

# Uredaj za ispučavanje loptica za stolni tenis

---

Ričko, Andrija

**Undergraduate thesis / Završni rad**

**2019**

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture / Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje**

Permanent link / Trajna poveznica: <https://urn.nsk.hr/urn:nbn:hr:235:679903>

Rights / Prava: [In copyright/Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2024-05-14**

Repository / Repozitorij:

[Repository of Faculty of Mechanical Engineering and Naval Architecture University of Zagreb](#)



SVEUČILIŠTE U ZAGREBU  
FAKULTET STROJARSTVA I BRODOGRADNJE

**ZAVRŠNI RAD**

**Andrija Ričko**

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU  
FAKULTET STROJARSTVA I BRODOGRADNJE

**ZAVRŠNI RAD**

Mentor:

Prof. dr. sc. Mario Štorga, dipl. ing

Student:

Andrija Ričko

0035200783

Zagreb, 2019.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu.

Zahvaljujem svom mentoru prof. dr. sc. Mariu Štorgi koji mi je svojom pomoći i savjetima pomogao tijekom izrade ovog završnog rada.

Također, zahvaljujem se svojoj djevojci, obitelji, kolegama i prijateljima koji su mi bili podrška tijekom studiranja.

Andrija Ričko



SVEUČILIŠTE U ZAGREBU  
FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite



Povjerenstvo za završne ispite studija strojarstva za smjerove:  
procesno-energetski, konstrukcijski, brodostrojarski i inženjersko modeliranje i računalne simulacije

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

## ZAVRŠNI ZADATAK

Student: **Andrija Ričko** Mat. br.: 0035200783

Naslov rada na  
hrvatskom jeziku:

**Uredaj za ispučavanje loptica za stolni tenis**

Naslov rada na  
engleskom jeziku:

**Table Tennis Ball Serving Machine**

Opis zadatka:

Razvoj sportske industrije rezultirao je potrebom za razvojem uređaja za automatizaciju sportskih treninga. Sukladno tome, kod ljudi koji treniraju stolni tenis pojavila se ideja o uređaju koji bi samostalno ispučavao teniske loptice te tako omogućio individualni trening sportaša. Uredaj bi trebao omogućiti raznolikost treninga, rotaciju loptice, snagu ispučavanja, promjenjivi domet i kut ispučavanja, te tako simulirati serviranje loptice na različite pozicije i uz različite uvijete ispučavanja loptice.

U radu je potrebno:

- Izraditi tehničku specifikaciju za razvoj uređaja.
- Metodičkom razradom obuhvatiti različita konceptualna rješenja.
- Tehno-ekonomskom analizom odabrati projektno rješenje.
- Odabranje rješenje razraditi uz uporabu standardnih sklopova, te s potrebnim proračunima nestandardnih dijelova. Pri konstrukcijskoj razradi paziti na tehnološko oblikovanje komponenti te sigurnost korisnika pri korištenju uređaja.
- Izraditi računalni 3D model uređaja i tehničku dokumentaciju.

Opseg konstrukcijske razrade, modeliranja i izrade tehničke dokumentacije dogovoriti tijekom izrade rada.

U radu navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

29. studenog 2018.

Rok predaje rada:

1. rok: 22. veljače 2019
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2018.

Predviđeni datumi obrane:

1. rok: 25.2. – 1.3.2019.
2. rok (izvanredni): 2.7.2019.
3. rok: 23.9. – 27.9.2019.

Zadatak zadao:

Prof. dr. sc. Mario Štorga

Predsjednik Povjerenstva:

Prof. dr. sc. Igor Balen

# SADRŽAJ

Sadržaj .....	1
Popis slika .....	3
Popis tablica .....	4
Popis tehničke dokumentacije i programskog koda .....	5
Popis oznaka .....	6
Sažetak .....	7
1. Uvod .....	8
1.1. Što je to robot za stolni tenis?[1] .....	8
1.2. Načelo rada uređaja .....	8
1.3. Prostor potreban za uređaj .....	8
1.4. Zašto kupiti uređaj? .....	9
2. Analiza tržišta .....	10
2.1. Tvrta iPong .....	10
2.1.1. iPong Topspin .....	10
2.1.2. iPong Original .....	11
2.1.3. iPong Pro .....	12
2.2. Tvrta Newgy Industries .....	13
2.2.1. Model 1040+ .....	13
2.2.2. Model 2055 .....	14
2.2.3. Model 3050XL .....	15
2.3. Tvrta Tamasu Butterfly .....	16
2.3.1. Uređaji tvrtke Tamasu Butterfly .....	16
2.4. Usporedba postojećih proizvoda .....	19
3. Pregled patenata .....	20
3.1. Patent US4844458A <i>Table tennis ball serving device</i> [5] .....	20

3.2.	Patent US20070221187A1 <i>Table tennis ball service machine</i> [6] .....	21
3.3.	Patent US3878827A <i>Table tennis ball serving apparatus</i> [7] .....	22
4.	Tehnička specifikacija .....	23
5.	Funkcijska dekompozicija.....	24
6.	Morfološka matrica parcijalnih rješenja .....	25
7.	Koncepti .....	29
7.1.	Koncept 1 .....	29
7.2.	Koncept 2.....	30
7.3.	Vrednovanje koncepata .....	30
8.	Konstrukcijska razrada.....	32
8.1.	Dimenzije stola i brzina loptice.....	32
8.2.	Odabir motora i tarenica za ispučavanje loptica i servo motora za nagib i zakret uređaja	
	32	
8.2.1.	Motori i tarenice za ispučavanje loptica.....	32
8.2.2.	Servo motor za nagib i zakret uređaja .....	35
8.3.	Oblikovanje rješenja.....	36
8.4.	Izvod jednadžbi za ispučavanje loptice .....	39
8.5.	Izvod jednadžbi za rotaciju loptice.....	41
8.6.	Razvoj korisničko sučelje aplikacije za upravljanje uređajem .....	44
9.	Zaključak.....	47
Literatura.....		48
Prilozi.....		49

## **POPIS SLIKA**

Slika 2.1. Logo tvrtke iPong [2].....	10
Slika 2.2. iPong Topspin s upravljačem [2].....	11
Slika 2.3. iPong Original s upravljačem [2].....	11
Slika 2.4. iPong Pro s upravljačem [2] .....	12
Slika 2.5. Logo tvrtke Newgy Industries [3].....	13
Slika 2.6. Model 1040+[3].....	14
Slika 2.7. Model 2055 s upravljačkim uređajem [3] .....	14
Slika 2.8. Model 3050XL i Newgy aplikacija [3] .....	15
Slika 2.9. Logo tvrtke Tamasu Butterfly [4].....	16
Slika 2.10. Model Amicus Start s kontrolnom pločom[4].....	17
Slika 2.11. Model Amicus Expert s kontrolnom pločom[4].....	17
Slika 2.12. Model Amicus Prime s aplikacijom[4] .....	18
Slika 3.1. Patent US4844458A[5] .....	20
Slika 3.2. Patent US20070221187A1[6] .....	21
Slika 3.3. Patent US3878827A[7] .....	22
Slika 7.1. Koncept 1 .....	29
Slika 7.2. Koncept 2 .....	30
Slika 8.1. Dimenzije stola za stolni tenis .....	32
Slika 8.2. Tarenica za ispučavanje loptica .....	32
Slika 8.3. Motor za ispučavanje loptice.....	33
Slika 8.4. Kontroler za motore .....	34
Slika 8.5. Servo motor .....	35
Slika 8.6. Uredaj za ispučavanje loptica za stolni tenis - Pogled 1 .....	36
Slika 8.7. Uredaj za ispučavanje loptica za stolni tenis - Pogled 2 .....	37
Slika 8.8. Uredaj za ispučavanje loptica za stolni tenis - Pogled 3 .....	38
Slika 8.9. Putanja loptice: gornja slika-nacrt stola; donja slika-tlocrt stola.....	40
Slika 8.10. Rotacija loptice .....	42
Slika 8.11. Sučelje aplikacije za upravljanje uređajem .....	44
Slika 8.12. Sučelje aplikacije s označenim elementima .....	45

## **POPIS TABLICA**

Tablica 2.1. Usporedba postojećih proizvoda.....	19
Tablica 4.1. Tehnička specifikacija.....	23
Tablica 6.1. Morfološka matrica rješenja .....	25
Tablica 7.1. Vrednovanje koncepata .....	31
Tablica 8.1. Karakteristike motora za ispučavanje loptica .....	33
Tablica 8.2. Karakteristike kontrolera za motore.....	34
Tablica 8.3. Karakteristike servo motora.....	35

## **POPIS TEHNIČKE DOKUMENTACIJE I PROGRAMSKOG KODA**

AR-2019-000 Uređaj za ispučavanje loptica za stolni tenis

AR-2019-001 Kućište A

AR-2019-002 Kućište B

AR-2019-003 Središnja ploča kućišta

AR-2019-004 Nosač A

AR-2019-005 Nosač B

AR-2019-006 Postolje A

AR-2019-007 Postolje B

AR-2019-008 Gumena nožica

AR-2019-009 Nastavak za servo motor

AR-2019-010 Lopatice

AR-2019-011 Gumeni rub

MainActivity.java

BtSend.java

OnReceiveInterface.java

activity\_main.xml

AndroidManifest.xml

## POPIS OZNAKA

OZNAKA	MJERNA JEDINICA	OPIS
$a_{max}$	$m/s^2$	Maksimalno ubrzanje loptice
$B$	$m$	Širina stola
$d$	$m$	Udaljenost
$d_L$	$m$	Promjer loptice
$d_t$	$m$	Promjer tarenice
$F_{max}$	$N$	Sila na lopticu
$g$	$m/s^2$	Ubrzanje slobodnog pada
$h$	$m$	Visina uređaja
$h_0$	$m$	Visina mrežice
$L$	$m$	Dužina stola
$M_{max}$	$Nm$	Maksimalni moment na tarenicama
$m$	$g$	Masa loptice
$R$	—	Vektor
$s$	$m$	Udaljenost
$t_{min}$	$s$	Minimalno vrijeme ubrzanja
$v$	$m/s$	Brzina loptice
$v_1$	$m/s$	Obodna brzina tarenice 1
$v_2$	$m/s$	Obodna brzina tarenice 2
$v_3$	$m/s$	Obodna brzina tarenice 3
$v_{max}$	$m/s$	Maksimalna brzina loptice
$v_{Tmax}$	$m/s$	Maksimalna obodna brzina tarenice
$z_{rot}$	—	Koordinata
$z_{T1}$	—	Koordinata
$z_{T2}$	—	Koordinata
$z_{T3}$	—	Koordinata
$x_0$	—	Koordinata
$x_1$	—	Koordinata
$y_{rot}$	—	Koordinata
$y_0$	—	Koordinata
$y_1$	—	Koordinata
$y_{T1}$	—	Koordinata
$y_{T2}$	—	Koordinata
$y_{T3}$	—	Koordinata
$\alpha$	$rad$	Kut nagiba uređaja
$\varphi$	$rad$	Kut zakreta uređaja
$\Pi$	—	Ravnina

## **SAŽETAK**

U radu je prikazan razvoj uređaja koji služi za automatsko ispučavanje loptica za stolni tenis kao pomoć kod individualnog treninga. Analizom tržišta utvrđeno je da slični uređaji postoje no usporedbom rješenja različitih tvrtki zaključeno je da nisu pogodni za većinu sportaša/klubova zbog visoke cijene ili ograničenih funkcija. Na temelju funkcijске dekompozicije i morfološke matrice predložena su 2 koncepta rješenja. Vrednovanjem tih koncepta odabранo je jedno rješenje temeljem čega je napravljena konstrukcijska razrada uređaja. Napravljen je proračun kritičnih komponenti, te je na kraju izrađen 3D model i tehnička dokumentacija dijelova uređaja korištenjem programskog paketa Solid Edge 2019. Za uređaj je napravljen i fizički prototip koji se koristio u svrhu potvrde koncepta te prikupljanja podataka potrebnih za razradu i optimiranje.

Ključne riječi: ispučavanje, stolni tenis, robot, trening, automatizacija, ping-pong

## **1. UVOD**

Uređaj za ispučavanje loptica za stolni tenis, ili skraćeno robot za stolni tenis, komercijalno dostupni od kasnih 1980-ih.

Možda ste čuli za robote za stolni tenis, ali ne znate što su oni ili što mogu učiniti? U nastavku ovog rada je opisano što su oni i kako rade.

### **1.1. Što je to robot za stolni tenis?[1]**

Robot za stolni tenis je, najjednostavnije objašnjeno, uređaj koji na kontrolirani način može automatski ispučavati loptice za stolni tenis s jednog kraja stola na drugi kraj.

Roboti za stolni tenis obično se sastoje od pet glavnih elemenata:

1. Spremnika za loptice
2. Mehanizma za ispučavanje loptica
3. Mehanizma za određivanje pozicije ispučane loptice
4. Upravljačke jedinice pomoću koje se kontrolira način ispučavanja loptice
5. Mreže za hvatanje povratnih loptica

Neki od osnovnih modela imaju ograničen mehanizam za određivanje pozicije ispučane loptice te ne uključuju mrežu za hvatanje povratnih loptica. Također imaju ograničenu funkciju rotacije loptice, te raznolikost opcija za treniranje.

### **1.2. Načelo rada uređaja**

Danas postoji nekoliko različitih varijanti uređaja koji svi slijede ista načela korištenja.

Uređaj se pričvrsti ili postavi na jedan kraj stola za stolni tenis i usmjeri se prema suprotnom kraju stola. Zatim se spremnik za loptice napuni s lopticama. Kada je robot programiran (ako postoji ta opcija) i uključen, ispučavat će loptice iz mehanizma za ispučavanje na jedan ili na više različitih položaja na stolu.

### **1.3. Prostor potreban za uređaj**

Dobra vijest o robotima je da ne zauzimaju mnogo prostora, to jest zauzimaju mnogo manje prostora nego da trenirate s drugom osobom.

Budući da je robot obično pričvršćen na jedan kraj stola, ili samo postavljen na stol, taj kraj stola se može gurnuti uz zid.

Naravno da je i dalje potrebno imati dovoljno prostora na strani stola gdje se nalazi igrač, ali ukupni prostor je mnogo manji nego da igrate protiv druge osobe.

#### **1.4. Zašto kupiti uredaj?**

1. Nije potreban drugi igrač, a i puno je bolji od “povratne ploče“
2. Može se vježbati bez ograničenja
3. Omogućava brzo i učinkovito savladavanje poteza ili rutine
4. Lakše vježbanje određenih udaraca zbog ponovljivosti udaraca
5. Poboljšava rad nogu, mogu se programirati za ispučavanje loptica na određene lokacije(samo određeni modeli).
6. Može biti koristan za poboljšanje kondicije

## 2. ANALIZA TRŽIŠTA

Analizom tržišta obuhvatit će se uređaji za automatsko ispučavanje loptica za stolni tenis koji se trenutno nalaze na tržištu. Usporediti će se karakteristike i bitne značajke proizvoda.

Danas postoji samo nekoliko tvrtki koje se bave izradom takvih uređaja. U analizi su spomenute tri tvrtke koje imaju takve uređaje na tržištu. Odabrane su tvrtke od kojih jedna proizvodi uređaje za povremene igrače i neke manje klubove, tvrtka koja ima uređaje namijenjene za profesionalne igrače i klubove, te tvrtka koja se nalazi negdje između.

### 2.1. Tvrta iPong



Slika 2.1. Logo tvrtke iPong [2]

Tvrta iPong osnovan je 2010. godine. Sjedište tvrtke iPong nalazi se u Rockvilleu, u Marylandu, u SAD-u. Ima oko 40 zaposlenika i procijenjeni godišnji prihod od 1,3 milijuna dolara. [2]

Tvrta se bavi izradom relativno jeftinih i pristupačnih uređaja za ispučavanje loptica za stolni tenis.

#### 2.1.1. *iPong Topspin*

Model iPong Topspin ima vrlo jednostavnu konstrukciju. Sastoje se od tri dijela, donji dio kao postolje, srednji dio u kojemu se nalazi mehanizam za ispučavanje, te gornji dio koji služi za pohranu loptica. Ima kapacitet od 100 loptica. Dimenzije su mu 32 cm x 32 cm x 48cm i masa 1.14 kg. Može ispučavati između 12 i 70 loptica u minuti. Napaja se putem baterije. Kod ispučavanja loptice nije moguća kontrola putanja loptice te omogućena samo jedna vrsta rotacije loptice (*Top*). Za upravljanje uređajem koristi kontrolna ploča koja je povezana s uređajem pomoću žice. Cijena ovog uređaja je \$99.95.

Slika 2.1. prikazuje uređaj iPong Topspin, te žični upravljač.



Slika 2.2. iPong Topspin s upravljačem [2]

### 2.1.2. *iPong Original*

Model iPong Original ima jednaku konstrukciju kao i model iPong Topspin. Također ima kapacitet od prilike 100 loptica. Dimenzije su mu 32 cm x 32 cm x 48cm i masa 1.14 kg. Može ispučavati između 12 i 70 loptica u minuti. Napaja se iz mreže. Kod ispučavanja loptice nije moguća kontrola putanja loptice te su moguće samo dvije vrsta rotacija loptice (*Top i Back*). Za upravljanje uređajem koristi se žični upravljač. Cijena ovog uređaja je \$149.95.

Slika 2.3. prikazuje uređaj iPong Topspin, te žični upravljač.



Slika 2.3. iPong Original s upravljačem [2]

### **2.1.3. iPong Pro**

Model iPong Pro je u skoro potpunosti jednak modelu Original. Jedina je razlika u tome da je kod ovog modela moguća veća kontrola putanja loptice, automatsko pozicioniranje uređaja tako da loptica može udariti na više različitih položaja na stolu. Za upravljanje uređajem koristi se žični upravljač. Cijena ovog uređaja je \$199.95.

Slika 2.4. prikazuje uređaj iPong Topspin, te žični upravljač.



**Slika 2.4. iPong Pro s upravljačem [2]**

## 2.2. Tvrta Newgy Industries



Slika 2.5. Logo tvrtke Newgy Industries [3]

Newgy Industries, Inc. osnovao je Joseph E. Newgarden, Jr. (1929-2017), entuzijast za stolni tenis, koji je izvorno počeo raditi na izumu Robo-Pong (uredaj za automatsko ispučavanje loptica za stolni tenis) 1972. godine. Tri godine kasnije njegov je hobi postao posao. Robot za stolni tenis uzeo je 16 godina istraživanja i razvoja prije svog prvog komercijalnog izdanja 1988. [3]

Newgy se zalaže za proizvode za stolni tenis koji su pristupačni, kvalitetni i jednostavnii za korištenje.

### 2.2.1. Model 1040+

Model 1040+ najjednostavniji je model tvrtke Newgy. Sastoje od košare u kojoj je pohranjeno oko 200 loptica i mehanizma koji može dovoditi od 26 do 94 loptice po minuti prema glavi za ispučavanje loptica. Glava za ispučavanje loptica ima mogućnost automatiziranog zakretanja glava lijevo-desno te ručno zakretanje glave oko osi smjera ispučavanja loptice. Mehanizam za ispučavanje loptica sastoji se od elektromotora na kojem se nalazi tarenica koja u zahvatu s lopticom izbacuje lopticu iz uređaja određenom brzinom. Kod takve izvedbe moguće je dobiti rotaciju loptice samo u jednom smjeru pa je iz tog razloga dodana mogućnost zakretanja glave oko osi smjera ispučavanja loptice. Za upravljanje uređajem koristi se žični kontroler na kojemu je moguće podešiti brzinu ispučavanja loptice, frekventnost ispučavanja i smjer ispučavanja loptice. Dimenzije uređaja su 33x50x23cm, a masa uređaja je 6,5kg. Cijena uređaja je \$399.

Slika 2.6. prikazuje uređaj sa svim dodacima koji kod kupovine dolaze uz uređaj.



Slika 2.6. Model 1040+[3]

### 2.2.2. *Model 2055*

Model 2055 po konstrukciji mehanizma za dovođenje loptica i mehanizma za ispučavanje loptica jednak je modelu 1040+. Od modela 1040+ razlikuje se po tome što dolazi s mrežom za hvatanje loptica i naprednjim upravljačkim uređajem pomoću kojega je moguće programirati do 64 različitih vježbi. Dimenzije sklopljenog uređaja su 36x84x28cm, rasklopljenog 153x79x46cm, a masa uređaja je 11kg. Cijena uređaja je \$999.

Slika 2.7. prikazuje Model 2055 koji je montiran za stol s upravljačkim uređajem.

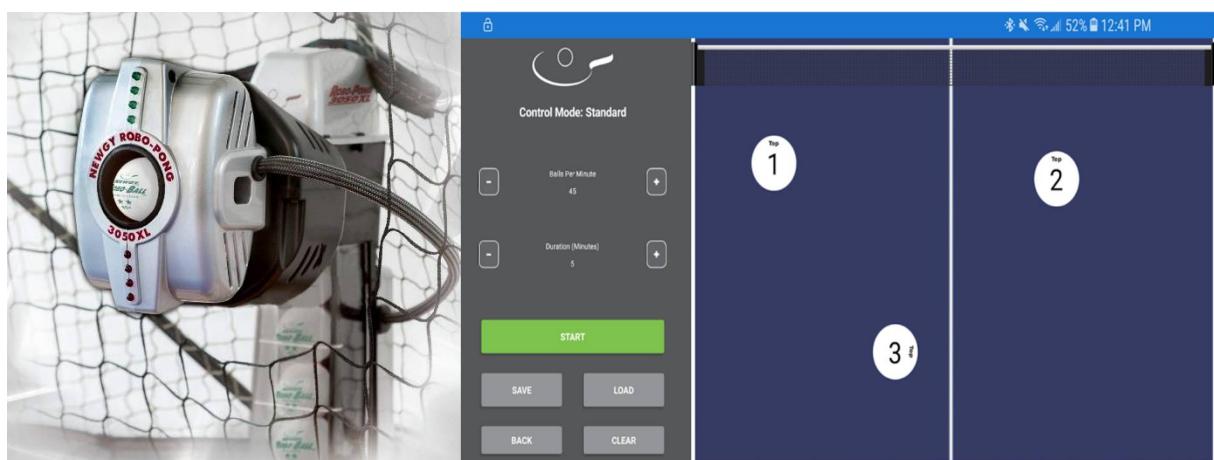


Slika 2.7. Model 2055 s upravljačkim uređajem [3]

### 2.2.3. Model 3050XL

Model 3050XL je najnapredniji model tvrtke Newgy. Po konstrukciji mehanizma za dovođenje loptice jednak je Modelu 1040+ i 2055, a mehanizam za ispučavanje loptica nešto je napredniji. Za razliku od prijašnjih modela, ovaj model ima 2 elektromotora u mehanizmu za ispučavanje loptica. Također rotacija glave, oko osi smjera ispučavanja loptice, je kod ovog uređaja automatizirana. Za upravljanje uređajem koristi se pametni telefon ili tablet pomoću kojeg se putem aplikacije korisnik povezuje preko *Bluetooth-a* sa uređajem. Dimenzije uređaja jednake su kao i kod modela 2055, a masa je nešto manja i iznosi 8,2kg. Cijena uređaja je \$1699.

Slika 2.8. prikazuje izgled glave za ispučavanje loptica kod modela 3050XL, te sučelje aplikacije za upravljanje uređajem.



