski manipulator te implementirana njegova upravljačka logika.

Radionica se sastojala od dva dijela. Prvi dio odnosio se na usvajanje teorije i praktičnog rada pri upravljanju jednostavnim robotskim manipulatorom. Konstruirana robotska ruka s 3D printanim dijelovima i *Servo* motorima omogućila je podizanje i premještanje laganih predmeta.

Drugi dio je bio posvećen videoigri *Santa’s Delivery Drone*. U središtu pozornosti osim same kreirane igre, bile bila je igraća konzola u obliku božićnog poklona s upravljačkim gumbićima od želatine. Osim humanitarnog, cilj radionice je bio promocija STEM područja među djecom na zabavan i edukativan način.



**Ključne riječi:** robotski manipulator,meka robotika, videoigra

## ABSTRACT

**Filip Bradarić, Marko Cavalli, Ivona Krajačić, Matija Kunštek, Andrea Rebec, Jelena Tabak**

**Workshop for children: "Robofun"**

As part of the humanitarian aid "Advent fever took over the robots at FER" for the "Society for improving the quality of life of poor and neglected children *Mali zmaj*", at the Faculty of Electrical Engineering and Computing in December 2019, there was organised a workshop for primary and secondary school children called *Robofun*. The set of interactive workshops was prepared by the graduate students of FER in cooperation with a student of the Faculty of Architecture. As a preparation for the workshop, a novel video game was developed, components of aforementioned game were designed and the game was connected with the hardware part of the system, including controllers made of gelatin. In addition, robotic manipulator was constructed and its control system has been implemented.

The workshop consisted of two parts. The first part referred to the overcoming of theory and practical work in controlling a simple robotic manipulator. A constructed robotic arm with 3D printed parts and *Servo* motors made it possible to lift and move light objects. The second part was dedicated to the video game *Santa's Delivery Drone*. In the focus of the workshop, in addition to the game itself, was a game console in the form of a Christmas gift with gelatin control buttons.

Besides to the humanitarian, the goal of the workshop was to promote the STEM area among children in a fun and educational way.



**Keywords:** robotic manipulator, soft robotics, video game