Slika 2.8. Model 3050XL i Newgy aplikacija [3]

## 2.3. Tvrta Tamasu Butterfly



Slika 2.9. Logo tvrtke Tamasu Butterfly [4]

19. prosinca 1950. Hikosuke Tamasu osnovao je Tamasu Co., Ltd. u malom gradu Yanai City u Japanu. To je bio početak poslovne karijere koja je gotovo jedinstvena u stolnom tenisu. [4]

Osnivač tvrtke Tamasu je ispunio svoj životni san kada je prije pola stoljeća svoj hobi posvetio svojoj profesiji. U to vrijeme nije mogao ni zamisliti da bi Butterfly učinio vodećom robnom markom u svijetu. [4]

Tvrta se bavi proizvodnjom visoko kvalitetne opreme za stolni tenis koja je namijenjena profesionalnim klubovima za trening stolnog tenisa.

### 2.3.1. Uređaji tvrtke Tamasu Butterfly

Tvrta Tamasu Butterfly nudi 3 uređaja za ispučavanje loptica za stolni tenis. Amicus Start, Amicus Expert i Amicus Prime. Ta 3 uređaja su po konstrukciji jednaka te jedina razlika je u veličini košare za pohranu loptica te kako se upravlja uređajem.

Uređaj je konstruiran na način da se pričvrsti na jednu stranu stola. Košara za pohranu loptica nalazi se ispod stola odnosno ispod samog uređaja. Unutar te košare nalazi se mehanizam koji prikuplja loptice koje se nalaze u košari te ih podiže do mehanizma za ispučavanje. Mehanizam za ispučavanje sastoji se od 3 tarenice koje u zahvatu s lopticom ispučavaju lopticu prema metalnoj pločici koju je moguće zakretati za postizanje različitih kutova kojim loptica udara u tu pločicu i samim time određuje položaj na koji će loptica udariti u stol. Uređajem je moguće dobiti bilo kakvu rotaciju loptica što se postiže kontroliranjem brzine vrtnje svake tarenice zasebno. Svaki model dolazi s mrežom za hvatanje loptica koja se nalazi iza uređaja i usmjeruje vraćene loptice u košaru za pohranu loptica. Okvirne dimenzije su 158x86x160cm dok je masa uređaja s mrežom za hvatanje loptica 8,8kg.

### 2.3.1.1. Model Amicus Start

Model Amicus Start dolazi s kontrolnom pločom na kojoj je predefiniran određen broj vježbi i nije moguće programiranje vlastitih vježbi, već samo podešavanje pojedinih parametara postojećih vježbi. Predefinirane vježbe uključuju razne udarce kao što su serviranje i vraćanje loptica s različitim rotacijama i pozicijama ispucavanja. Svaka vježba može imati do 6 različitih načina ispucavanja loptica. Cijena uređaja iznosi \$1279.

Slika 2.10. prikazuje model Amicus Start sa kontrolnom pločom.



Slika 2.10. Model Amicus Start s kontrolnom pločom[4]

### 2.3.1.2. Model Amicus Expert

Model Amicus Expert dolazi s kontrolnom pločom s 99 predefiniranih vježbi koje je također moguće podešavati po vlastitoj želji i programiranje vlastitih vježbi. Moguće je podešavanje rotacije loptice, položaja i frekventnosti ispucavanja loptica. Svaka vježba može sadržavati do 7 različitih načina ispucavanja. Cijena uređaja iznosi \$1829.

Slika 2.11. prikazuje model Amicus Expert sa kontrolnom pločom.

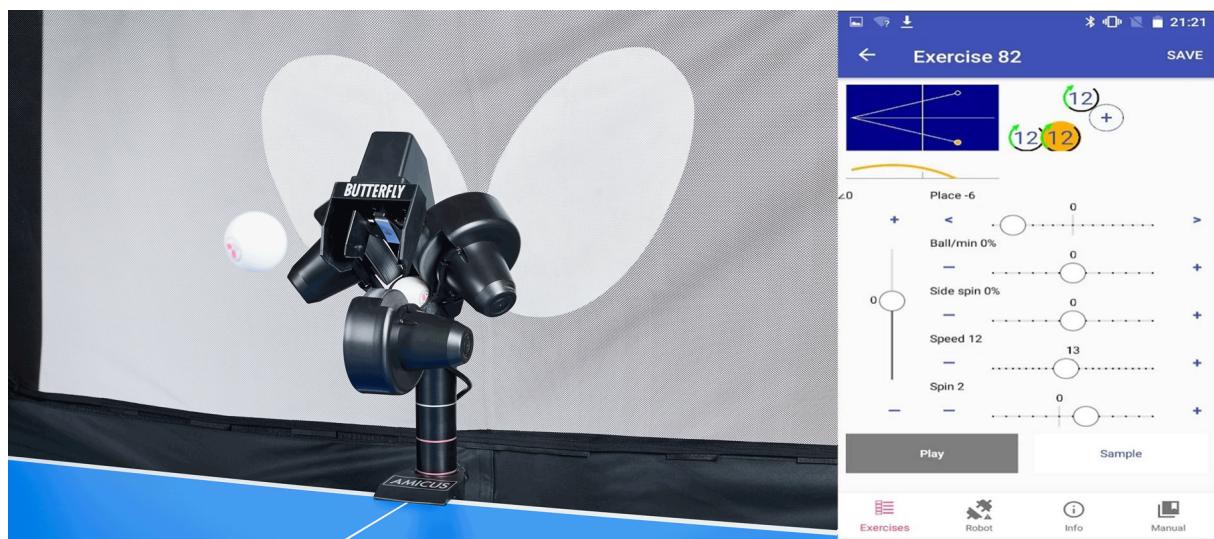


Slika 2.11. Model Amicus Expert s kontrolnom pločom[4]

### 2.3.1.3. Model Amicus Prime

Model Amicus Prime dolazi s tabletom na kojemu je instalirana aplikacija koja služi za upravljanje uređajem. Također tu aplikaciju je moguće instalirati na bilo koji pametni telefon. U aplikaciji se mogu programirati svi parametri ispučavanja loptica te je moguće spremanje bilo koji broj vježbi od kojih svaka može sadržavati do 10 različitih načina ispučavanja. Cijena uređaja iznosi \$2179.

Slika 2.12. prikazuje model Amicus Prime sa aplikacijom kojom se upravlja sa uređajem.



Slika 2.12. Model Amicus Prime s aplikacijom[4]

## 2.4. Usporedba postojećih proizvoda

Tablica 2.1. Usporedba postojećih proizvoda

Svojstvo	iPong			Newgy Industries			Tamasu Butterfly		
	Topspin	Original	Pro	1040+	2055	3050XL	Start	Expert	Prime
Dimenzije [mm]	320 x320 x480	320 x320 x480	320 x320 x480	330 x500 x230	1530 x790 x460	1530 x790 x460	1580 x860 x1600	1580 x860 x1600	1580 x860 x1600
Masa [kg]	1,14	1,14	1,14	6,5	11	8,2	8,8	8,8	8,8
Kontrola putanje loptice	X	X	X	Ručna	Ručna	Auto	Auto	Auto	Auto
Rotacija loptice <sup>1</sup>	H1	H2	H2 V2+	H2 V2+	H2 V2+	H2 V2+	H2 V2+	H2 V2+	H2 V2+
Uključuje mrežu za hvatanje loptica	X	X	X	X	✓	✓	✓	✓	✓
Nasumično ispučavanje loptica	X	X	✓	✓	✓	✓	✓	✓	✓
Bežično upravljanje	X	X	X	X	X	✓	X	X	✓
Frekvencija/kapacitet ispučavanja loptica/minuti (min-max)	12-70	12-70	12-70	26-94	1-170	1-170	11-100	1-100	5-120
Mogućnost programiranja uređaja	X	X	✓	X	✓ 64 vježbi	✓ 100+ vježbi	X	✓ 99 ∞ vježbi	✓ ∞ vježbi
Napajanje	Baterija	AC mreža	AC mreža	AC mreža	AC mreža	AC mreža	AC mreža	AC mreža	AC mreža
Kapacitet loptica	100	100	100	200	120+	120+	-	-	-
Cijena	\$99,95	\$149,95	\$199.95	\$399	\$999	\$1699	\$1279	\$1829	\$2179

Detaljnom analizom opisanih proizvoda vidimo da svaka tvrtka temelji svoje proizvode na jednoj tehnološkoj platformi i da na cijenu modela najviše utječe način upravljanja i mogućnost programiranja vlastitih vježbi. To je jedan od najkompleksnijih aspekata ove vrste uređaja i to jako utječe na cijenu proizvoda kao što je prikazano u tablici 2.1.

<sup>1</sup> H1/V1 = rotacija oko horizontalne/vertikalne osi samo u jednom smjeru

H2/V2 = rotacija oko horizontalne/vertikalne osi u dva smjera

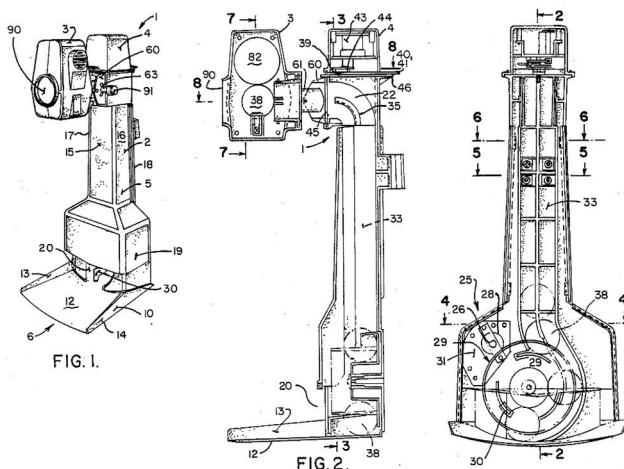
+ = kombinacija horizontalne i vertikalne rotacije

### 3. PREGLED PATENATA

#### 3.1. Patent US4844458A *Table tennis ball serving device* [5]

Patent je prijavljen 1985. godine u Americi, a istekao je 2006. godine.

Patent opisuje prijenosni uređaj za automatsko ispučavanje loptica za stolni tenis koji uključuje i mrežu za hvatanje lopte. Uredaj se sastoji od glave u kojoj je mehanizam za ispučavanje koja je montirana tako da se može rotirati oko osi smjera ispučavanja loptica. Prema glavi se dovode loptice iz vodilice koja je povezana s glavom i bazom uređaja. Za hvatanje loptica koje igrač vraća koristi se sklopiva mreža. Uredaj se nalazi unutar mreže, te povratne loptice, preko mehanizma za dovođenje i vodilice, dovodi ponovno do glave. Robotom upravljuju tri motora, koji se mogu pojedinačno kontrolirati kako bi se osiguralo kontinuirano i raznoliko ispučavanje loptica. Konstrukcija robota osigurava razne tehnike ispučavanja loptica, koje se sve kontroliraju tijekom rada uređaja.



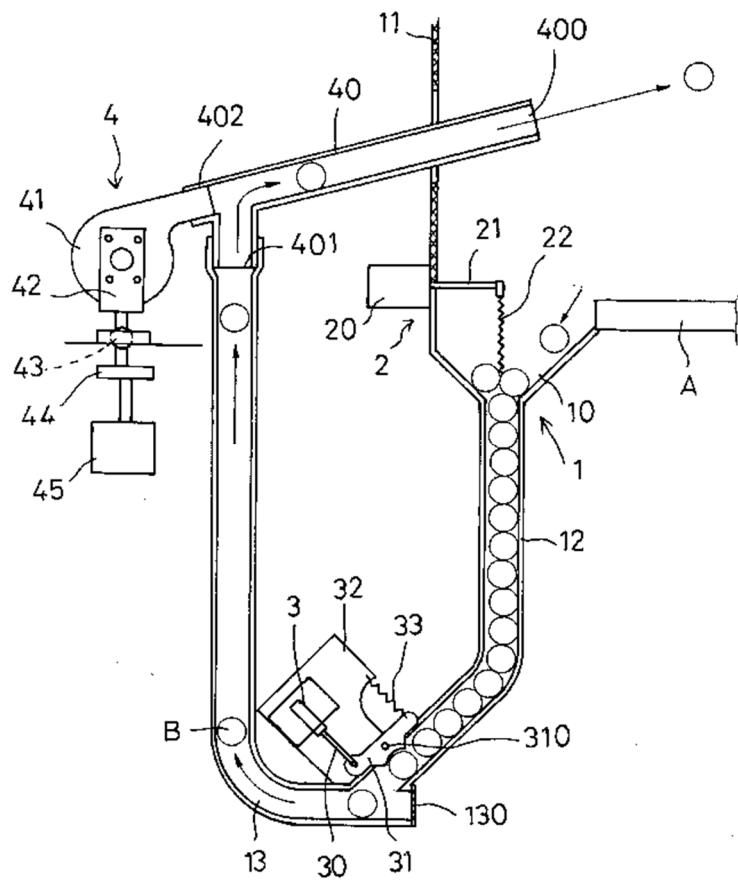
Slika 3.1. Patent US4844458A[5]

Slika 3.1. Prikazuje uređaj za ispučavanje loptica za stolni tenis. Na slici je prikazana glava (3) u kojoj se nalazi mehanizam za ispučavanje loptica. Mehanizam za ispučavanje loptica konstruiran je pomoću jednog elektromotora na kojem se nalazi tarenica (82) koja u zahvatu s lopticom (38) izbacuje lopticu određenom brzinom i rotacijom. Nakon što igrač vrati lopticu u mrežu, loptica pada na bazu (12) uređaja te se preko mehanizma za dovođenje (30) i vodilice (33) ponovno vraća u glavu (3) uređaja i proces se ponavlja. Mehanizam za dovođenje (30) sastoji se od jednog elektromotora na kojemu se nalazi bubanj (29) koji je izrađen tako da na sebi ima lopatice koje služe za kupljenje i podizanje loptica duž vodilice (33).

### 3.2. Patent US20070221187A1 *Table tennis ball service machine* [6]

Patent je prijavljen 2006. godine u Americi te je još uvijek aktivan.

Patent opisuje uređaj za ispučavanje loptica za stolni tenis koji se sastoji sklopa za sakupljanje loptica, upravljačkog ventila i mehanizma za ispučavanje loptica. Sklop za sakupljanje loptica sastoji se od mreže za hvatanje povratnih loptica, žlijeba za skupljanje loptica koji je spojen sa stolom te je donji dio žlijeba pomoću cijevi spojen s upravljačkim ventilom. Upravljački ventil smješten je na donjem kraju cijevi za sakupljanje loptica te kontrolira frekvenciju ispučavanja loptica. Mehanizam za ispučavanje loptica povezan je s gornjom stranom cijevi za dovođenje loptica i sastoji se od cijevi za ispučavanje, kompresora i nosive konstrukcije. Cijev za ispučavanje loptica je jednim krajem povezana na kompresora, a drugi kraj je, kroz mrežu za hvatanje loptica, usmjeren prema igraču.

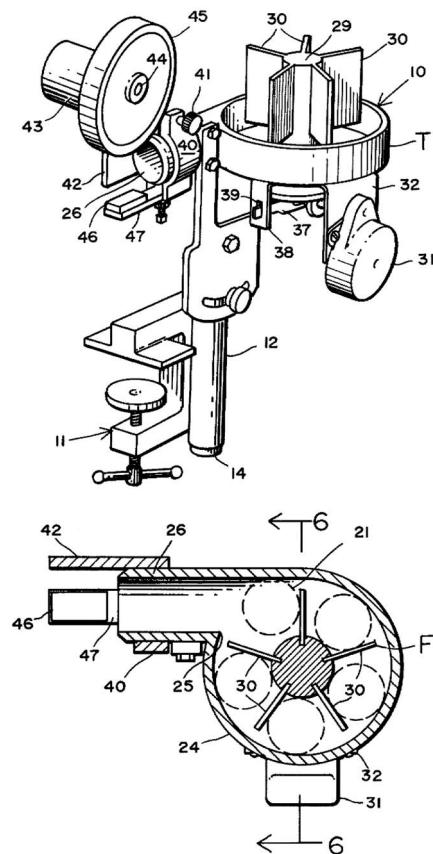


Slika 3.2. Patent US20070221187A1[6]

### 3.3. Patent US3878827A *Table tennis ball serving apparatus* [7]

Patent je prijavljen 1973. godine u Americi, a istekao je 1992. godine.

Ovaj patent predstavlja uređaj za ispučavanje loptica s mehanizmom koji dovodi loptice do tarenice koja svojom rotacijom zahvaća lopticu te ju ispučava u željenom smjeru i željenom rotacijom. Rotacija loptice može biti *top spin* i *bottom spin*. Mehanizam za dovođenje loptice do tarenice sastoji se od rotirajućih lopatica među kojima se nalaze loptice. Prilikom rotacije, lopatice guraju i usmjeravaju loptice prema mehanizmu za ispučavanje (prema tarenici). Cijeli uređaj ima mehanizam koji služi za učvršćivanje uređaja za stol i nagib uređaja.



Slika 3.3. Patent US3878827A[7]

## 4. TEHNIČKA SPECIFIKACIJA

Na temelju gabarita uređaja, maksimalne brzine ispučavanja loptica te upravljivosti uređaja kreirat će se tehnička specifikacija.

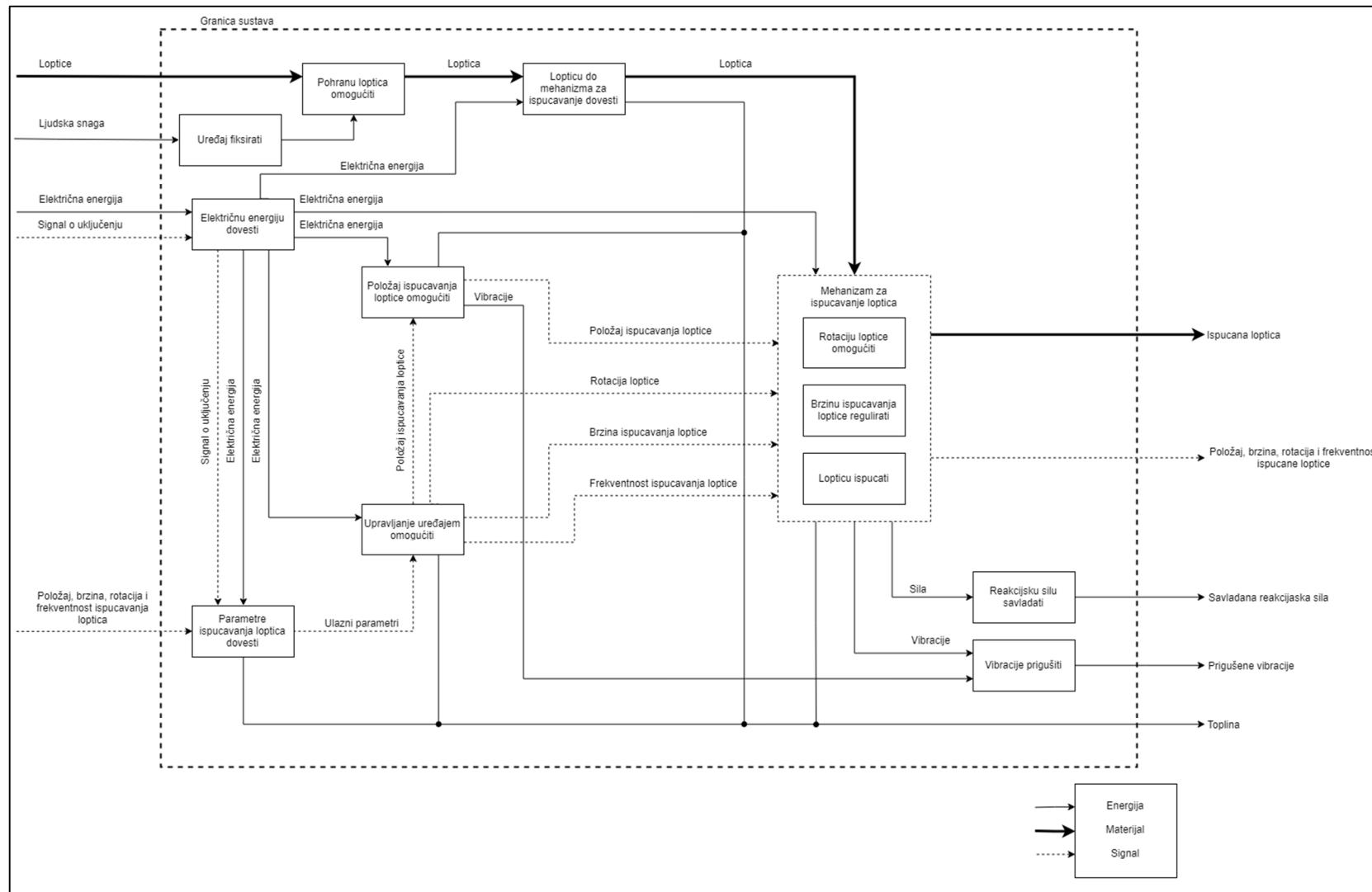
Početne tehničke specifikacije s kojima se krenulo u daljnju razradu, odnosno izradu funkcijске dekompozicije, morfološke matrice i koncepata su sljedeće [Tablica 4.1]:

**Tablica 4.1. Tehnička specifikacija**

Napajanje	Punjive baterije (12V)
Automatsko zakretanje uređaja	✓
Automatsko naginjanje uređaja	✓
Brzina ispučavanja loptice	10-30m/s
Frekvencija ispučavanja loptica [loptica/minuti]	1-120
Kapacitet loptica	30
Težina uređaja	1,5kg
Gabariti [mm]	300x250x200
Bežično upravljanje	✓
Mogućnost programiranja uređaja	✓ ( $\infty$ vježbi)

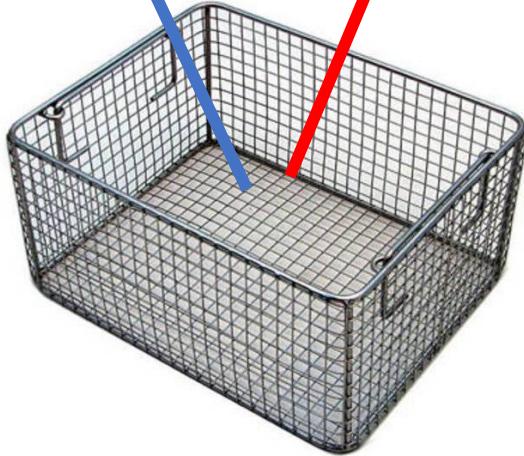
Na temelju tehničke specifikacije izrađuje se funkcijска dekompozicija u kojoj se definiraju sve funkcije uređaja te se izrađuje morfološka matrica kod koje se za svaku funkciju uređaja predlaže parcijalno rješenje, koje se nakon toga koristi pri izradi koncepata.

## 5. FUNKCIJSKA DEKOMPOZICIJA

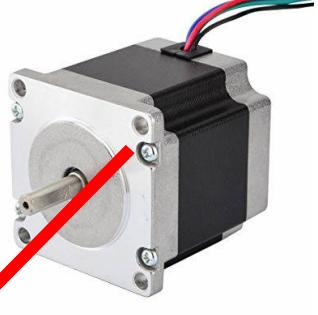
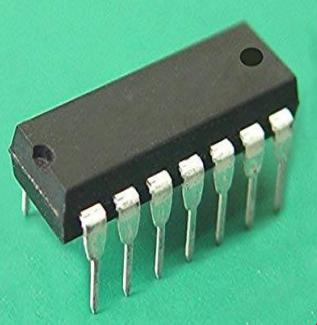
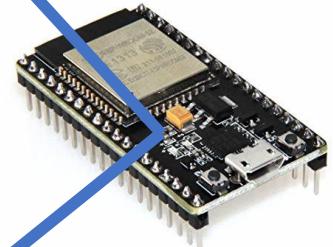
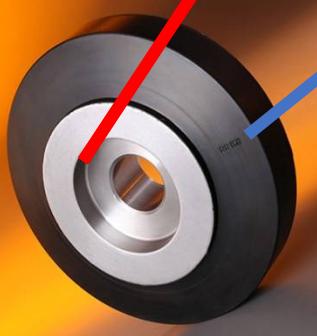


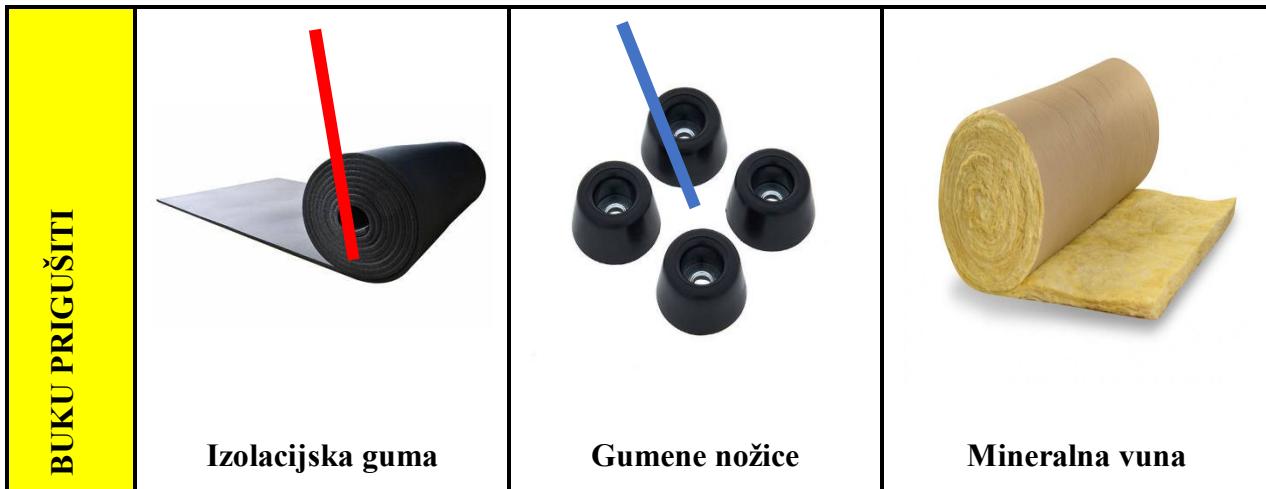
## 6. MORFOLOŠKA MATRICA PARCIJALNIH RJEŠENJA

Tablica 6.1. Morfološka matrica rješenja

FUNKCIJA	RJEŠENJA	
UREĐAJ FIKSIRATI	 <b>Gumene nožice</b>	 <b>Vijčana veza za stol</b>
POHRANITI LOPTICE	 <b>Košara</b>	

ELEKTRIČNA ENERGIJA	Baterija	Spajanje na električnu mrežu	Solarna energija
PARAMetri ISPUCAVANJA LOPTICE	Kontrolna ploča	Pametni telefon	
DOVODENJE LOPTICE DO MEHANIZMA ZA ISPUCAVANJE	Spiralni kavez	Rotirajuće lopatice	

POZICIONIRANJE SMJERA ISPUCAVANJA			
UPRAVLJANJE UREDAJEM			
MEHANIZAM ZA ISPUCAVANJE LOPTICA			

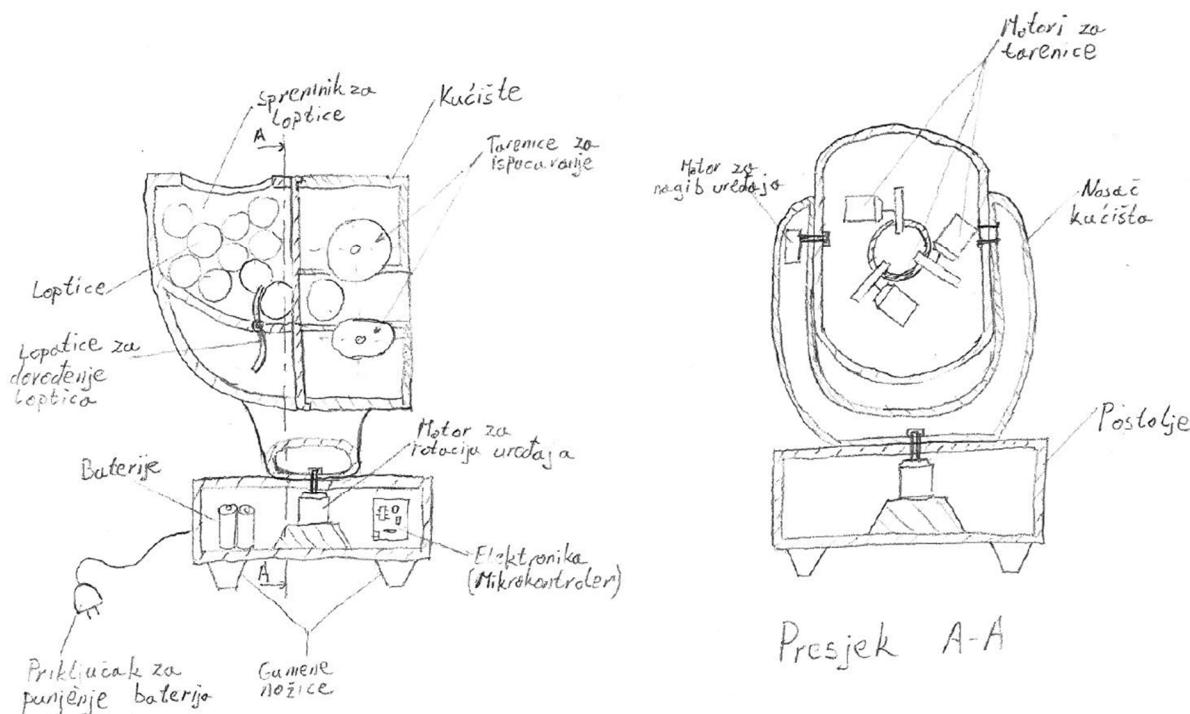


U morfološkoj matrici [Tablica 6.1.] dana su neka od mogućih rješenja za funkcije proizvoda. Iz njih će se izraditi nekoliko konceptualnih rješenja koja će se zatim vrednovati i usporediti. Treba napomenuti da rješenja u morfološkoj matrici nisu razrađena do detalja, već je u obzir uzeta samo ideja i određene karakteristike, kao što su težina izvedbe, cijena izvedbe, funkcionalnost, masa, zadovoljavanje potreba korisnika, itd. Odabrana rješenja za pojedine koncepte spojena su linijama u morfološkoj matrici (koncept 1 – crvena linija; koncept 2 – plava linija).

## 7. KONCEPTI

Nakon definiranja funkcijskog modela i morfološke matrice, nastavljamo s izradom koncepata koji se temelje na rješenjima danim u morfološkoj matrici. Nakon izrade koncepata slijedi njihova evaluacija te vrednovanje.

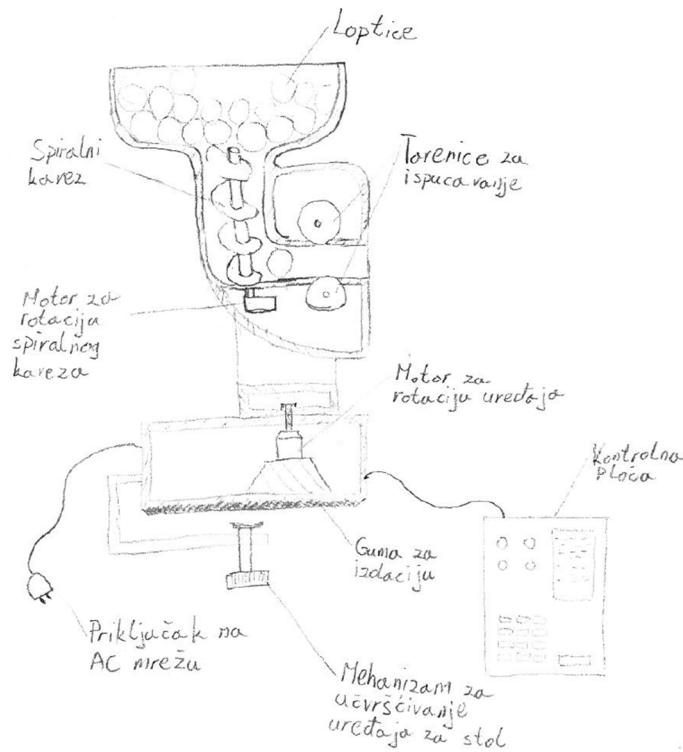
### 7.1. Koncept 1



Slika 7.1. Koncept 1

Na skici [Slika 7.1] prikazano je rješenje prvog koncepta. Spremnik za loptice potrebno je napuniti lopticama. Nakon podešavanja željenih parametara ispučavanja pomoću mobilne aplikacije, podaci se s aplikacije šalju na mikrokontroler preko *Bluetooth-a*. Najprije se, pomoću motora za rotaciju i motora za zakretanje, uređaj postavi u određeni položaj tako da se ispune uvjeti zadani u aplikaciji. Zatim se motori, na kojima su tarenice, počinju rotirati brzinama koje su također definirane u aplikaciji, te se na kaju počinju rotirati i lopatice koje dovode loptice do tarenica koje, u zahvatu s lopticom, ispučavaju lopticu na željeno mjesto. Taj se postupak ponavlja sve dok ne ponestane loptica u spremniku ili dok se ne završi vježba. Za napajanje uređaja koriste se ugrađene punjive baterije koje bi se punile na sličan način kao i slična elektronika (prijenosna računala, pametni telefoni, itd.)

## 7.2. Koncept 2



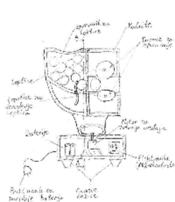
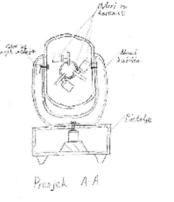
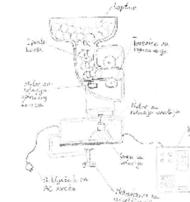
**Slika 7.2. Koncept 2**

Slika 7.2 prikazuje rješenje drugog koncepta. U drugom konceptu koristi se spiralni kavez kao sredstvo za dovođenje loptica do tarenica. Nakon što se uređaj pozicionira i tarenice počnu rotirati, poseban motor počinje okretati spiralni kavez oko vertikalne osi koji tako dovodi loptice do tarenica. Za upravljanje je korištena kontrolna ploča koja je s kabelom povezana sa uređajem. Izveden je s većim spremnikom za loptice te je fiksiran za stol preko vijčane veze. Za napajanje uređaja koristi se isključivo napajanje iz mreže.

## 7.3. Vrednovanje koncepata

Tablica vrednovanja [Tablica 7.1] sadrži kriterije za odabir koncepta za daljnju razradu i proračun. Koncept 1 i Koncept 2 međusobno će se usporediti u odnosu na svaki postavljeni kriterij. Kriteriji za vrednovanje koncepata odabrani su tako da uređaj može zadovoljiti sve potrebe korisnika, da bude što jednostavniji za korištenje i da cijena proizvoda bude prihvatljiva većini korisnika. Koncept koji bolje zadovoljava određeni kriterij dobit će plus(+) dok će drugi dobiti minus(-). Koncept koji zadovoljava više kriterija bit će razrađen u idućim fazama rada. Daljnja razrada će biti zaključena modeliranjem uređaja u programskom paketu Solid Edge 2019 te izradom tehničke dokumentacije.

Tablica 7.1. Vrednovanje koncepta

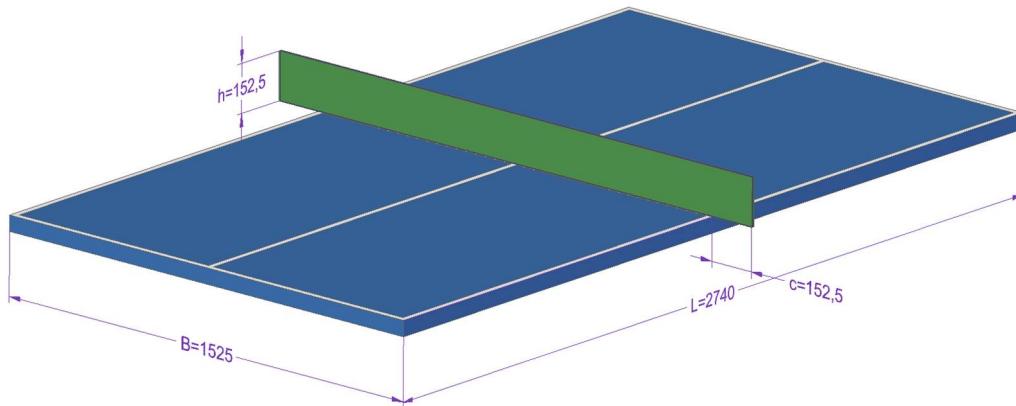
KRITERIJ	OCJENA	
	KONCEPT 1	KONCEPT 2
	 	 
Masa uređaja	+	-
Cijena izrade	-	+
Težina izrade	-	+
Jednostavnost korištenja	+	-
Dimenzije uređaja	+	-
Prenosivost	+	-
Funkcionalnost	+	+
Spremnik za loptice	-	+
<b>SUMA</b>	<b>5</b>	<b>4</b>

Koncept 1 dobio je veću ukupnu ocjenu te će se koristiti kao podloga za daljnju konstrukciju uređaja. Iako će cijena uređaja biti malo veća, bitan faktor je da uređaj bude čim jednostavniji za korištenje te da je lako prenosiv. Uređaj će se konstruirati tako da će se moći postaviti na bilo koju poziciju na stolu za stolni tenis te se tako povećava raznolikost ispučavanja loptica.

## 8. KONSTRUKCIJSKA RAZRADA

### 8.1. Dimenziije stola i brzina loptice

Za odabir motora za ispučavanje loptice i servo motora za pozicioniranje uređaja potrebno je znati sve dimenzije stola za stolni tenis te prosječnu i maksimalnu brzinu loptice. Slika 8.1. prikazuje sve dimenzije stola. Sve dimenzije izražene su u milimetrima.



Slika 8.1. Dimenziije stola za stolni tenis

Prilikom istraživanja [8] utvrđeno je da je prosječna brzina loptice za stolni tenis, tijekom igre,  $11 \text{ m/s}$ , a maksimalna izmjerena brzina je  $30 \text{ m/s}$ .

### 8.2. Odabir motora i tarenica za ispučavanje loptica i servo motora za nagib i zakret uređaja

#### 8.2.1. Motori i tarenice za ispučavanje loptica

Za tarenice odabrani su diskovi promjera 42mm koji po obodu imaju gumu da bi se uklonilo proklizavanje između loptice i tarenice. Razlog odabira baš takvih tarenica bio je iterativan na temelju broja okretaja motora, a i zbog praktičnih razloga jer postoje takve standardne tarenice te također povećavaju kompaktnost uređaja. Jedna od iteracija prikazana je u nastavku.



Slika 8.2. Tarenica za ispučavanje loptica

Da bi odabrali motor za ispučavanje loptica trebamo odrediti moment potreban za ubrzanje loptice do maksimalne brzine ispučavanja loptice.

- $m = 2,7g$  - masa loptica
- $s \approx 0,03m$  - put na kojem loptica ubrzava
- $v_{max} = 30m/s$  - maksimalna brzina ispučavanja
- $d_t = 0,042m$  - promjer tarenice

Minimalno vrijeme ubrzavanja loptice iznosi:

$$t_{min} = \frac{s}{v_{max}} = \frac{0.03}{30} = 1 \cdot 10^{-3}s, \quad (8.1)$$

potrebna akceleracija da se dostigne maksimalna brzina:

$$a_{max} = \frac{v_{max}}{t_{min}} = \frac{30}{1 \cdot 10^{-3}} = 30000m/s^2, \quad (8.2)$$

sila na obodu tarenice da se postigne maksimalno ubrzanje:

$$F_{max} = m \cdot a_{max} = 0,0027 \cdot 30000 = 81N, \quad (8.3)$$

potreban moment motora da se postigne maksimalna sila na obodu tarenice:

$$M_{max} = F_{max} \cdot \frac{d_t}{2} = 121,5 \cdot 0,02 = 1,62Nm, \quad (8.4)$$

pošto se koriste 3 motora, maksimalan moment na jednom motoru je  $M_{max} = 0,54Nm$ .

Odabrana su 3 motora *Turnigy D2822/14 Outrunner Brushless Motor*. Tablica 8.1 sadrži karakteristike motora za ispučavanje loptica

**Tablica 8.1. Karakteristike motora za ispučavanje loptica**

Tip motora	DC motor bez četkica
Napajanje	12V
Broj okretaja	1450o/min/V
Maksimalna iskoristivost	80%
Struja praznog hoda	0,5A
Maksimalna struja	13A
Maksimalna snaga	160W
Masa	38g
Dimenzije	$\varnothing 28mm \times 28mm$
Promjer izlaznog vratila	3,2mm



**Slika 8.3. Motor za ispučavanje loptice**

Na temelju podataka iz Tablica 8.1 te podatka da je promjer tarenica 42 mm, možemo dobiti da je maksimalna moguća obodna brzina na tarenicama uz odabrane motore:

$$v_{Tmax} = \frac{n_V \cdot U \cdot d_r \cdot \pi}{60} = \frac{1450 \cdot 12 \cdot 0,042 \cdot \pi}{60} = 38,26m/s, \quad (8.5)$$

$$v_{max} = 30m/s < v_{Tmax} = 38,26m/s \quad -\text{ZADOVOLJAVA.}$$

Za odabrane motore također je potrebno koristiti odgovarajuće kontrolere. Oni služe za generiranje trofaznog napona kojem promjenom frekvencije utječe na brzinu rotacije motora. Stoga su odabrana 3 kontrolera *G-Sun 20A ESC Brushless motor Speed Controller*. Tablica 8.2 prikazuje karakteristike kontrolera za motore.

**Tablica 8.2. Karakteristike kontrolera za motore**

Nazivna struja	20A
Maksimalna struja	25A
Napajanje	8,4V – 12V
Dimenzije	48mm x 26mm x 8mm
Masa	20g



**Slika 8.4. Kontroler za motore**

### 8.2.2. Servo motor za nagib i zakret uređaja

Za nagib i zakret uređaja korišteni su servo motori. To su motori kojima je moguće pozicionirati i pratiti zakret izlaznog vratila. Izlazno vratilo u princip nije moguće kontinuirano okretati već je moguće samo zakretanje samo unutar radnoga kuta, obično oko  $180^\circ$ . Odabrani su servo motori *Tower pro SG90*. Za njihov odabir nije korišten posebni proračun jer su to jedni od najmanjih i najjeftinijih servo motora na tržištu koji po svojima karakteristikama zadovoljavaju potrebne zahtjeve za upravljanje uređajem. Za rotaciju i nagib uređaja potreban je jako mali okretni moment ( $\approx 0.05Nm$ ), jer je masa samog uređaja mala te će konstrukcija uređaja biti takva da se rotacija i nagib vrše u centru težišta uređaja kako bi se smanjilo opterećenje servo motora. Tablica 8.3 prikazuje njihove karakteristike.

**Tablica 8.3. Karakteristike servo motora**

Napajanje	4,8V – 6V
Nazivni moment	0,2Nm
Brzina	0,1s/ $60^\circ$
Radni kut	$175^\circ$
Promjer izlaznog vratila	5mm
Dimenzije	23mm x 12mm x 29mm
Masa	14,7g



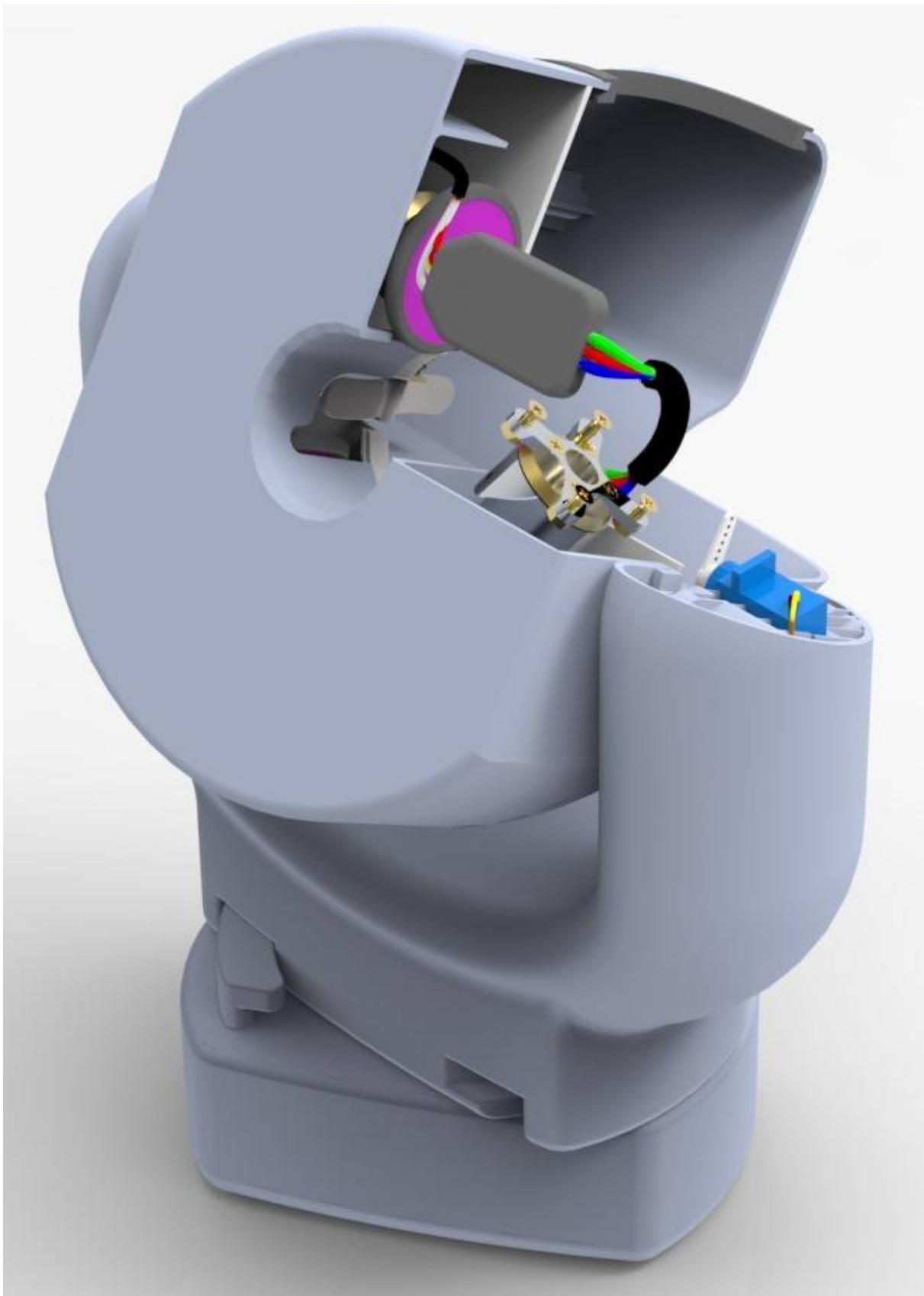
**Slika 8.5. Servo motor**

### 8.3. Oblikovanje rješenja

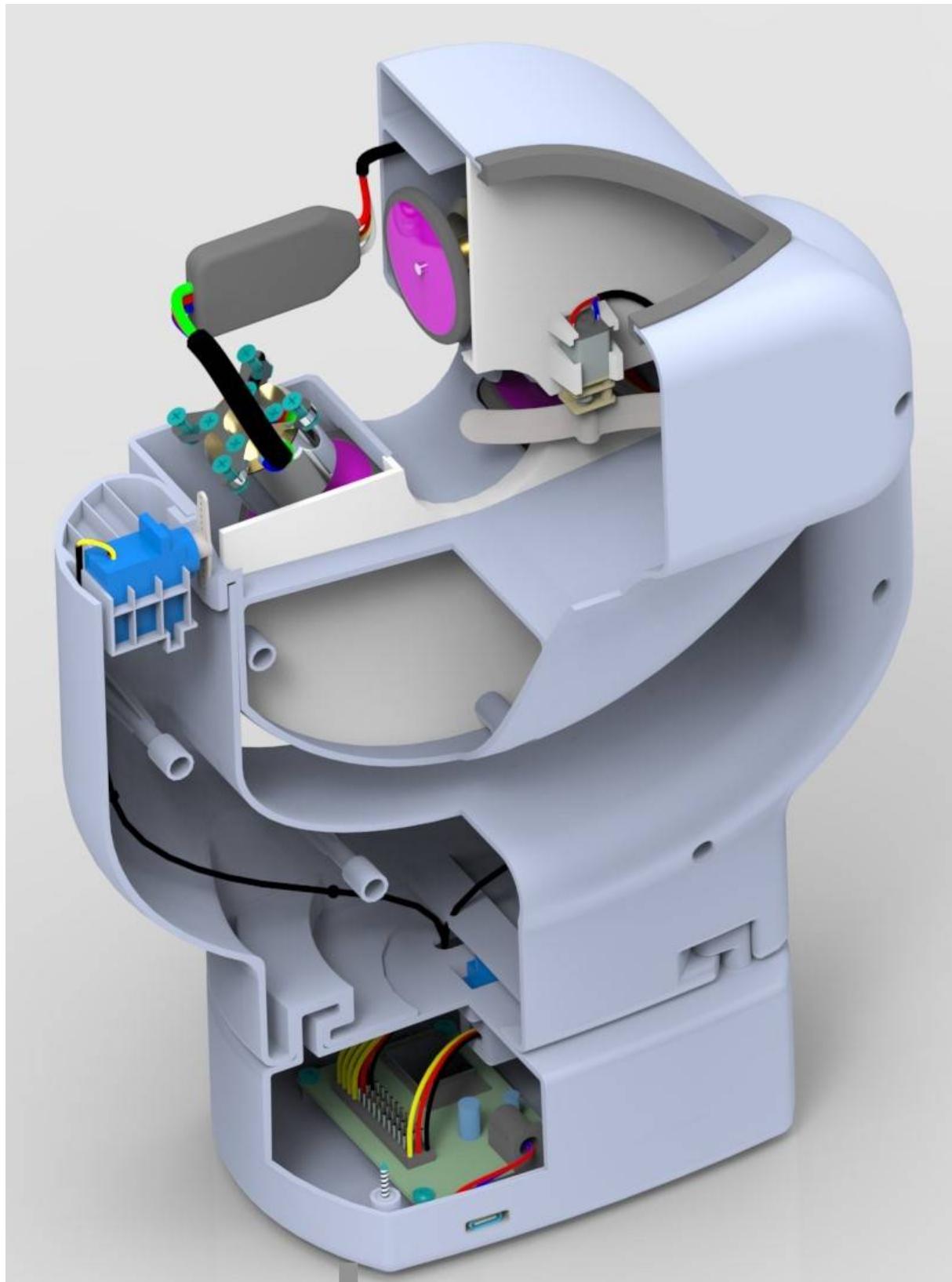
Na temelju skice [Slika 7.1] kreiran je 3D model komponenti i uređaja za ispučavanje loptica za stolni tenis korištenjem programskog paketa Solid Edge 2019. Polazište za izradu modela bio je mehanizam za ispučavanje loptica (3 motora s tarenicama). Nakon što se odredio položaj motora, oko njih se je gradilo plastično kućište tako da uređaj bude što kompaktniji. Sastoji se od 3 glavna dijela, kućišta, nosača kućišta te postolja. Kućište sadrži mehanizam za ispučavanje loptica, spremnik za loptice i mehanizam za dovođenje loptica. U nosaču kućišta nalazi se mehanizam za nagib, a u postolju se nalazi mehanizam za zakret uređaja te upravljački mikrokontroler. Sljedećih nekoliko slika prikazuje izrađeni model [Slika 8.6, Slika 8.7 i Slika 8.8]



Slika 8.6. Uredaj za ispučavanje loptica za stolni tenis - Pogled 1



Slika 8.7. Uredaj za ispučavanje loptica za stolni tenis - Pogled 2



Slika 8.8. Uredaj za ispučavanje loptica za stolni tenis - Pogled 3

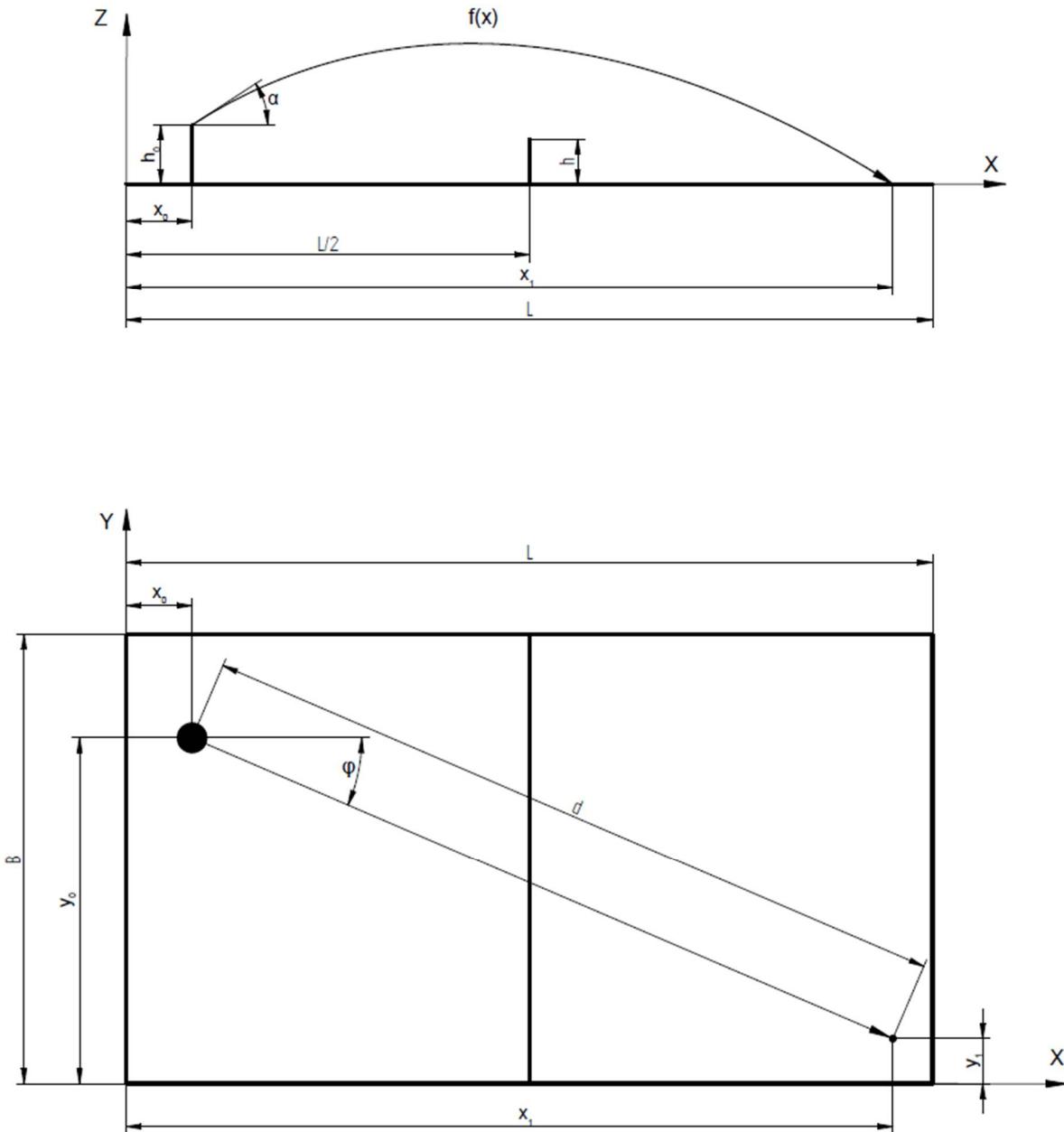
#### 8.4. Izvod jednadžbi za ispučavanje loptice

Nakon izrade modela potrebno je dobiti način kojim će se lako moći povezati želja korisnika te izvršne jedinice uređaja. Recimo da korisnik postavi uređaj na neko mjesto na stolu za stolni tenis te preko sučelja u aplikaciji označi gdje se nalazi uređaj, poziciju na koju loptica treba udariti te željenu rotaciju loptice. Da bi to mogli postići potrebno je dobiti nekoliko jednadžbi koje opisuju rotaciju i putanju loptice u zraku. Rješavanjem tih jednadžbi, pomoću ulaznih parametara koje dobijemo od korisnika, moći će se dobiti kut nagiba i zakreta uređaja te brzina rotacije svih triju motora za ispučavanje loptice.

U izvodu svih jednadžbi pretpostavka je bila da nema klizanja između loptica i tarenica te nije uzet otpor zraka i Magnusov efekt radi jednostavnosti proračuna. Takav pristup je odabran zato što se ne radi o nekom kritičnom sustavu te je moguće u programskom kodu, nakon izrade prototipa, ubaciti odgovarajuće konstante, dobivene testiranjem, koji će korigirati eventualne gubitke.

Slijedi izvod jednadžbi za putanju loptice:

- $L = 2,74m$  - dužina stola
- $B = 1,525m$  - širina stola
- $h = 0,1525m$  - visina mrežice
- $h_0 = 0,2m$  - visina uređaja
- $g = 9,81 \text{ m/s}^2$  - ubrzanje slobodnog pada
- $x_0$  - x koordinata uređaja
- $y_0$  - y koordinata uređaja
- $\alpha$  - kut nagiba uređaja
- $\varphi$  - kut zakreta uređaja
- $v$  - brzina ispučavanja loptice
- $x_1$  - x koordinata na stolu na koju se ispučava loptica
- $y_1$  - y koordinata na stolu na koju se ispučava loptica
- $d$  - udaljenost koju mora prijeći loptica



Slika 8.9. Putanja loptice: gornja slika-načrt stola; donja slika-tlocrt stola

Varijable  $x_0$ ,  $y_0$ ,  $x_1$ ,  $y_1$  i  $v$  biti će poznate nakon što ih korisnik unese u aplikaciju za upravljanje uređajem. Te na temelju tih podataka možemo izračunati kut zakreta uređaja i kut nagiba uređaja.

Kut zakreta uređaja:

$$\varphi = \arctg \left( \frac{y_1 - y_0}{x_1 - x_0} \right). \quad (8.6)$$

Kut nagiba uređaja izračunava se preko jednadžbe kosog hitca:

$$f(x) = -\frac{g}{2v \cdot \cos(\alpha)} \cdot x^2 + \tan(\alpha) \cdot x + h_0, \quad (8.7)$$

ako jednadžbu 8.7 izjednačimo s 0 u točki  $x=d$  dobivamo:

$$-\frac{g}{2v \cdot \cos(\alpha)} \cdot d^2 + \tan(\alpha) \cdot d + h_0 = 0, \quad (8.8)$$

$$-gd^2 + 2vd \cdot \sin(\alpha) + 2vh_0 \cdot \cos(\alpha) = 0, \quad (8.9)$$

$$d \cdot \sin(\alpha) + h_0 \cdot \cos(\alpha) = \frac{gd^2}{2v}. \quad (8.10)$$

Rješavanjem jednadžbe 8.10 pomoću programa Wolfram Alpha dobivamo da nagib uređaja iznosi:

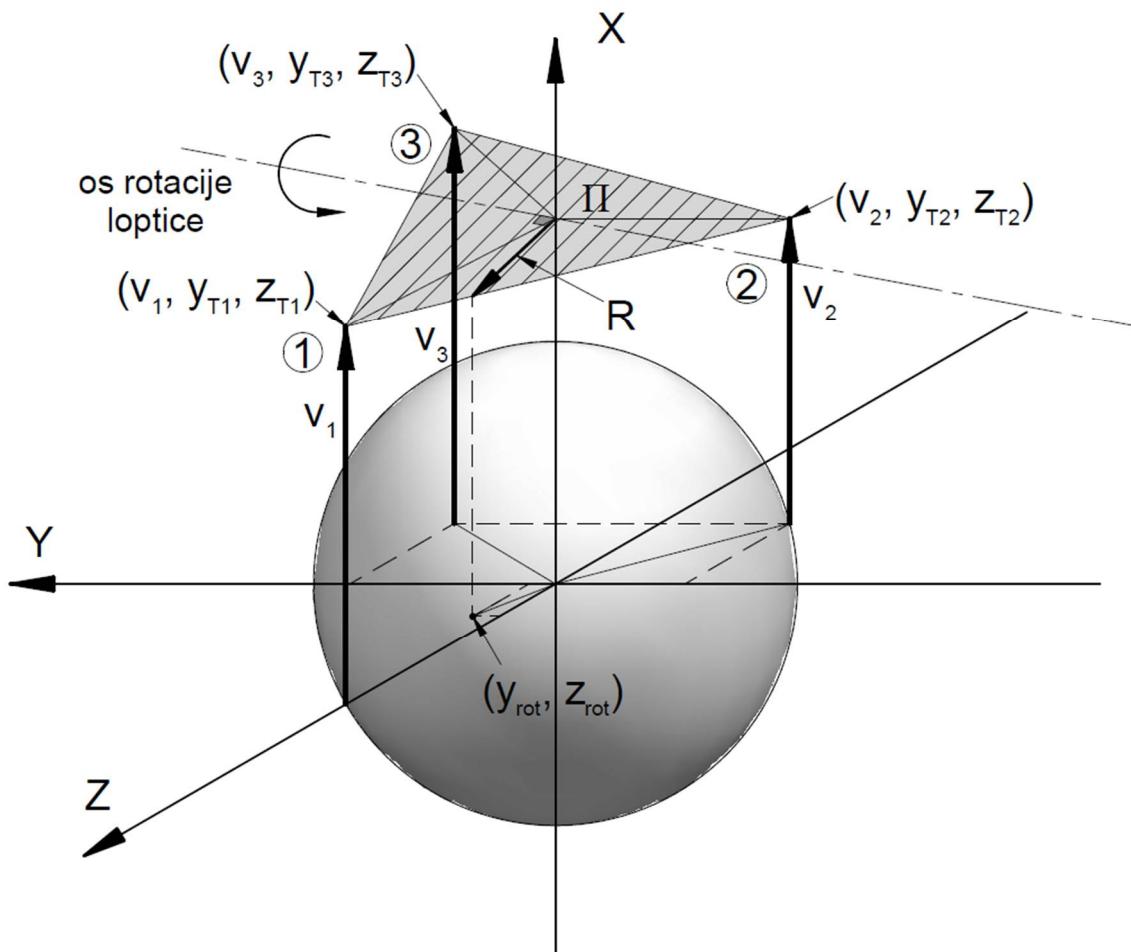
$$\alpha = 2 \cdot \operatorname{atg} \left( \frac{d - \sqrt{d^2 + h_0^2 - \left( \frac{gd^2}{2v} \right)^2}}{h_0 + \frac{gd^2}{2v}} \right). \quad (8.11)$$

## 8.5. Izvod jednadžbi za rotaciju loptice

Nakon što smo odredili kut zakreta i nagiba uređaja potrebno je odrediti i brzinu rotacije svakog motora za ispučavanje loptica s obzirom na željenu rotaciju loptice. Iz konstrukcije je vidljivo da su moguće rotacije loptice oko Y i Z osi. Pošto su motori postavljeni u YZ ravnini, te su zakrenuti  $120^\circ$  jedan od drugoga, moguće je odrediti rotaciju i jačinu rotacije oko bilo koje osi koja leži na toj ravnini. To se postiže tako da se za željenu rotaciju odrede brzine rotacija svakog motora zasebno. Te brzine rotacija opisane su sljedećim jednadžbama.

- $d_L = 0,04m$  - promjer loptice za stolni tenis
- $y_{T1} = 0$  - y koordinata tarenice 1
- $z_{T1} = 0,02$  - z koordinata tarenice 1
- $y_{T2} = -0,01732$  - y koordinata tarenice 2
- $z_{T2} = -0,01$  - z koordinata tarenice 2
- $y_{T3} = 0,01732$  - y koordinata tarenice 3

- $z_{T3} = -0,01$  - z koordinata tarenice 3
- $v_1$  - vektor koji predstavlja obodnu brzinu tarenice 1
- $v_2$  - vektor koji predstavlja obodnu brzinu tarenice 2
- $v_3$  - vektor koji predstavlja obodnu brzinu tarenice 3
- $\Pi$  - ravnina koju tvore vrhovi vektora  $v_1, v_2$  i  $v_3$
- $R$  - vektor koji predstavlja najbrži pad ravnine  $\pi$
- $y_{rot}$  - y koordinata vektora  $R$
- $z_{rot}$  - z koordinata vektora  $R$



Slika 8.10. Rotacija loptice

Slika 8.10 prikazuje lopticu za stolni tenis koja se nalazi u ishodištu koordinatnog sustava te vektore  $v_1, v_2$  i  $v_3$  koji predstavljaju obodne brzine tarenica koje nalaze u YZ ravnini zakrenute za  $120^\circ$ . Vrhovi vektora  $v_1, v_2$  i  $v_3$  tvore ravninu iz koje dobivamo vektor  $R$  koji predstavlja najbrži pad ravnine  $\Pi$ . Os oko koje će se rotirati loptica je okomita na taj vektor  $R$  te leži na ravnini  $\Pi$ . Sljedeće jednadžbe opisuju rotaciju loptice.

Ako počnemo od toga da korisnik određuje smjer i jačinu rotacije loptice te brzinu ispučavanja loptice, možemo zaključiti da su nam podaci,  $y_{rot}$ ,  $z_{rot}$  i  $v$ , poznati. Potrebno je dobiti jednadžbe kojima možemo na temelju tih poznatih podataka odrediti vrijednosti vektora  $v_1$ ,  $v_2$  i  $v_3$ . To možemo odrediti tako da dobijemo jednadžbu ravnine  $\Pi$  na sljedeći način.

Jednadžba ravnine kroz 3 točke:

- $(v_1, y_{T1}, z_{T1})$
- $(v_2, y_{T2}, z_{T2})$
- $(v_3, y_{T3}, z_{T3})$

$$\begin{vmatrix} x - v_1 & y - y_{T1} & z - z_{T1} \\ v_2 - v_1 & y_{T2} - y_{T1} & z_{T2} - z_{T1} \\ v_3 - v_2 & y_{T3} - y_{T2} & z_{T3} - z_{T2} \end{vmatrix} = 0. \quad (8.12)$$

Nakon uvrštavanja brojeva i rješavanja jednadžbe 8.12 u programu Matlab dobivamo:

$$\begin{aligned} 0,001039 \cdot x + 0,03(v_2 - v_3) \cdot y - 0,01732(2v_1 - v_2 - v_3) \cdot z \\ = 0,0003464(v_1 + v_2 + v_3). \end{aligned} \quad (8.13)$$

Dijeljenjem cijele jednadžbe s 0,001039 i premještanjem svih članova na desnu stranu osim x dobivamo jednadžbu ravnine:

$$x = \frac{v_1 + v_2 + v_3}{3} + (v_3 - v_2) \cdot 28,8675y + \frac{2v_1 - v_2 - v_3}{3} \cdot 50z, \quad (8.14)$$

ako u jednadžbu 8.14 uvrstimo  $y = z = 0$  dobivamo:

$$\frac{v_1 + v_2 + v_3}{3} = x = v. \quad (8.15)$$

To nam govori da će ukupna brzina loptice biti jednaka prosječnoj obodnoj brzini svih 3 tarenica što ima smisla.

Da bi dobili poveznicu između jednadžbe ravnine  $\pi$  i vektora  $R$ , koji nam je poznat, potrebno je parcijalno derivirati jednadžbu ravnine po koordinati  $y$  i  $z$ .

$$\frac{\delta x}{\delta y} = (v_3 - v_2) \cdot 28,8675 = y_{rot}, \quad (8.16)$$

$$\frac{\delta x}{\delta z} = \frac{2v_1 - v_2 - v_3}{3} \cdot 50 = z_{rot}. \quad (8.17)$$

Nakon što smo dobili jednadžbe 8.15, 8.16 i 8.17 možemo ih zapisati u matričnom obliku i riješiti tako da dobijemo  $v_1$ ,  $v_2$  i  $v_3$ :

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & -28,8675 & 28,8676 \\ \frac{100}{3} & -\frac{50}{3} & -\frac{50}{3} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v \\ y_{rot} \\ z_{rot} \end{bmatrix}, \quad (8.18)$$

nakon rješavanja jednadžbe 8.18 dobivamo izraze za obodne brzine svih 3 tarenica na temelju ulaznih podataka koje korisnik unosi preko aplikacije:

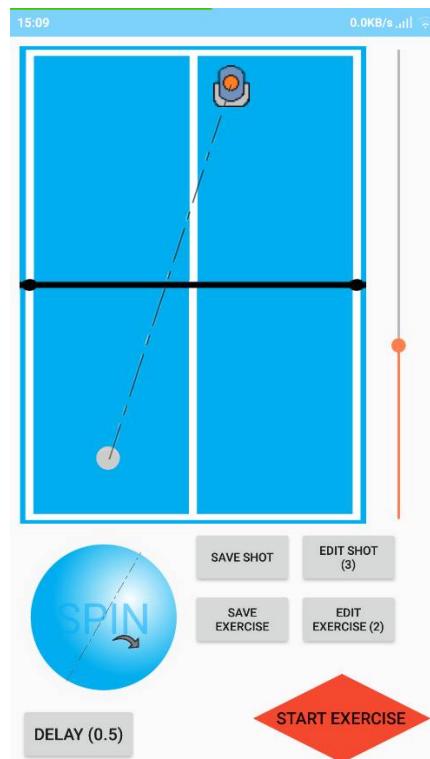
$$v_1 = v + 0,02z_{rot}, \quad (8.19)$$

$$v_2 = v - 0,01732y_{rot} - 0,01z_{rot}, \quad (8.20)$$

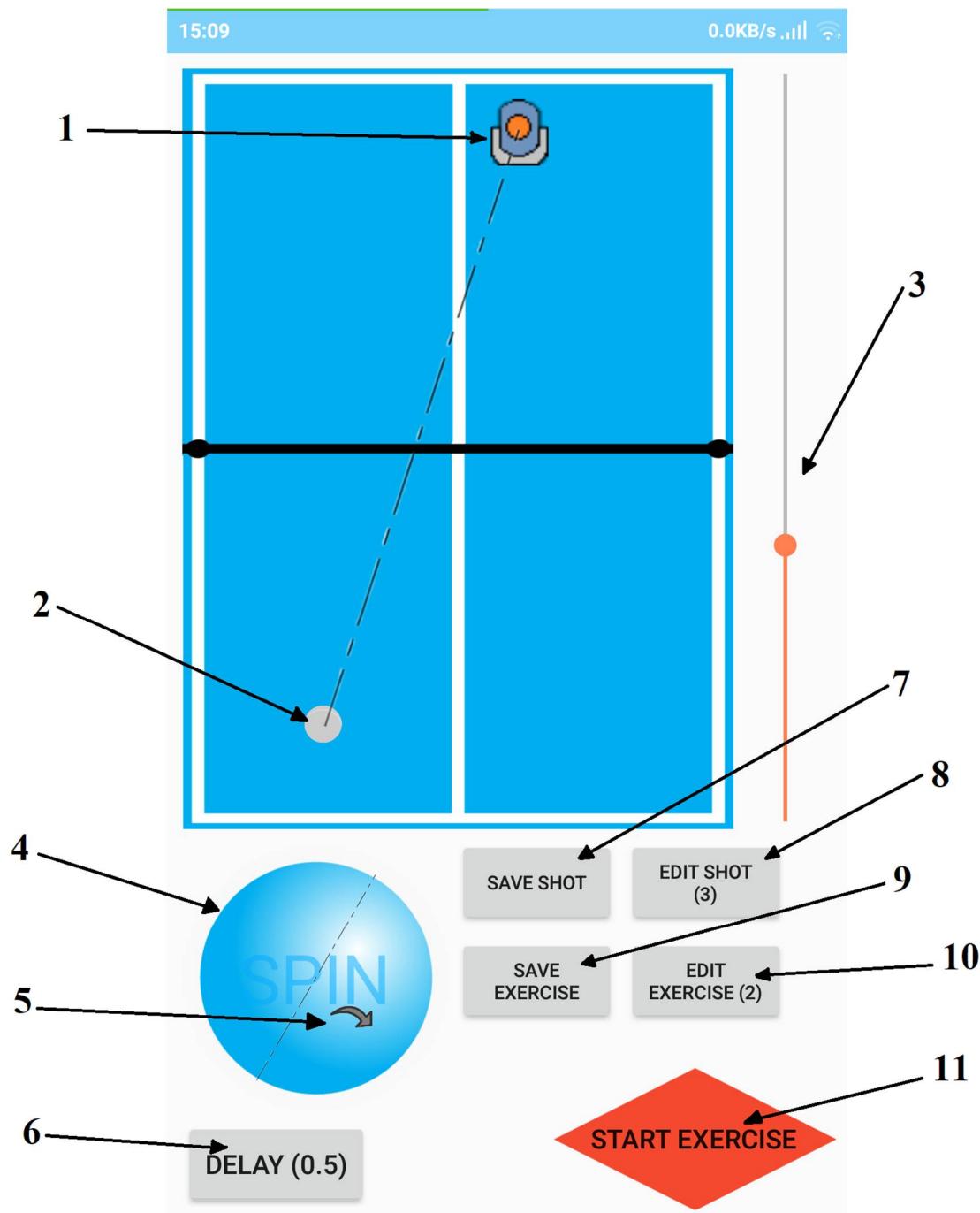
$$v_3 = v + 0,01732y_{rot} - 0,01z_{rot}. \quad (8.21)$$

## 8.6. Razvoj korisničko sučelje aplikacije za upravljanje uređajem

Nakon što smo dobili sve jednadžbe koje su nam potrebne za upravljanje svih motora, trebamo izraditi sučelje aplikacije za pametni telefon tako da bude intuitivno i što jednostavnije za korisnika.



Slika 8.11. Sučelje aplikacije za upravljanje uređajem



Slika 8.12. Sučelje aplikacije s označenim elementima

Nakon što se uređaj postavi na stol, na bilo koje mjesto na stolu, u aplikaciji pokazivač za položaj uređaja (1) postavimo na približni položaj kao što je i u stvarnosti. Zatim pokazivač za pozicioniranje loptice (2) postavimo na mjesto na koje želimo da uređaj ispuca lopticu te se pomoću klizača (3) podešava izlazna brzina ispučavanja loptice. Nakon toga u području za definiranje rotacije (4), pomoću pokazivača za rotaciju (5) postavljamo željenu rotaciju loptice. Rotacija je definirana tako da područje za definiranje rotacije (4) predstavlja lopticu koja gleda prema igraču, te će rotirati oko prikazane osi u smjeru pokazivača (5). Što je pokazivač

udaljeniji od osi rotacije to će rotacija loptice biti jača. Zatim pomoću tipke za definiranje frekventnosti ispučavanja loptice (6) određujemo koliko će vremena proći nakon što je ispučana prva loptica do ispučavanja druge. Na toj tipki broj u zagradi predstavlja to vrijeme u sekundama. Pritiskom na tipku za spremanje udaraca (7) spremamo sve parametre koje smo u prethodnim koracima definirali te ponavljamo isti postupak koliko god puta želimo. Pomoću tipke za uređivanje udaraca (8) možemo odabrati neki od udaraca koje smo spremili te uređivati ostale parametre. Broj u zagradi označava koliko smo udaraca spremili do sada. Nakon što su spremljeni svi udarci koje želimo imati u vježbi, pritiskom na tipku za spremanje vježbi (9), će se spremiti svi udarci koje smo definirali, kao jednu vježbu. Pritiskom na tipku *START EXERCISE* (11) u pozadini aplikacije će se, pomoću jednadžbi izvedenih u poglavljima 7.4 i 7.5, izračunati nagib i zakret uređaja te brzina rotacije triju motora za ispučavanje loptica. Dobivene vrijednosti će se preko *Bluetooth-a* poslati na mikrokontroler koji se nalazi na uređaju te će vježba započeti.

## 9. ZAKLJUČAK

U ovom radu opisana je konstrukcija uređaja za ispučavanje loptica za stolni tenis koji bi omogućio individualni trening sportaša. Analizom tržišta utvrđeno je da takvi uređaji već postoje te je cilj bio konstrukcija takvog uređaja koji je jeftiniji, jednostavniji za korištenje i ima više funkcija od postojećih.

Uređaj je konstruiran tako da se sastoji od 3 glavna dijela, kućišta, nosača kućišta i postolja. Kućište sadrži mehanizam za ispučavanje loptica, spremnik za loptice i mehanizam za dovođenje loptica do mehanizma za ispučavanje. U nosaču kućišta nalazi se mehanizam za nagib, a u postolju se nalazi mehanizam za zakret uređaja te upravljački mikrokontroler. Takvom izvedbom omogućena je rotacija loptice, snaga ispučavanja, promjenjivi domet i kut ispučavanja. Također u radu je izrađena aplikacija za pametni telefon pomoću koje se upravlja s uređajem te je moguće spremanje raznih parametara ispučavanja loptica. Pametni telefon povezuje se preko *Bluetooth-a* s uređajem te se tako ostvaruje komunikacija.

Prednosti u odnosu na postojeće proizvode su programiranje neograničenog broja vlastitih vježbi, moguća veća kontrola rotacije loptice kako je opisano u poglavlju 8.5, prenosivost zbog malih dimenzija, napajanja preko baterija i manja cijena uređaja. Mane u odnosu na postojeće proizvode su manji kapacitet loptica te obavezno posjedovanje pametnog telefona da bi korisnik mogao upravljati uređajem.

Kao daljnji razvoj ovog uređaja trebalo bi se malo više osvrnuti na funkcionalnost i jednostavnost mobilne aplikacije. Za vizualizaciju putanje loptice moglo bi se izraditi kao 3D model, ograničiti postavljanje parametara za koje nije moguće ostvariti ispučavanje loptice, te dodati povratnu informaciju iz uređaja koja će nam govoriti u kakvom je stanju uređaj, ima li loptica u spremniku ili ne, koliko loptica je ispučano, itd. Također u dalnjem razvoju bilo bi dobro da se poveća kapacitet loptica.

Svi plastični dijelovi uređaja bili bi izrađeni od ABS materijala injekcijskim prešanjem. Takva tehnologija izrade inicijalno je skupa zbog izrade alata, ali ako se radi o velikim serijama cijena izrade dijelova postaje jako mala. Stoga, ako bi se uređaj radio u jako velikim serijama, cijena bi se kretala od 750 do 1000kn što je u rangu s trenutno najjeftinijim uređajem na tržištu s tim da su mogućnosti ovog uređaja puno veće.

## LITERATURA

- [1] <https://www.allabouttabletennis.com/table-tennis-robot.html>
- [2] <http://www.ipong.net/joomla/>
- [3] <https://www.newgy.com/>
- [4] <https://en.butterfly.tt/>
- [5] <https://patents.google.com/patent/US4844458A/en?q=robot&q=table+tennis>
- [6] <https://patents.google.com/patent/US20070221187A1/en?q=Table+tennis+ball&q=serving&oq=Table+tennis+ball+serving>
- [7] <https://patents.google.com/patent/US3878827A/en?q=Table+tennis+ball&q=serving&oq=Table+tennis+ball+serving>
- [8] <http://www.jayandwanda.com/tt/speed.html>

## PRILOZI

- CD-R disc
- Tehnička dokumentacija
- Programski kod aplikacije za upravljanje

```
1 package com.example.tateapp;
2
3 import android.annotation.SuppressLint;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.MotionEvent;
7 import android.view.View;
8 import android.widget.AdapterView;
9 import android.widget.Button;
10 import android.widget.ImageView;
11 import android.widget.SeekBar;
12 import android.widget.Spinner;
13 import android.widget.TextView;
14
15 public class MainActivity extends AppCompatActivity {
16
17
18     TextView alphaTxt, phiTxt, v0Txt, v1Txt, v2Txt, v3Txt;
19
20
21     boolean enableDevice = true;
22
23     float v0;
24     float v1;
25     float v2;
26     float v3;
27     float phi;
28     double alpha;
29     float delay = 1000f;
30     float d = 1;
31     int deviceViewX = 1;
32     int deviceViewY = 1;
33     int ballViewX = 1;
34     int ballViewY = 1;
35     float deviceViewXmapped = 1;
36     float deviceViewYmapped = 1;
37     float ballViewXmapped = 1;
38     float ballViewYmapped = 1;
39     int spinPointerViewX = 0;
40     int spinPointerViewY = 0;
41     float xRot = 0;
42     float yRot = 0;
43     float sec = 1;
44
45     private static final float g = 9.81f;
46     private static final float h0 = 0.2f;
47     private static final int v0min = 10;
48     private static final int v0max = 25;
49     private static final int xu_min = -25;
50     private static final int xu_max = 25;
51     private static final int yu_min = -145;
52     private static final int yu_max = 145;
53
54     @SuppressLint({"ClickableViewAccessibility", "DefaultLocale"})
55     @Override
56     protected void onCreate(Bundle savedInstanceState) {
57         super.onCreate(savedInstanceState);
58         setContentView(R.layout.activity_main);
59
60         final View tableView = findViewById(R.id.tableView);
61         final ImageView deviceView = findViewById(R.id.deviceView);
62         final ImageView ballView = findViewById(R.id.ballView);
63         final View spinCanvasView = findViewById(R.id.spinCanvasView);
64         final ImageView spinAxisView = findViewById(R.id.spinAxisView);
65         final ImageView spinPointerView = findViewById(R.id.spinPointerView);
66         final SeekBar speedSeekBar = findViewById(R.id.speedSeekBar);
67         final Spinner delaySpinner = findViewById(R.id.delaySpinner);
68         final Button startBtn = findViewById(R.id.startBtn);
```

```

69
70     phiTxt = findViewById(R.id.phiTxt);
71     alphaTxt = findViewById(R.id.alphaTxt);
72     v0Txt = findViewById(R.id.v0Txt);
73     v1Txt = findViewById(R.id.v1Txt);
74     v2Txt = findViewById(R.id.v2Txt);
75     v3Txt = findViewById(R.id.v3Txt);
76
77     final BtSend bt = new BtSend();
78     bt.init();
79     bt.run(this, (rcString, thisss) ->
80             thisss.runOnUiThread(() ->
81                     startBtn.setText(rcString)));
82
83
84
85
86     tableView.setOnLongClickListener(event -> {
87         if (enableDevice) {
88             enableDevice = false;
89             deviceView.setBackgroundResource(R.drawable.device_nobg_gratty);
90         }
91         else {
92             enableDevice = true;
93             deviceView.setBackgroundResource(R.drawable.device_nobg);
94         }
95         return true;
96     });
97
98     tableView.setOnTouchListener((v, event)->{
99
100         deviceViewX = (int) deviceView.getX() + deviceView.getWidth()/2;
101         deviceViewY = (int) deviceView.getY() + deviceView.getHeight()/2;
102         ballViewX = (int) ballView.getX() + ballView.getWidth()/2;
103         ballViewY = (int) ballView.getY() + ballView.getHeight()/2;
104
105         v0 = (float) map(0, speedSeekBar.getMax(),
106                         v0min, v0max, speedSeekBar.getProgress());
107
108         spinPointerViewX = (int) spinPointerView.getX() +
109             spinPointerView.getWidth()/2;
110         spinPointerViewY = (int) spinPointerView.getY() +
111             spinPointerView.getHeight()/2;
112
113         xRot = (float) map(spinCanvasView.getY(), spinCanvasView.getY() +
114                             spinCanvasView.getHeight() - spinPointerView.getHeight(),
115                             xu_min, xu_max, spinPointerView.getY());
116         yRot = (float) map(spinCanvasView.getX(), spinCanvasView.getX() +
117                             spinCanvasView.getWidth() - spinPointerView.getWidth(),
118                             yu_min, yu_max, spinPointerView.getX());
119
120         v1 = v0 + 0.02f * yRot;
121         v2 = v0 - 0.0173f * xRot - 0.01f * yRot;
122         v3 = v0 + 0.0173f * xRot - 0.01f * yRot;
123
124         int tableViewXStart = (int) tableView.getX();
125         int tableViewXEnd = (int) tableView.getX() + tableView.getWidth();
126         int tableViewYStart = (int) tableView.getY();
127         int tableViewYEnd = (int) tableView.getY() + tableView.getHeight();
128
129         deviceViewX = (int) deviceView.getX() + deviceView.getWidth()/2;
130         deviceViewY = (int) deviceView.getY() + deviceView.getHeight()/2;
131         ballViewX = (int) ballView.getX() + ballView.getWidth()/2;
132         ballViewY = (int) ballView.getY() + ballView.getHeight()/2;
133
134         if(event.getX()>ballViewX-50 && event.getX()<ballViewX+50 &&
              event.getY()>ballViewY-50 && event.getY()<ballViewY+50) {

```

```

135         ballView.setX(event.getX()-ballView.getWidth()/2f);
136         ballView.setY(event.getY()-ballView.getHeight()/2f);
137
138         if (event.getX()<tableViewXStart+ballView.getWidth()/2) {
139             ballView.setX(tableViewXStart);
140         }
141         if (event.getX()>tableViewXEnd-ballView.getWidth()/2) {
142             ballView.setX(tableViewXEnd-ballView.getWidth());
143         }
144         if (event.getY()<tableViewYStart+ballView.getHeight()/2) {
145             ballView.setY(tableViewYStart);
146         }
147         if (event.getY()>tableViewYEnd-ballView.getHeight()/2) {
148             ballView.setY(tableViewYEnd-ballView.getHeight());
149         }
150     }
151
152     if(event.getX()>deviceViewX-50 && event.getX()<deviceViewX+50 &&
153         event.getY()>deviceViewY-50 && event.getY()<deviceViewY+50 &&
154         enableDevice) {
155         deviceView.setX(event.getX()-deviceView.getWidth()/2f);
156         deviceView.setY(event.getY()-deviceView.getHeight()/2f);
157
158         if (event.getX()<tableViewXStart+deviceView.getWidth()/2) {
159             deviceView.setX(tableViewXStart);
160         }
161         if (event.getX()>tableViewXEnd-deviceView.getWidth()/2) {
162             deviceView.setX(tableViewXEnd-deviceView.getWidth());
163         }
164         if (event.getY()<tableViewYStart+deviceView.getHeight()/2) {
165             deviceView.setY(tableViewYStart);
166         }
167         if (event.getY()>tableViewYEnd-deviceView.getHeight()/2) {
168             deviceView.setY(tableViewYEnd-deviceView.getHeight());
169         }
170     }
171
172     if(event.getAction()==MotionEvent.ACTION_UP) {
173         if(deviceViewX==ballViewX && deviceViewY==ballViewY) {
174             ballView.setY(ballView.getY()+300);
175         }
176     }
177
178     deviceViewXmapped = (float) map(tableView.getX(), tableView.getWidth(),
179                                     0f, 1.525, deviceViewX);
180     deviceViewYmapped = (float) map(tableView.getY(), tableView.getHeight(),
181                                     0f, 2.74, deviceViewY);
182     ballViewXmapped = (float) map(tableView.getX(), tableView.getWidth(),
183                                     0f, 1.525, ballViewX);
184     ballViewYmapped = (float) map(tableView.getY(), tableView.getHeight(),
185                                     0f, 2.74, ballViewY);
186
187     d = (float) Math.sqrt ((deviceViewXmapped-ballViewXmapped)*
188                           (deviceViewXmapped-ballViewXmapped)+(deviceViewYmapped
189                           -ballViewYmapped)*
190                           (deviceViewYmapped -ballViewYmapped));
191     phi = (float) Math.atan((ballViewXmapped-deviceViewXmapped)/
192                           (ballViewYmapped-deviceViewYmapped));
193     alpha =
194     2f*Math.atan((d-Math.sqrt(d*d+h0*h0-(g*d*d/(2f*v0))* (g*d*d/(2f*v0)))))/
195     (h0+(g*d*d/(2f*v0))));;
196
197     setAll();
198
199     return true;
}

```

```

200     spinCanvasView.setOnTouchListener((v, event) ->{
201
202         deviceViewX = (int) deviceView.getX() + deviceView.getWidth()/2;
203         deviceViewY = (int) deviceView.getY() + deviceView.getHeight()/2;
204         ballViewX = (int) ballView.getX() + ballView.getWidth()/2;
205         ballViewY = (int) ballView.getY() + ballView.getHeight()/2;
206
207         v0 = (float) map(0, speedSeekBar.getMax(),
208                         v0min, v0max, speedSeekBar.getProgress());
209
210         deviceViewXmapped = (float) map(tableView.getX(), tableView.getWidth(),
211                                         0f, 1.525, deviceViewX);
212         deviceViewYmapped = (float) map(tableView.getY(), tableView.getHeight(),
213                                         0f, 2.74, deviceViewY);
214         ballViewXmapped = (float) map(tableView.getX(), tableView.getWidth(),
215                                         0f, 1.525, ballViewX);
216         ballViewYmapped = (float) map(tableView.getY(), tableView.getHeight(),
217                                         0f, 2.74, ballViewY);
218         d = (float) Math.sqrt((deviceViewXmapped-ballViewXmapped)*
219                               (deviceViewXmapped-ballViewXmapped)+(deviceViewYmapped-
220                               ballViewYmapped)*
221                               (deviceViewYmapped-ballViewYmapped));
222         phi = (float) Math.atan((ballViewXmapped-deviceViewXmapped)/
223                               (ballViewYmapped-deviceViewYmapped));
224         alpha =
225             2f*Math.atan((d-Math.sqrt(d*d+h0*h0-(g*d*d/(2f*v0))* (g*d*d/(2f*v0))))/
226             (h0+(g*d*d/(2f*v0))));
```

spinPointerViewX = (int) spinPointerView.getX() +
 spinPointerView.getWidth()/2;
 spinPointerViewY = (int) spinPointerView.getY() +
 spinPointerView.getHeight()/2;

```

227
228         xRot = (float) map(spinCanvasView.getY(), spinCanvasView.getY() +
229                             spinCanvasView.getHeight() - spinPointerView.getHeight(),
230                             xu_min, xu_max, spinPointerView.getY());
231         yRot = (float) map(spinCanvasView.getX(), spinCanvasView.getX() +
232                             spinCanvasView.getWidth() - spinPointerView.getWidth(),
233                             yu_min, yu_max, spinPointerView.getX());
```

float a = event.getX() + spinCanvasView.getX();
 float b = event.getY() + spinCanvasView.getY();
 int r = spinCanvasView.getWidth()/2 - spinPointerView.getWidth()/2;
 float x00 = spinCanvasView.getX() + r;
 float y00 = spinCanvasView.getY() + r;

```

236
237         if (event.getX() > spinCanvasView.getWidth() -
238             spinPointerView.getWidth()) {
239             a = spinCanvasView.getX() + spinCanvasView.getWidth() -
240                 spinPointerView.getWidth();
241         }
242         if (event.getX() < 0) {
243             a = (int) spinCanvasView.getX();
244         }
```

double y1\_lim = Math.sqrt(Math.pow(r, 2)-Math.pow(a-x00, 2))+y00;
 double y2\_lim = -Math.sqrt(Math.pow(r, 2)-Math.pow(a-x00, 2))+y00;

```

245         if (y1_lim < b)
246             b = (float) y1_lim;
247         if (y2_lim > b)
248             b = (float) y2_lim;
```

spinPointerView.setX(a);
 spinPointerView.setY(b);

```

253
254         float x_lok = (float) map(spinCanvasView.getX(),
```

```

262     spinCanvasView.getWidth() +
263         spinCanvasView.getX()- spinPointerView.getWidth(),
264         -spinCanvasView.getWidth()/2f, spinCanvasView.getWidth()/2f, a);
265     float y_lok = (float) map(spinCanvasView.getY(),
266     spinCanvasView.getHeight() +
267         spinCanvasView.getX()- spinPointerView.getHeight(),
268         spinCanvasView.getHeight()/2f, spinCanvasView.getHeight()*2, b);
269     int axisRotDeg= (int) -(Math.atan2(y_lok, x_lok)*180/Math.PI);
270     spinAxisView.setRotation(axisRotDeg);
271     spinPointerView.setRotation(axisRotDeg);

272     v1 = v0 + 0.02f * yRot;
273     v2 = v0 - 0.0173f * xRot - 0.01f * yRot;
274     v3 = v0 + 0.0173f * xRot - 0.01f * yRot;

275     setAll();
276
277     return true;
278 });
279
280
281 speedSeekBar.setOnSeekBarChangeListener(new
282 SeekBar.OnSeekBarChangeListener() {
283     @Override
284     public void onProgressChanged(SeekBar seekBar, int progress, boolean
285     fromUser) {
286
287     }
288
289     @Override
290     public void onStartTrackingTouch(SeekBar seekBar) {
291
292     }
293
294     @Override
295     public void onStopTrackingTouch(SeekBar seekBar) {
296
297         deviceViewX = (int) deviceView.getX() + deviceView.getWidth()/2;
298         deviceViewY = (int) deviceView.getY() + deviceView.getHeight()/2;
299         ballViewX = (int) ballView.getX() + ballView.getWidth()/2;
300         ballViewY = (int) ballView.getY() + ballView.getHeight()/2;
301
302         v0 = (float) map(0, speedSeekBar.getMax(),
303             v0min, v0max, speedSeekBar.getProgress());
304
305         deviceViewXmapped = (float) map(tableView.getX(),
306             tableView.getWidth(),
307                 0f, 1.525, deviceViewX);
308         deviceViewYmapped = (float) map(tableView.getY(),
309             tableView.getHeight(),
310                 0f, 2.74, deviceViewY);
311         ballViewXmapped = (float) map(tableView.getX(), tableView.getWidth(),
312             0f, 1.525, ballViewX);
313         ballViewYmapped = (float) map(tableView.getY(), tableView.getHeight(),
314             0f, 2.74, ballViewY);
315         d = (float) Math.sqrt((deviceViewXmapped-ballViewXmapped)*
316             (deviceViewXmapped-ballViewXmapped)+(deviceViewYmapped-
317             ballViewYmapped)*
318             (deviceViewYmapped -ballViewYmapped));
319         phi = (float) Math.atan((ballViewXmapped-deviceViewXmapped)/
320             (ballViewYmapped-deviceViewYmapped));
321         alpha =
322             2f*Math.atan((d-Math.sqrt(d*d+h0*h0-(g*d*d/(2f*v0)))*(g*d*d/(2f*v0))))/
323             (h0+(g*d*d/(2f*v0))));
324
325         spinPointerViewX = (int) spinPointerView.getX() +
326             spinPointerView.getWidth()/2;
327         spinPointerViewY = (int) spinPointerView.getY() +

```

## MainActivity.java

```

321     spinPointerView.getHeight() / 2;
322
323     xRot = (float) map(spinCanvasView.getY(), spinCanvasView.getY() +
324                         spinCanvasView.getHeight() - spinPointerView.getHeight(),
325                         xu_min, xu_max, spinPointerView.getY());
326     yRot = (float) map(spinCanvasView.getX(), spinCanvasView.getX() +
327                         spinCanvasView.getWidth() - spinPointerView.getWidth(),
328                         yu_min, yu_max, spinPointerView.getX());
329
330     v1 = v0 + 0.02f * yRot;
331     v2 = v0 - 0.0173f * xRot - 0.01f * yRot;
332     v3 = v0 + 0.0173f * xRot - 0.01f * yRot;
333
334     setAll();
335 }
336
337 delaySpinner.setOnItemSelectedListener(new
338 AdapterView.OnItemSelectedListener() {
339     @Override
340     public void onItemSelected(AdapterView<?> parent, View view, int
341 position, long id) {
342         String delayS = (String) delaySpinner.getSelectedItem();
343         delayS = delayS.replace("DELAY (", " ");
344         delayS = delayS.replace(")", " ");
345         delay = Float.parseFloat(delayS)*1000;
346         setAll();
347     }
348
349     @Override
350     public void onNothingSelected(AdapterView<?> parent) {
351     }
352 }
353
354 startBtn.setOnClickListener(v -> {
355     bt.write(String.format("Hello %.2f, %.2f, %.2f, %.2f, %.2f, %.2f,
356     %.2f\n",
357             v1, v2, v3, phi, alpha, delay));
358 }
359
360 }
361
362 public void setAll(){
363     phiTxt.setText(String.format("%s", toText(phi,2)));
364     alphaTxt.setText(String.format("%s", toText(alpha,2)));
365     v0Txt.setText(String.format("%s", toText(v0,2)));
366     v1Txt.setText(String.format("%s", toText(d*d+h0*h0-sec*sec,2)));
367     v2Txt.setText(String.format("%s", toText(v2,2)));
368     v3Txt.setText(String.format("%s", toText(v3,2)));
369 }
370
371 public double map(double xmin, double xmax, double xbmin, double xbmax, double
372 xa){
373     double xb;
374     xb=(xbmax-xbmin)*(xa-xamin)/(xmax-xamin)+xbmin;
375     return xb;
376 }
377
378 public double toText (double xu, int d){
379     double xi;
380     xi=((double) ((int) (xu*Math.pow(10, d)))/Math.pow(10, d));
381     return xi;
382 }
383 }
```

```

1 package com.example.tateapp;
2
3 import android.bluetooth.BluetoothAdapter;
4 import android.bluetooth.BluetoothDevice;
5 import android.bluetooth.BluetoothSocket;
6 import android.os.ParcelUuid;
7 import android.util.Log;
8 import java.io.BufferedReader;
9 import java.io.InputStream;
10 import java.io.InputStreamReader;
11 import java.io.OutputStream;
12 import java.io.OutputStreamWriter;
13 import java.util.Iterator;
14 import java.util.Set;
15
16 public class BtSend {
17
18     private OutputStream outputStream;
19     private InputStream inStream;
20     private OutputStreamWriter outWrite;
21     private BufferedReader inWriter;
22
23     public void init() {
24         BluetoothAdapter blueAdapter = BluetoothAdapter.getDefaultAdapter();
25         if (blueAdapter != null) {
26             if (blueAdapter.isEnabled()) {
27                 Set<BluetoothDevice> bondedDevices = blueAdapter.getBondedDevices();
28
29                 if (bondedDevices.size() > 0) {
30
31                     BluetoothDevice device = null;
32
33                     for (Iterator<BluetoothDevice> it = bondedDevices.iterator();
34                         it.hasNext(); ) {
35                         BluetoothDevice d = it.next();
36                         if (d.getAddress().equals("24:0A:C4:06:DD:32"))
37                             device = d;
38                     }
39                     try {
40                         ParcelUuid[] uuids = device.getUuids();
41                         BluetoothSocket socket =
42
43                             device.createRfcommSocketToServiceRecord(uuids[0].getU
44                             uid());
45                         socket.connect();
46                         outputStream = socket.getOutputStream();
47                         inStream = socket.getInputStream();
48                         inWriter = new BufferedReader(new
49                             InputStreamReader(inStream));
50                         outWrite = (new OutputStreamWriter(outputStream));
51
52                     } catch (Exception e) {
53                         Log.e("error", "No appropriate paired devices.");
54                     } else {
55                         Log.e("error", "Bluetooth is disabled.");
56                     }
57                 }
58             }
59             public void write(String msg) {
60                 try {
61                     outWrite.write(msg);
62                     outWrite.flush();
63                 } catch (Exception e) {
64                     e.printStackTrace();
65                 }
66             }
67         }
68     }
69
70     public void close() {
71         try {
72             if (inWriter != null)
73                 inWriter.close();
74             if (outWrite != null)
75                 outWrite.close();
76             if (socket != null)
77                 socket.close();
78         } catch (IOException e) {
79             e.printStackTrace();
80         }
81     }
82
83     public void read() {
84         try {
85             String line;
86             while ((line = inWriter.readLine()) != null)
87                 Log.d("read", line);
88         } catch (IOException e) {
89             e.printStackTrace();
90         }
91     }
92
93     public void write(String msg) {
94         try {
95             outWrite.write(msg);
96             outWrite.flush();
97         } catch (IOException e) {
98             e.printStackTrace();
99         }
100    }
101
102    public void close() {
103        try {
104            if (inWriter != null)
105                inWriter.close();
106            if (outWrite != null)
107                outWrite.close();
108            if (socket != null)
109                socket.close();
110        } catch (IOException e) {
111            e.printStackTrace();
112        }
113    }
114
115    public void read() {
116        try {
117            String line;
118            while ((line = inWriter.readLine()) != null)
119                Log.d("read", line);
120        } catch (IOException e) {
121            e.printStackTrace();
122        }
123    }
124
125    public void write(String msg) {
126        try {
127            outWrite.write(msg);
128            outWrite.flush();
129        } catch (IOException e) {
130            e.printStackTrace();
131        }
132    }
133
134    public void close() {
135        try {
136            if (inWriter != null)
137                inWriter.close();
138            if (outWrite != null)
139                outWrite.close();
140            if (socket != null)
141                socket.close();
142        } catch (IOException e) {
143            e.printStackTrace();
144        }
145    }
146
147    public void read() {
148        try {
149            String line;
150            while ((line = inWriter.readLine()) != null)
151                Log.d("read", line);
152        } catch (IOException e) {
153            e.printStackTrace();
154        }
155    }
156
157    public void write(String msg) {
158        try {
159            outWrite.write(msg);
160            outWrite.flush();
161        } catch (IOException e) {
162            e.printStackTrace();
163        }
164    }
165
166    public void close() {
167        try {
168            if (inWriter != null)
169                inWriter.close();
170            if (outWrite != null)
171                outWrite.close();
172            if (socket != null)
173                socket.close();
174        } catch (IOException e) {
175            e.printStackTrace();
176        }
177    }
178
179    public void read() {
180        try {
181            String line;
182            while ((line = inWriter.readLine()) != null)
183                Log.d("read", line);
184        } catch (IOException e) {
185            e.printStackTrace();
186        }
187    }
188
189    public void write(String msg) {
190        try {
191            outWrite.write(msg);
192            outWrite.flush();
193        } catch (IOException e) {
194            e.printStackTrace();
195        }
196    }
197
198    public void close() {
199        try {
200            if (inWriter != null)
201                inWriter.close();
202            if (outWrite != null)
203                outWrite.close();
204            if (socket != null)
205                socket.close();
206        } catch (IOException e) {
207            e.printStackTrace();
208        }
209    }
210
211    public void read() {
212        try {
213            String line;
214            while ((line = inWriter.readLine()) != null)
215                Log.d("read", line);
216        } catch (IOException e) {
217            e.printStackTrace();
218        }
219    }
220
221    public void write(String msg) {
222        try {
223            outWrite.write(msg);
224            outWrite.flush();
225        } catch (IOException e) {
226            e.printStackTrace();
227        }
228    }
229
230    public void close() {
231        try {
232            if (inWriter != null)
233                inWriter.close();
234            if (outWrite != null)
235                outWrite.close();
236            if (socket != null)
237                socket.close();
238        } catch (IOException e) {
239            e.printStackTrace();
240        }
241    }
242
243    public void read() {
244        try {
245            String line;
246            while ((line = inWriter.readLine()) != null)
247                Log.d("read", line);
248        } catch (IOException e) {
249            e.printStackTrace();
250        }
251    }
252
253    public void write(String msg) {
254        try {
255            outWrite.write(msg);
256            outWrite.flush();
257        } catch (IOException e) {
258            e.printStackTrace();
259        }
260    }
261
262    public void close() {
263        try {
264            if (inWriter != null)
265                inWriter.close();
266            if (outWrite != null)
267                outWrite.close();
268            if (socket != null)
269                socket.close();
270        } catch (IOException e) {
271            e.printStackTrace();
272        }
273    }
274
275    public void read() {
276        try {
277            String line;
278            while ((line = inWriter.readLine()) != null)
279                Log.d("read", line);
280        } catch (IOException e) {
281            e.printStackTrace();
282        }
283    }
284
285    public void write(String msg) {
286        try {
287            outWrite.write(msg);
288            outWrite.flush();
289        } catch (IOException e) {
290            e.printStackTrace();
291        }
292    }
293
294    public void close() {
295        try {
296            if (inWriter != null)
297                inWriter.close();
298            if (outWrite != null)
299                outWrite.close();
300            if (socket != null)
301                socket.close();
302        } catch (IOException e) {
303            e.printStackTrace();
304        }
305    }
306
307    public void read() {
308        try {
309            String line;
310            while ((line = inWriter.readLine()) != null)
311                Log.d("read", line);
312        } catch (IOException e) {
313            e.printStackTrace();
314        }
315    }
316
317    public void write(String msg) {
318        try {
319            outWrite.write(msg);
320            outWrite.flush();
321        } catch (IOException e) {
322            e.printStackTrace();
323        }
324    }
325
326    public void close() {
327        try {
328            if (inWriter != null)
329                inWriter.close();
330            if (outWrite != null)
331                outWrite.close();
332            if (socket != null)
333                socket.close();
334        } catch (IOException e) {
335            e.printStackTrace();
336        }
337    }
338
339    public void read() {
340        try {
341            String line;
342            while ((line = inWriter.readLine()) != null)
343                Log.d("read", line);
344        } catch (IOException e) {
345            e.printStackTrace();
346        }
347    }
348
349    public void write(String msg) {
350        try {
351            outWrite.write(msg);
352            outWrite.flush();
353        } catch (IOException e) {
354            e.printStackTrace();
355        }
356    }
357
358    public void close() {
359        try {
360            if (inWriter != null)
361                inWriter.close();
362            if (outWrite != null)
363                outWrite.close();
364            if (socket != null)
365                socket.close();
366        } catch (IOException e) {
367            e.printStackTrace();
368        }
369    }
370
371    public void read() {
372        try {
373            String line;
374            while ((line = inWriter.readLine()) != null)
375                Log.d("read", line);
376        } catch (IOException e) {
377            e.printStackTrace();
378        }
379    }
380
381    public void write(String msg) {
382        try {
383            outWrite.write(msg);
384            outWrite.flush();
385        } catch (IOException e) {
386            e.printStackTrace();
387        }
388    }
389
390    public void close() {
391        try {
392            if (inWriter != null)
393                inWriter.close();
394            if (outWrite != null)
395                outWrite.close();
396            if (socket != null)
397                socket.close();
398        } catch (IOException e) {
399            e.printStackTrace();
400        }
401    }
402
403    public void read() {
404        try {
405            String line;
406            while ((line = inWriter.readLine()) != null)
407                Log.d("read", line);
408        } catch (IOException e) {
409            e.printStackTrace();
410        }
411    }
412
413    public void write(String msg) {
414        try {
415            outWrite.write(msg);
416            outWrite.flush();
417        } catch (IOException e) {
418            e.printStackTrace();
419        }
420    }
421
422    public void close() {
423        try {
424            if (inWriter != null)
425                inWriter.close();
426            if (outWrite != null)
427                outWrite.close();
428            if (socket != null)
429                socket.close();
430        } catch (IOException e) {
431            e.printStackTrace();
432        }
433    }
434
435    public void read() {
436        try {
437            String line;
438            while ((line = inWriter.readLine()) != null)
439                Log.d("read", line);
440        } catch (IOException e) {
441            e.printStackTrace();
442        }
443    }
444
445    public void write(String msg) {
446        try {
447            outWrite.write(msg);
448            outWrite.flush();
449        } catch (IOException e) {
450            e.printStackTrace();
451        }
452    }
453
454    public void close() {
455        try {
456            if (inWriter != null)
457                inWriter.close();
458            if (outWrite != null)
459                outWrite.close();
460            if (socket != null)
461                socket.close();
462        } catch (IOException e) {
463            e.printStackTrace();
464        }
465    }
466
467    public void read() {
468        try {
469            String line;
470            while ((line = inWriter.readLine()) != null)
471                Log.d("read", line);
472        } catch (IOException e) {
473            e.printStackTrace();
474        }
475    }
476
477    public void write(String msg) {
478        try {
479            outWrite.write(msg);
480            outWrite.flush();
481        } catch (IOException e) {
482            e.printStackTrace();
483        }
484    }
485
486    public void close() {
487        try {
488            if (inWriter != null)
489                inWriter.close();
490            if (outWrite != null)
491                outWrite.close();
492            if (socket != null)
493                socket.close();
494        } catch (IOException e) {
495            e.printStackTrace();
496        }
497    }
498
499    public void read() {
500        try {
501            String line;
502            while ((line = inWriter.readLine()) != null)
503                Log.d("read", line);
504        } catch (IOException e) {
505            e.printStackTrace();
506        }
507    }
508
509    public void write(String msg) {
510        try {
511            outWrite.write(msg);
512            outWrite.flush();
513        } catch (IOException e) {
514            e.printStackTrace();
515        }
516    }
517
518    public void close() {
519        try {
520            if (inWriter != null)
521                inWriter.close();
522            if (outWrite != null)
523                outWrite.close();
524            if (socket != null)
525                socket.close();
526        } catch (IOException e) {
527            e.printStackTrace();
528        }
529    }
530
531    public void read() {
532        try {
533            String line;
534            while ((line = inWriter.readLine()) != null)
535                Log.d("read", line);
536        } catch (IOException e) {
537            e.printStackTrace();
538        }
539    }
540
541    public void write(String msg) {
542        try {
543            outWrite.write(msg);
544            outWrite.flush();
545        } catch (IOException e) {
546            e.printStackTrace();
547        }
548    }
549
550    public void close() {
551        try {
552            if (inWriter != null)
553                inWriter.close();
554            if (outWrite != null)
555                outWrite.close();
556            if (socket != null)
557                socket.close();
558        } catch (IOException e) {
559            e.printStackTrace();
560        }
561    }
562
563    public void read() {
564        try {
565            String line;
566            while ((line = inWriter.readLine()) != null)
567                Log.d("read", line);
568        } catch (IOException e) {
569            e.printStackTrace();
570        }
571    }
572
573    public void write(String msg) {
574        try {
575            outWrite.write(msg);
576            outWrite.flush();
577        } catch (IOException e) {
578            e.printStackTrace();
579        }
580    }
581
582    public void close() {
583        try {
584            if (inWriter != null)
585                inWriter.close();
586            if (outWrite != null)
587                outWrite.close();
588            if (socket != null)
589                socket.close();
590        } catch (IOException e) {
591            e.printStackTrace();
592        }
593    }
594
595    public void read() {
596        try {
597            String line;
598            while ((line = inWriter.readLine()) != null)
599                Log.d("read", line);
600        } catch (IOException e) {
601            e.printStackTrace();
602        }
603    }
604
605    public void write(String msg) {
606        try {
607            outWrite.write(msg);
608            outWrite.flush();
609        } catch (IOException e) {
610            e.printStackTrace();
611        }
612    }
613
614    public void close() {
615        try {
616            if (inWriter != null)
617                inWriter.close();
618            if (outWrite != null)
619                outWrite.close();
620            if (socket != null)
621                socket.close();
622        } catch (IOException e) {
623            e.printStackTrace();
624        }
625    }
626
627    public void read() {
628        try {
629            String line;
630            while ((line = inWriter.readLine()) != null)
631                Log.d("read", line);
632        } catch (IOException e) {
633            e.printStackTrace();
634        }
635    }
636
637    public void write(String msg) {
638        try {
639            outWrite.write(msg);
640            outWrite.flush();
641        } catch (IOException e) {
642            e.printStackTrace();
643        }
644    }
645
646    public void close() {
647        try {
648            if (inWriter != null)
649                inWriter.close();
650            if (outWrite != null)
651                outWrite.close();
652            if (socket != null)
653                socket.close();
654        } catch (IOException e) {
655            e.printStackTrace();
656        }
657    }
658
659    public void read() {
660        try {
661            String line;
662            while ((line = inWriter.readLine()) != null)
663                Log.d("read", line);
664        } catch (IOException e) {
665            e.printStackTrace();
666        }
667    }
668
669    public void write(String msg) {
670        try {
671            outWrite.write(msg);
672            outWrite.flush();
673        } catch (IOException e) {
674            e.printStackTrace();
675        }
676    }
677
678    public void close() {
679        try {
680            if (inWriter != null)
681                inWriter.close();
682            if (outWrite != null)
683                outWrite.close();
684            if (socket != null)
685                socket.close();
686        } catch (IOException e) {
687            e.printStackTrace();
688        }
689    }
690
691    public void read() {
692        try {
693            String line;
694            while ((line = inWriter.readLine()) != null)
695                Log.d("read", line);
696        } catch (IOException e) {
697            e.printStackTrace();
698        }
699    }
700
701    public void write(String msg) {
702        try {
703            outWrite.write(msg);
704            outWrite.flush();
705        } catch (IOException e) {
706            e.printStackTrace();
707        }
708    }
709
710    public void close() {
711        try {
712            if (inWriter != null)
713                inWriter.close();
714            if (outWrite != null)
715                outWrite.close();
716            if (socket != null)
717                socket.close();
718        } catch (IOException e) {
719            e.printStackTrace();
720        }
721    }
722
723    public void read() {
724        try {
725            String line;
726            while ((line = inWriter.readLine()) != null)
727                Log.d("read", line);
728        } catch (IOException e) {
729            e.printStackTrace();
730        }
731    }
732
733    public void write(String msg) {
734        try {
735            outWrite.write(msg);
736            outWrite.flush();
737        } catch (IOException e) {
738            e.printStackTrace();
739        }
740    }
741
742    public void close() {
743        try {
744            if (inWriter != null)
745                inWriter.close();
746            if (outWrite != null)
747                outWrite.close();
748            if (socket != null)
749                socket.close();
750        } catch (IOException e) {
751            e.printStackTrace();
752        }
753    }
754
755    public void read() {
756        try {
757            String line;
758            while ((line = inWriter.readLine()) != null)
759                Log.d("read", line);
760        } catch (IOException e) {
761            e.printStackTrace();
762        }
763    }
764
765    public void write(String msg) {
766        try {
767            outWrite.write(msg);
768            outWrite.flush();
769        } catch (IOException e) {
770            e.printStackTrace();
771        }
772    }
773
774    public void close() {
775        try {
776            if (inWriter != null)
777                inWriter.close();
778            if (outWrite != null)
779                outWrite.close();
780            if (socket != null)
781                socket.close();
782        } catch (IOException e) {
783            e.printStackTrace();
784        }
785    }
786
787    public void read() {
788        try {
789            String line;
790            while ((line = inWriter.readLine()) != null)
791                Log.d("read", line);
792        } catch (IOException e) {
793            e.printStackTrace();
794        }
795    }
796
797    public void write(String msg) {
798        try {
799            outWrite.write(msg);
800            outWrite.flush();
801        } catch (IOException e) {
802            e.printStackTrace();
803        }
804    }
805
806    public void close() {
807        try {
808            if (inWriter != null)
809                inWriter.close();
810            if (outWrite != null)
811                outWrite.close();
812            if (socket != null)
813                socket.close();
814        } catch (IOException e) {
815            e.printStackTrace();
816        }
817    }
818
819    public void read() {
820        try {
821            String line;
822            while ((line = inWriter.readLine()) != null)
823                Log.d("read", line);
824        } catch (IOException e) {
825            e.printStackTrace();
826        }
827    }
828
829    public void write(String msg) {
830        try {
831            outWrite.write(msg);
832            outWrite.flush();
833        } catch (IOException e) {
834            e.printStackTrace();
835        }
836    }
837
838    public void close() {
839        try {
840            if (inWriter != null)
841                inWriter.close();
842            if (outWrite != null)
843                outWrite.close();
844            if (socket != null)
845                socket.close();
846        } catch (IOException e) {
847            e.printStackTrace();
848        }
849    }
850
851    public void read() {
852        try {
853            String line;
854            while ((line = inWriter.readLine()) != null)
855                Log.d("read", line);
856        } catch (IOException e) {
857            e.printStackTrace();
858        }
859    }
860
861    public void write(String msg) {
862        try {
863            outWrite.write(msg);
864            outWrite.flush();
865        } catch (IOException e) {
866            e.printStackTrace();
867        }
868    }
869
870    public void close() {
871        try {
872            if (inWriter != null)
873                inWriter.close();
874            if (outWrite != null)
875                outWrite.close();
876            if (socket != null)
877                socket.close();
878        } catch (IOException e) {
879            e.printStackTrace();
880        }
881    }
882
883    public void read() {
884        try {
885            String line;
886            while ((line = inWriter.readLine()) != null)
887                Log.d("read", line);
888        } catch (IOException e) {
889            e.printStackTrace();
890        }
891    }
892
893    public void write(String msg) {
894        try {
895            outWrite.write(msg);
896            outWrite.flush();
897        } catch (IOException e) {
898            e.printStackTrace();
899        }
900    }
901
902    public void close() {
903        try {
904            if (inWriter != null)
905                inWriter.close();
906            if (outWrite != null)
907                outWrite.close();
908            if (socket != null)
909                socket.close();
910        } catch (IOException e) {
911            e.printStackTrace();
912        }
913    }
914
915    public void read() {
916        try {
917            String line;
918            while ((line = inWriter.readLine()) != null)
919                Log.d("read", line);
920        } catch (IOException e) {
921            e.printStackTrace();
922        }
923    }
924
925    public void write(String msg) {
926        try {
927            outWrite.write(msg);
928            outWrite.flush();
929        } catch (IOException e) {
930            e.printStackTrace();
931        }
932    }
933
934    public void close() {
935        try {
936            if (inWriter != null)
937                inWriter.close();
938            if (outWrite != null)
939                outWrite.close();
940            if (socket != null)
941                socket.close();
942        } catch (IOException e) {
943            e.printStackTrace();
944        }
945    }
946
947    public void read() {
948        try {
949            String line;
950            while ((line = inWriter.readLine()) != null)
951                Log.d("read", line);
952        } catch (IOException e) {
953            e.printStackTrace();
954        }
955    }
956
957    public void write(String msg) {
958        try {
959            outWrite.write(msg);
960            outWrite.flush();
961        } catch (IOException e) {
962            e.printStackTrace();
963        }
964    }
965
966    public void close() {
967        try {
968            if (inWriter != null)
969                inWriter.close();
970            if (outWrite != null)
971                outWrite.close();
972            if (socket != null)
973                socket.close();
974        } catch (IOException e) {
975            e.printStackTrace();
976        }
977    }
978
979    public void read() {
980        try {
981            String line;
982            while ((line = inWriter.readLine()) != null)
983                Log.d("read", line);
984        } catch (IOException e) {
985            e.printStackTrace();
986        }
987    }
988
989    public void write(String msg) {
990        try {
991            outWrite.write(msg);
992            outWrite.flush();
993        } catch (IOException e) {
994            e.printStackTrace();
995        }
996    }
997
998    public void close() {
999        try {
1000            if (inWriter != null)
1001                inWriter.close();
1002            if (outWrite != null)
1003                outWrite.close();
1004            if (socket != null)
1005                socket.close();
1006        } catch (IOException e) {
1007            e.printStackTrace();
1008        }
1009    }
1010
1011    public void read() {
1012        try {
1013            String line;
1014            while ((line = inWriter.readLine()) != null)
1015                Log.d("read", line);
1016        } catch (IOException e) {
1017            e.printStackTrace();
1018        }
1019    }
1020
1021    public void write(String msg) {
1022        try {
1023            outWrite.write(msg);
1024            outWrite.flush();
1025        } catch (IOException e) {
1026            e.printStackTrace();
1027        }
1028    }
1029
1030    public void close() {
1031        try {
1032            if (inWriter != null)
1033                inWriter.close();
1034            if (outWrite != null)
1035                outWrite.close();
1036            if (socket != null)
1037                socket.close();
1038        } catch (IOException e) {
1039            e.printStackTrace();
1040        }
1041    }
1042
1043    public void read() {
1044        try {
1045            String line;
1046            while ((line = inWriter.readLine()) != null)
1047                Log.d("read", line);
1048        } catch (IOException e) {
1049            e.printStackTrace();
1050        }
1051    }
1052
1053    public void write(String msg) {
1054        try {
1055            outWrite.write(msg);
1056            outWrite.flush();
1057        } catch (IOException e) {
1058            e.printStackTrace();
1059        }
1060    }
1061
1062    public void close() {
1063        try {
1064            if (inWriter != null)
1065                inWriter.close();
1066            if (outWrite != null)
1067                outWrite.close();
1068            if (socket != null)
1069                socket.close();
1070        } catch (IOException e) {
1071            e.printStackTrace();
1072        }
1073    }
1074
1075    public void read() {
1076        try {
1077            String line;
1078            while ((line = inWriter.readLine()) != null)
1079                Log.d("read", line);
1080        } catch (IOException e) {
1081            e.printStackTrace();
1082        }
1083    }
1084
1085    public void write(String msg) {
1086        try {
1087            outWrite.write(msg);
1088            outWrite.flush();
1089        } catch (IOException e) {
1090            e.printStackTrace();
1091        }
1092    }
1093
1094    public void close() {
1095        try {
1096            if (inWriter != null)
1097                inWriter.close();
1098            if (outWrite != null)
1099                outWrite.close();
1100            if (socket != null)
1101                socket.close();
1102        } catch (IOException e) {
1103            e.printStackTrace();
1104        }
1105    }
1106
1107    public void read() {
1108        try {
1109            String line;
1110            while ((line = inWriter.readLine()) != null)
1111                Log.d("read", line);
1112        } catch (IOException e) {
1113            e.printStackTrace();
1114        }
1115    }
1116
1117    public void write(String msg) {
1118        try {
1119            outWrite.write(msg);
1120            outWrite.flush();
1121        } catch (IOException e) {
1122            e.printStackTrace();
1123        }
1124    }
1125
1126    public void close() {
1127        try {
1128            if (inWriter != null)
1129                inWriter.close();
1130            if (outWrite != null)
1131                outWrite.close();
1132            if (socket != null)
1133                socket.close();
1134        } catch (IOException e) {
1135            e.printStackTrace();
1136        }
1137    }
1138
1139    public void read() {
1140        try {
1141            String line;
1142            while ((line = inWriter.readLine()) != null)
1143                Log.d("read", line);
1144        } catch (IOException e) {
1145            e.printStackTrace();
1146        }
1147    }
1148
1149    public void write(String msg) {
1150        try {
1151            outWrite.write(msg);
1152            outWrite.flush();
1153        } catch (IOException e) {
1154            e.printStackTrace();
1155        }
1156    }
1157
1158    public void close() {
1159        try {
1160            if (inWriter != null)
1161                inWriter.close();
1162            if (outWrite != null)
1163                outWrite.close();
1164            if (socket != null)
1165                socket.close();
1166        } catch (IOException e) {
1167            e.printStackTrace();
1168        }
1169    }
1170
1171    public void read() {
1172        try {
1173            String line;
1174            while ((line = inWriter.readLine()) != null)
1175                Log.d("read", line);
1176        } catch (IOException e) {
1177            e.printStackTrace();
1178        }
1179    }
1180
1181    public void write(String msg) {
1182        try {
1183            outWrite.write(msg);
1184            outWrite.flush();
1185        } catch (IOException e) {
1186            e.printStackTrace();
1187        }
1188    }
1189
1190    public void close() {
1191        try {
1192            if (inWriter != null)
1193                inWriter.close();
1194            if (outWrite != null)
1195                outWrite.close();
1196            if (socket != null)
1197                socket.close();
1198        } catch (IOException e) {
1199            e.printStackTrace();
1200        }
1201    }
1202
1203    public void read() {
1204        try {
1205            String line;
1206            while ((line = inWriter.readLine()) != null)
1207                Log.d("read", line);
1208        } catch (IOException e) {
1209            e.printStackTrace();
1210        }
1211    }
1212
1213    public void write(String msg) {
1214        try {
1215            outWrite.write(msg);
1216            outWrite.flush();
1217        } catch (IOException e) {
1218            e.printStackTrace();
1219        }
1220    }
1221
1222    public void close() {
1223        try {
1224            if (inWriter != null)
1225                inWriter.close();
1226            if (outWrite != null)
1227                outWrite.close();
1228            if (socket != null)
1229                socket.close();
1230        } catch (IOException e) {
1231            e.printStackTrace();
1232        }
1233    }
1234
1235    public void read() {
1236        try {
1237            String line;
1238            while ((line = inWriter.readLine()) != null)
1239                Log.d("read", line);
1240        } catch (IOException e) {
1241            e.printStackTrace();
1242        }
1243    }
1244
1245    public void write(String msg) {
1246        try {
1247            outWrite.write(msg);
1248            outWrite.flush();
1249        } catch (IOException e) {
1250            e.printStackTrace();
1251        }
1252    }
1253
1254    public void close() {
1255        try {
1256            if (inWriter != null)
1257                inWriter.close();
1258            if (outWrite != null)
1259                outWrite.close();
1260            if (socket != null)
1261                socket.close();
1262        } catch (IOException e) {
1263            e.printStackTrace();
1264        }
1265    }
1266
1267    public void read() {
1268        try {
1269            String line;
1270            while ((line = inWriter.readLine()) != null)
1271                Log.d("read", line);
1272        } catch (IOException e) {
1273            e.printStackTrace();
1274        }
1275    }
1276
1277    public void write(String msg) {
1278        try {
1279            outWrite.write(msg);
1280            outWrite.flush();
1281        } catch (IOException e) {
1282            e.printStackTrace();
1283        }
1284    }
1285
1286    public void close() {
1287        try {
1288            if (inWriter != null)
1289                inWriter.close();
1290            if (outWrite != null)
1291                outWrite.close();
1292            if (socket != null)
1293                socket.close();
1294        } catch (IOException e) {
1295            e.printStackTrace();
1296        }
1297    }
1298
1299    public void read() {
1300        try {
1301            String line;
1302            while ((line = inWriter.readLine()) != null)
1303                Log.d("read", line);
1304        } catch (IOException e) {
1305            e.printStackTrace();
1306        }
1307    }
1308
1309    public void write(String msg) {
1310        try {
1311            outWrite.write(msg);
1312            outWrite.flush();
1313        } catch (IOException e) {
1314            e.printStackTrace();
1315        }
1316    }
1317
1318    public void close() {
1319        try {
1320            if (inWriter != null)
1321                inWriter.close();
1322            if (outWrite != null)
1323                outWrite.close();
1324            if (socket != null)
1325                socket.close();
1326        } catch (IOException e) {
1327            e.printStackTrace();
1328        }
1329    }
1330
1331    public void read() {
1332        try {
1333            String line;
1334            while ((line = inWriter.readLine()) != null)
1335                Log.d("read", line);
1336        } catch (IOException e) {
1337            e.printStackTrace();
1338        }
1339    }
1340
1341    public void write(String msg) {
1342        try {
1343            outWrite.write(msg);
1344            outWrite.flush();
1345        } catch (IOException e) {
1346            e.printStackTrace();
1347        }
1348    }
1349
1350    public void close() {
1351        try {
1352            if (inWriter != null)
1353                inWriter.close();
1354            if (outWrite != null)
1355                outWrite.close();
1356            if (socket != null)
1357                socket.close();
1358        } catch (IOException e) {
1359            e.printStackTrace();
1360        }
1361    }
1362
1363    public void read() {
1364        try {
1365            String line;
1366            while ((line = inWriter.readLine()) != null)
1367                Log.d("read", line);
1368        } catch (IOException e) {
1369            e.printStackTrace();
1370        }
1371    }
1372
1373    public void write(String msg) {
1374        try {
1375            outWrite.write(msg);
1376            outWrite.flush();
1377        } catch (IOException e) {
1378            e.printStackTrace();
1379        }
1380    }
1381
1382    public void close() {
1383        try {
1384            if (inWriter != null)
1385                inWriter.close();
1386            if (outWrite != null)
1387                outWrite.close();

```

```
65      }
66  }
67
68  Thread t;
69  public void run(MainActivity di, OnReceiveInterface ori) {
70      t = new Thread(() -> {
71          while(true) {
72              try {
73                  String a = inWriter.readLine();
74                  ori.Action(a, di);
75              } catch (Exception e) {
76                  e.printStackTrace();
77              }
78          }
79      });
80      t.start();
81  }
82 }
83
84
```

```
1 package com.example.tateapp;
2
3 public interface OnReceiveInterface {
4     void Action(String s, MainActivity di);
5 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <ImageView
11        android:id="@+id/tableImageView"
12        android:layout_width="290dp"
13        android:layout_height="400dp"
14        android:layout_marginStart="8dp"
15        android:layout_marginTop="8dp"
16        android:adjustViewBounds="false"
17        android:scaleType="fitXY"
18        app:layout_constraintStart_toStartOf="parent"
19        app:layout_constraintTop_toTopOf="parent"
20        app:srcCompat="@drawable/table" />
21
22    <Button
23        android:id="@+id/startBtn"
24        android:layout_width="150dp"
25        android:layout_height="75dp"
26        android:layout_marginEnd="8dp"
27        android:layout_marginBottom="8dp"
28        android:background="@drawable/start_btn"
29        android:text="START EXERCISE"
30        android:textSize="16sp"
31        app:layout_constraintBottom_toBottomOf="parent"
32        app:layout_constraintEnd_toEndOf="parent" />
33
34    <SeekBar
35        android:id="@+id/speedSeekBar"
36        android:layout_width="425dp"
37        android:layout_height="75dp"
38        android:layout_marginStart="113dp"
39        android:layout_marginTop="170dp"
40        android:rotation="270"
41        app:layout_constraintStart_toStartOf="parent"
42        app:layout_constraintTop_toTopOf="parent" />
43
44    <Button
45        android:id="@+id/saveShotBtn"
46        android:layout_width="85dp"
47        android:layout_height="wrap_content"
48        android:layout_marginStart="4dp"
49        android:layout_marginTop="4dp"
50        android:text="SAVE SHOT"
51        android:textSize="10sp"
52        app:layout_constraintStart_toEndOf="@+id/spinCanvasImageView"
53        app:layout_constraintTop_toBottomOf="@+id/tableImageView" />
54
55    <Button
56        android:id="@+id/editShotBtn"
57        android:layout_width="85dp"
58        android:layout_height="wrap_content"
59        android:layout_marginStart="4dp"
60        android:layout_marginTop="4dp"
61        android:text="EDIT SHOT (3)"
62        android:textSize="10sp"
63        app:layout_constraintStart_toEndOf="@+id/saveShotBtn"
64        app:layout_constraintTop_toBottomOf="@+id/tableImageView" />
65
66    <TextView
67        android:id="@+id/textView"
68        android:layout_width="wrap_content"
```

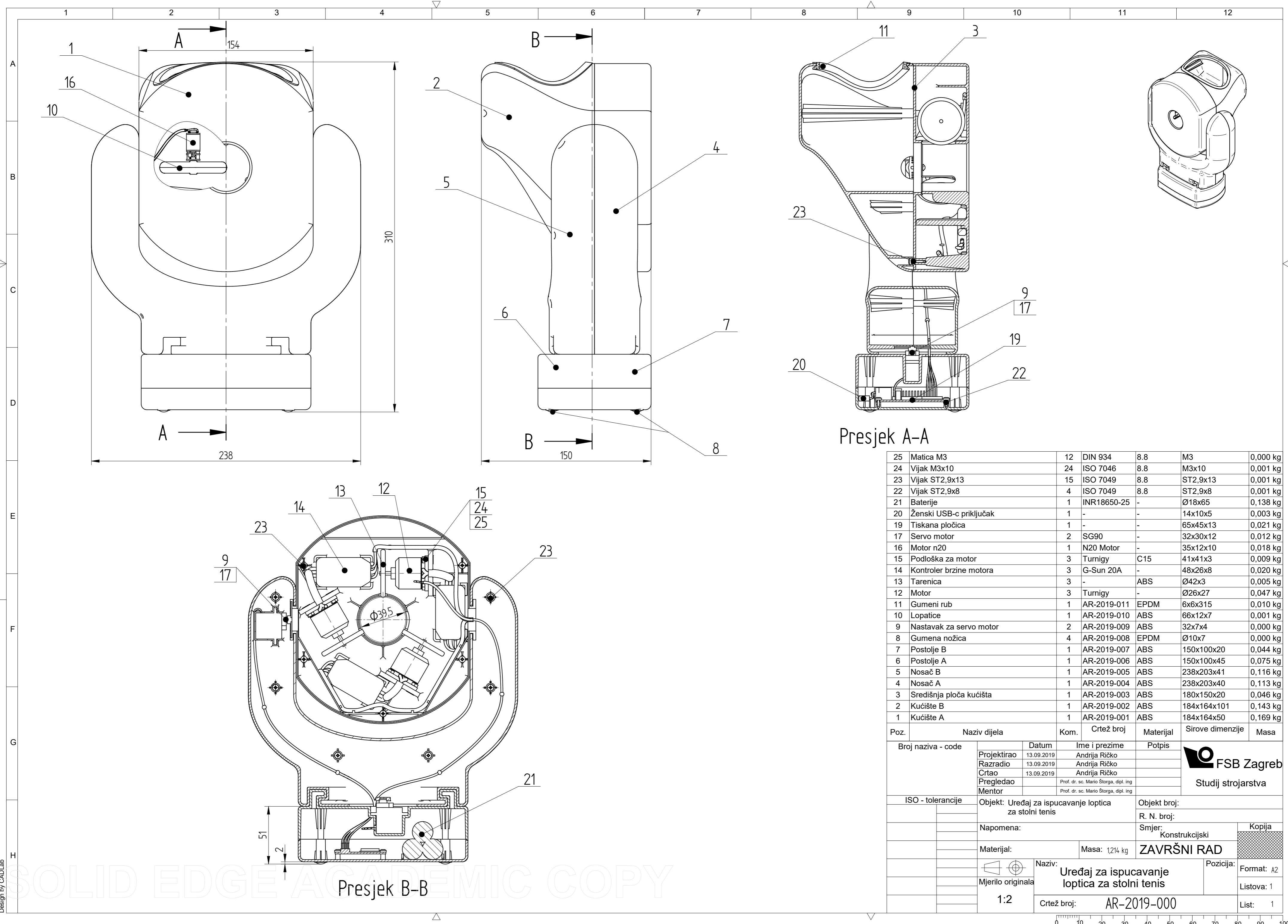
```
69         android:layout_height="wrap_content"
70         android:text="SPIN"
71         android:textColor="@color/deepSkyBlue"
72         android:textSize="36sp"
73         app:layout_constraintBottom_toBottomOf="@+id/spinCanvasImageView"
74         app:layout_constraintEnd_toEndOf="@+id/spinCanvasImageView"
75         app:layout_constraintStart_toStartOf="@+id/spinCanvasImageView"
76         app:layout_constraintTop_toTopOf="@+id/spinCanvasImageView" />
77
78     <Button
79         android:id="@+id/saveExerBtn"
80         android:layout_width="85dp"
81         android:layout_height="wrap_content"
82         android:layout_marginStart="4dp"
83         android:layout_marginTop="4dp"
84         android:text="SAVE EXERCISE"
85         android:textSize="10sp"
86         app:layout_constraintStart_toEndOf="@+id/spinCanvasImageView"
87         app:layout_constraintTop_toBottomOf="@+id/saveShotBtn" />
88
89     <Button
90         android:id="@+id/editExerBtn"
91         android:layout_width="85dp"
92         android:layout_height="wrap_content"
93         android:layout_marginStart="4dp"
94         android:layout_marginTop="4dp"
95         android:text="EDIT EXERCISE (2)"
96         android:textSize="10sp"
97         app:layout_constraintStart_toEndOf="@+id/saveExerBtn"
98         app:layout_constraintTop_toBottomOf="@+id/editShotBtn" />
99
100    <ImageView
101        android:id="@+id/spinCanvasImageView"
102        android:layout_width="140dp"
103        android:layout_height="140dp"
104        android:layout_marginStart="8dp"
105        android:layout_marginTop="8dp"
106        android:scaleType="fitXY"
107        app:layout_constraintStart_toStartOf="parent"
108        app:layout_constraintTop_toBottomOf="@+id/tableImageView"
109        app:srcCompat="@drawable/spin_ball" />
110
111    <ImageView
112        android:id="@+id/spinAxisView"
113        android:layout_width="125dp"
114        android:layout_height="125dp"
115        app:layout_constraintBottom_toBottomOf="@+id/spinCanvasImageView"
116        app:layout_constraintEnd_toEndOf="@+id/spinCanvasImageView"
117        app:layout_constraintStart_toStartOf="@+id/spinCanvasImageView"
118        app:layout_constraintTop_toTopOf="@+id/spinCanvasImageView"
119        app:srcCompat="@drawable/axis_removebg" />
120
121    <ImageView
122        android:id="@+id/spinPointerView"
123        android:layout_width="20dp"
124        android:layout_height="20dp"
125        android:layout_marginEnd="32dp"
126        android:layout_marginBottom="32dp"
127        app:layout_constraintBottom_toBottomOf="@+id/spinAxisView"
128        app:layout_constraintEnd_toEndOf="@+id/spinAxisView"
129        app:srcCompat="@drawable/spin_pointer" />
130
131    <ImageView
132        android:id="@+id/ballView"
133        android:layout_width="20dp"
134        android:layout_height="20dp"
135        android:layout_marginStart="64dp"
136        android:layout_marginBottom="45dp"
```

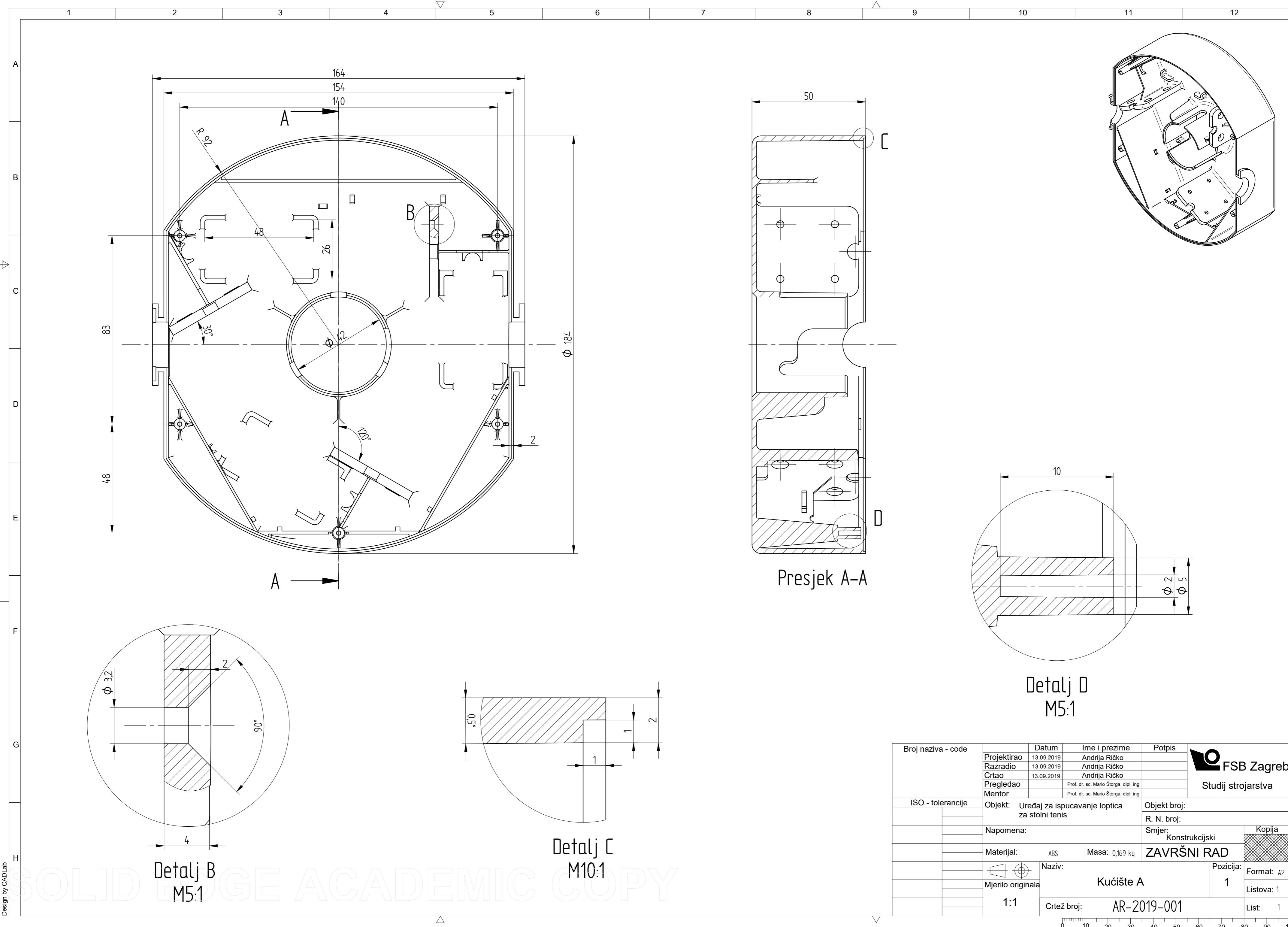
```
137     app:layout_constraintBottom_toBottomOf="@+id/tableImageView"
138     app:layout_constraintStart_toStartOf="@+id/tableImageView"
139     app:srcCompat="@drawable/ping_pong_ball" />
140
141 <ImageView
142     android:id="@+id/deviceView"
143     android:layout_width="35dp"
144     android:layout_height="35dp"
145     android:layout_marginTop="16dp"
146     android:layout_marginEnd="95dp"
147     app:layout_constraintEnd_toEndOf="@+id/tableImageView"
148     app:layout_constraintTop_toTopOf="@+id/tableImageView"
149     app:srcCompat="@drawable/device_nobg" />
150
151 <View
152     android:id="@+id/tableView"
153     android:layout_width="290dp"
154     android:layout_height="400dp"
155     android:layout_marginStart="8dp"
156     android:layout_marginTop="8dp"
157     app:layout_constraintStart_toStartOf="parent"
158     app:layout_constraintTop_toTopOf="parent" />
159
160 <View
161     android:id="@+id/spinCanvasView"
162     android:layout_width="140dp"
163     android:layout_height="140dp"
164     app:layout_constraintStart_toStartOf="@+id/spinCanvasImageView"
165     app:layout_constraintTop_toTopOf="@+id/spinCanvasImageView" />
166
167 <TextView
168     android:id="@+id/textView14"
169     android:layout_width="18dp"
170     android:layout_height="wrap_content"
171     android:layout_marginStart="40dp"
172     android:layout_marginBottom="8dp"
173     android:text="v3:"
174     android:textSize="5sp"
175     app:layout_constraintBottom_toBottomOf="parent"
176     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
177
178 <TextView
179     android:id="@+id/textView2"
180     android:layout_width="18dp"
181     android:layout_height="wrap_content"
182     android:layout_marginStart="40dp"
183     android:layout_marginBottom="8dp"
184     android:text="v2:"
185     android:textSize="5sp"
186     app:layout_constraintBottom_toTopOf="@+id/v3Txt"
187     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
188
189 <TextView
190     android:id="@+id/textView4"
191     android:layout_width="18dp"
192     android:layout_height="wrap_content"
193     android:layout_marginStart="40dp"
194     android:layout_marginBottom="8dp"
195     android:text="v1:"
196     android:textSize="5sp"
197     app:layout_constraintBottom_toTopOf="@+id/textView2"
198     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
199
200 <TextView
201     android:id="@+id/textView5"
202     android:layout_width="18dp"
203     android:layout_height="wrap_content"
204     android:layout_marginStart="40dp"
```

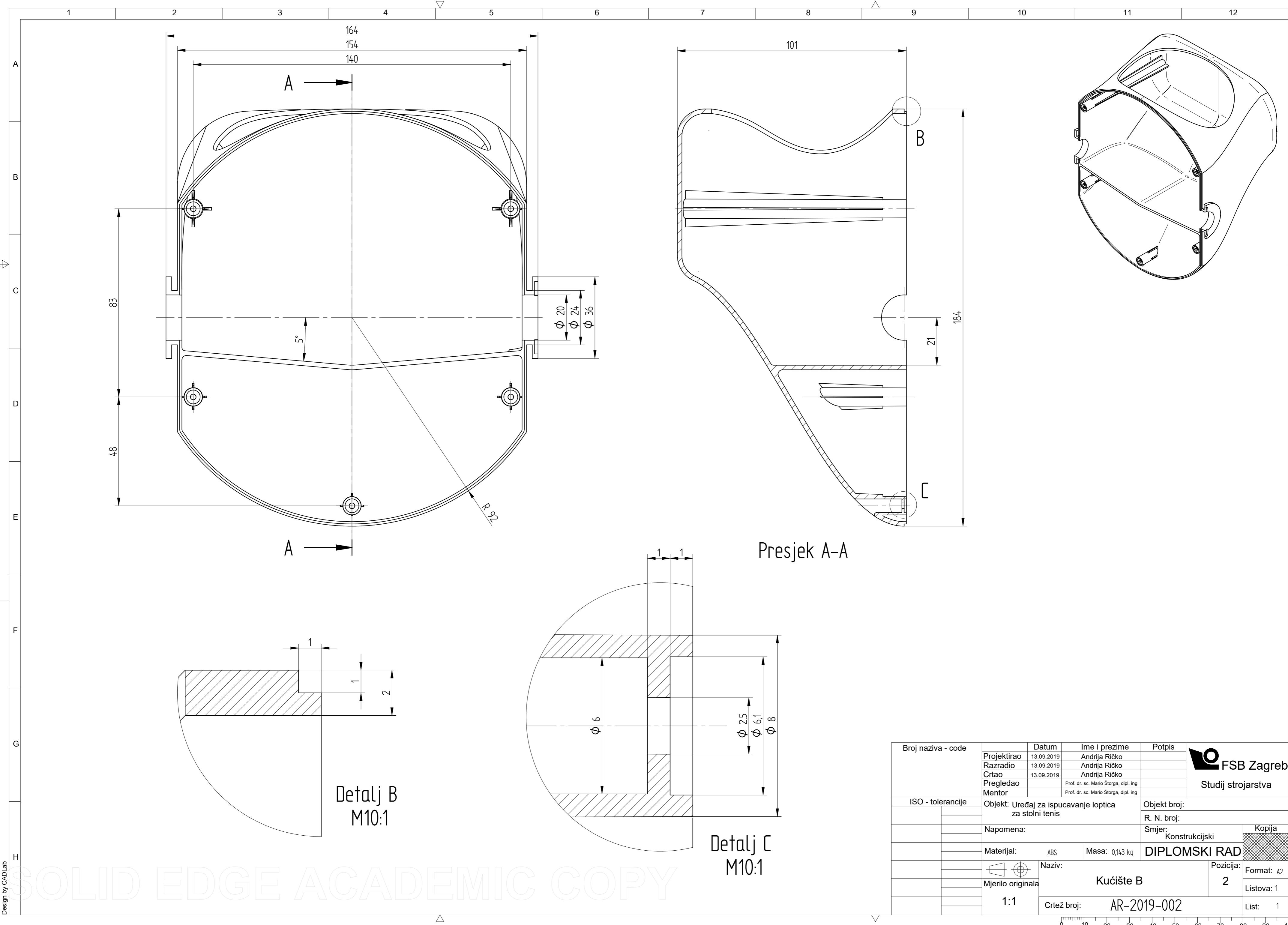
```
205     android:layout_marginBottom="8dp"
206     android:text="v0:"
207     android:textSize="5sp"
208     app:layout_constraintBottom_toTopOf="@+id/textView4"
209     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
210
211 <TextView
212     android:id="@+id/textView6"
213     android:layout_width="18dp"
214     android:layout_height="wrap_content"
215     android:layout_marginStart="40dp"
216     android:layout_marginBottom="8dp"
217     android:text="Alpha:"
218     android:textSize="5sp"
219     app:layout_constraintBottom_toTopOf="@+id/textView5"
220     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
221
222 <TextView
223     android:id="@+id/textView8"
224     android:layout_width="18dp"
225     android:layout_height="wrap_content"
226     android:layout_marginStart="40dp"
227     android:layout_marginBottom="8dp"
228     android:text="Phi:"
229     android:textSize="5sp"
230     app:layout_constraintBottom_toTopOf="@+id/textView6"
231     app:layout_constraintStart_toEndOf="@+id/delaySpinner" />
232
233 <TextView
234     android:id="@+id/v3Txt"
235     android:layout_width="18dp"
236     android:layout_height="wrap_content"
237     android:layout_marginStart="8dp"
238     android:layout_marginBottom="8dp"
239     android:text="0"
240     android:textSize="5sp"
241     app:layout_constraintBottom_toBottomOf="parent"
242     app:layout_constraintStart_toEndOf="@+id/textView14" />
243
244 <TextView
245     android:id="@+id/v2Txt"
246     android:layout_width="18dp"
247     android:layout_height="wrap_content"
248     android:layout_marginStart="8dp"
249     android:layout_marginBottom="8dp"
250     android:text="0"
251     android:textSize="5sp"
252     app:layout_constraintBottom_toTopOf="@+id/v3Txt"
253     app:layout_constraintStart_toEndOf="@+id/textView2" />
254
255 <TextView
256     android:id="@+id/v1Txt"
257     android:layout_width="18dp"
258     android:layout_height="wrap_content"
259     android:layout_marginStart="8dp"
260     android:layout_marginBottom="8dp"
261     android:text="0"
262     android:textSize="5sp"
263     app:layout_constraintBottom_toTopOf="@+id/v2Txt"
264     app:layout_constraintStart_toEndOf="@+id/textView4" />
265
266 <TextView
267     android:id="@+id/v0Txt"
268     android:layout_width="18dp"
269     android:layout_height="wrap_content"
270     android:layout_marginStart="8dp"
271     android:layout_marginBottom="8dp"
272     android:text="0"
```

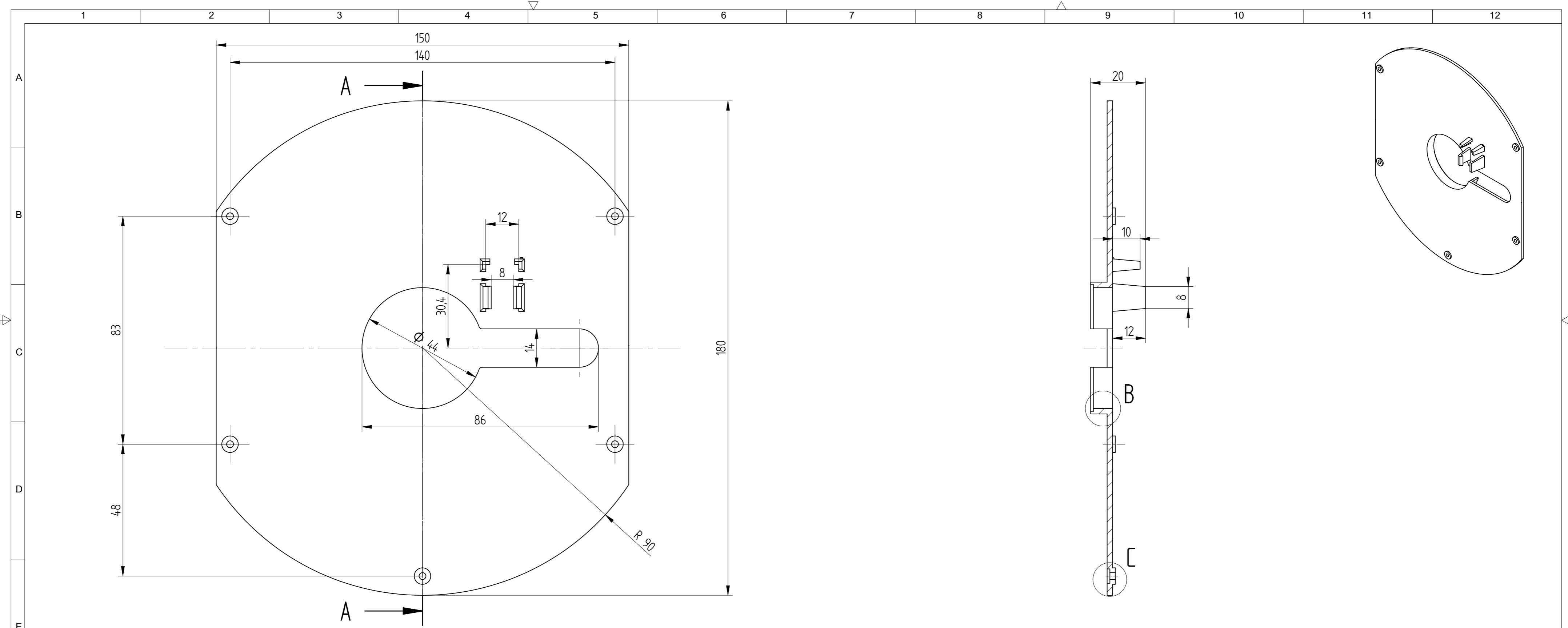
```
273     android:textSize="5sp"
274     app:layout_constraintBottom_toTopOf="@+id/v1Txt"
275     app:layout_constraintStart_toEndOf="@+id/textView5" />
276
277 <TextView
278     android:id="@+id/alphaTxt"
279     android:layout_width="18dp"
280     android:layout_height="wrap_content"
281     android:layout_marginStart="8dp"
282     android:layout_marginBottom="8dp"
283     android:text="0"
284     android:textSize="5sp"
285     app:layout_constraintBottom_toTopOf="@+id/v0Txt"
286     app:layout_constraintStart_toEndOf="@+id/textView6" />
287
288 <TextView
289     android:id="@+id/phiTxt"
290     android:layout_width="18dp"
291     android:layout_height="wrap_content"
292     android:layout_marginStart="8dp"
293     android:layout_marginBottom="8dp"
294     android:text="0"
295     android:textSize="5sp"
296     app:layout_constraintBottom_toTopOf="@+id/alphaTxt"
297     app:layout_constraintStart_toEndOf="@+id/textView8" />
298
299 <Spinner
300     android:id="@+id/delaySpinner"
301     android:layout_width="105dp"
302     android:layout_height="30dp"
303     android:layout_marginStart="8dp"
304     android:layout_marginBottom="8dp"
305     android:background="@color/btnGray"
306     android:entries="@array/delayTime"
307     android:spinnerMode="dropdown"
308     android:textSize="9pt"
309     app:layout_constraintBottom_toBottomOf="parent"
310     app:layout_constraintStart_toStartOf="parent" />
311
312 </android.support.constraint.ConstraintLayout>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     package="com.example.tateapp">
5
6     <uses-permission android:name="android.permission.BLUETOOTH" />
7     <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
8
9     <application
10         android:allowBackup="true"
11         android:icon="@mipmap/ic_launcher"
12         android:label="@string/app_name"
13         android:roundIcon="@mipmap/ic_launcher_round"
14         android:supportsRtl="true"
15         android:theme="@style/AppTheme"
16         tools:ignore="GoogleAppIndexingWarning">
17         <activity android:name=".MainActivity">
18             <intent-filter>
19                 <action android:name="android.intent.action.MAIN" />
20
21                 <category android:name="android.intent.category.LAUNCHER" />
22             </intent-filter>
23         </activity>
24     </application>
25
26 </manifest>
```

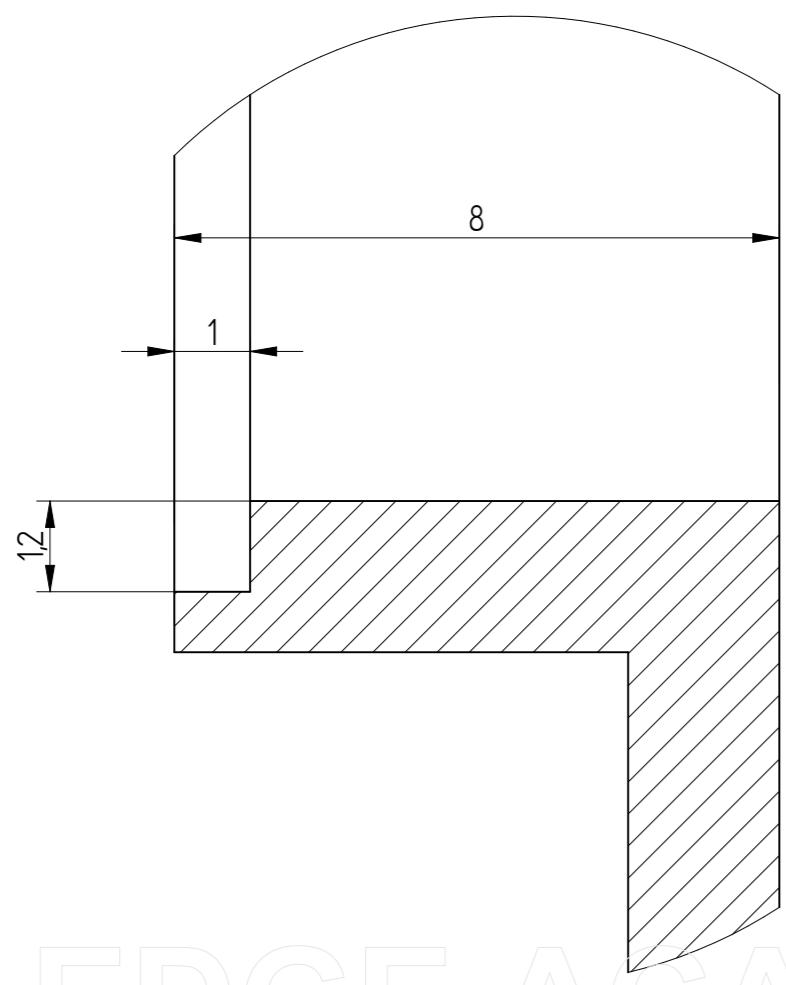




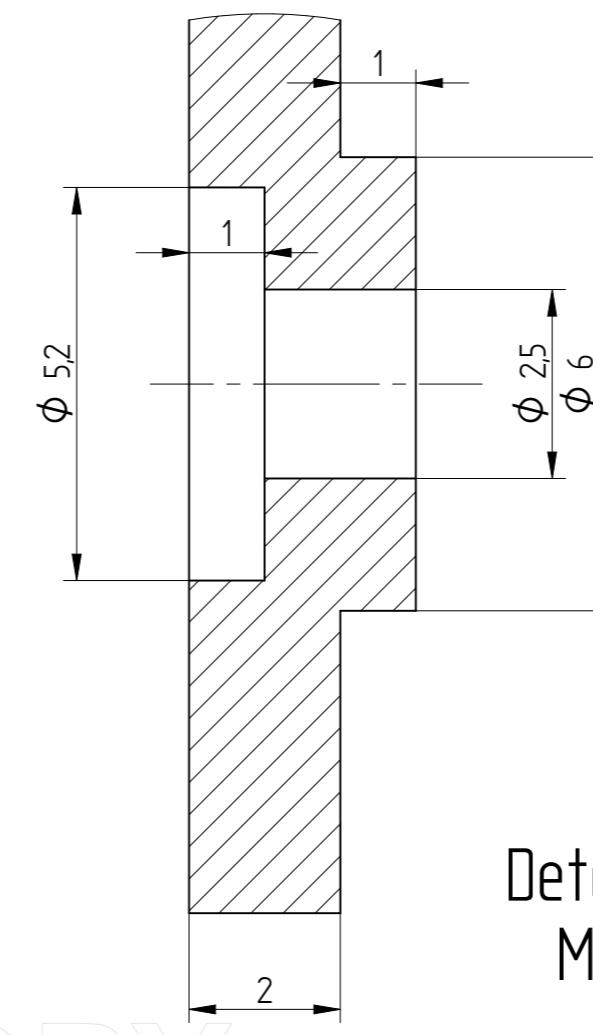




Presjek A-A

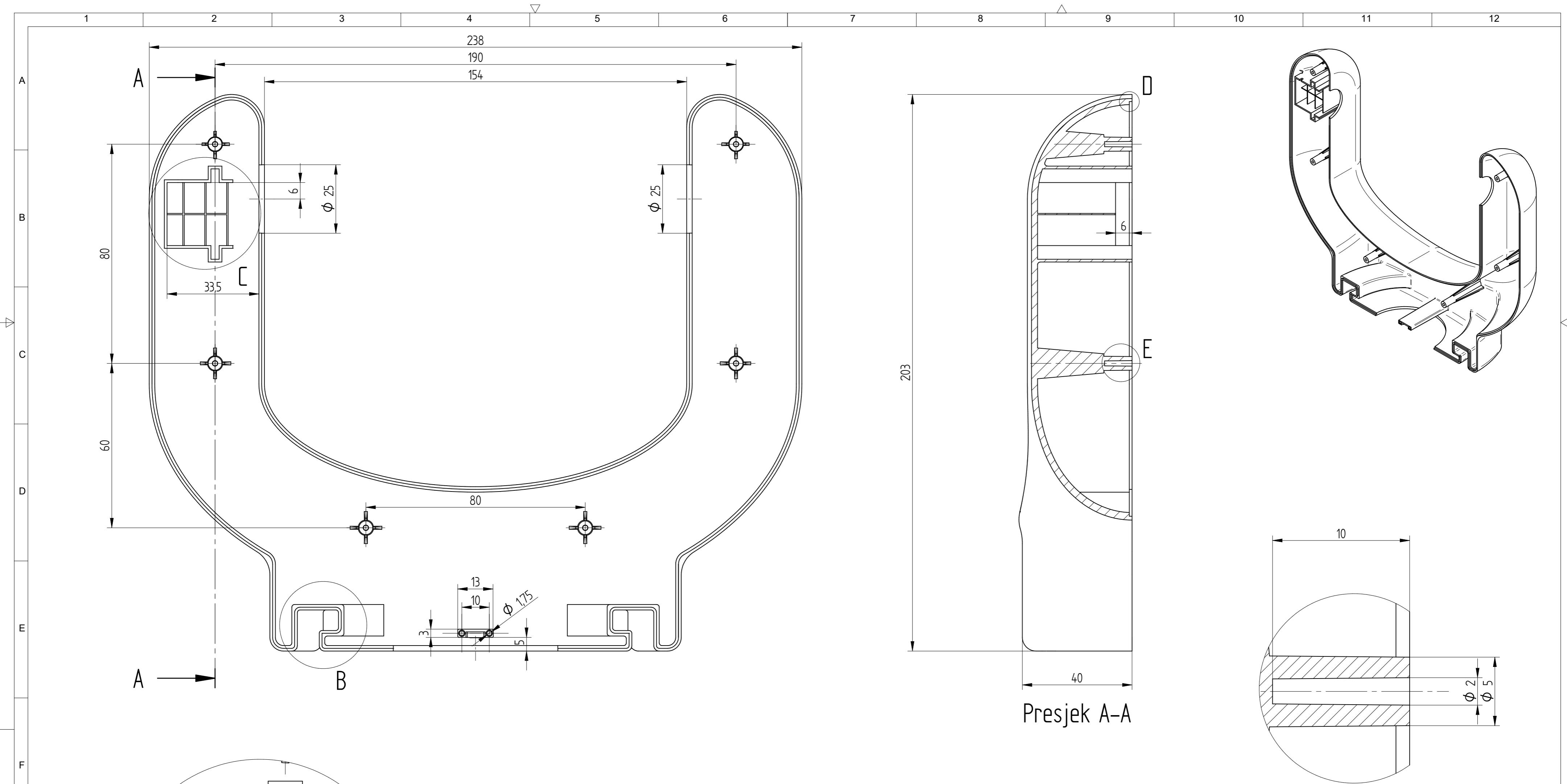


Detalj B  
M10:1



Detalj C  
M10:1

Broj naziva - code	Datum	Ime i prezime	Potpis
Projektirao	13.09.2019	Andrija Ričko	
Razradio	13.09.2019	Andrija Ričko	
Crtao	13.09.2019	Andrija Ričko	
Pregledao	Prof. dr. sc. Mario Štorga, dipl. ing.		
Mentor	Prof. dr. sc. Mario Štorga, dipl. ing.		
ISO - tolerancije	Objekt:	ZAVRŠNI RAD	
	Uredaj za ispučavanje loptica za stolni tenis		
	Objekt broj:		
	R. N. broj:		
	Napomena:	Smjer: Konstrukcijski	Kopija
	Materijal:	Masa: 0,046 kg	
	ABS		
	Naziv:		
Mjerilo originala	1:1	Crtež broj:	AR-2019-003
		Pozicija:	A2
			Format: A2
			Listova: 1
			List: 1



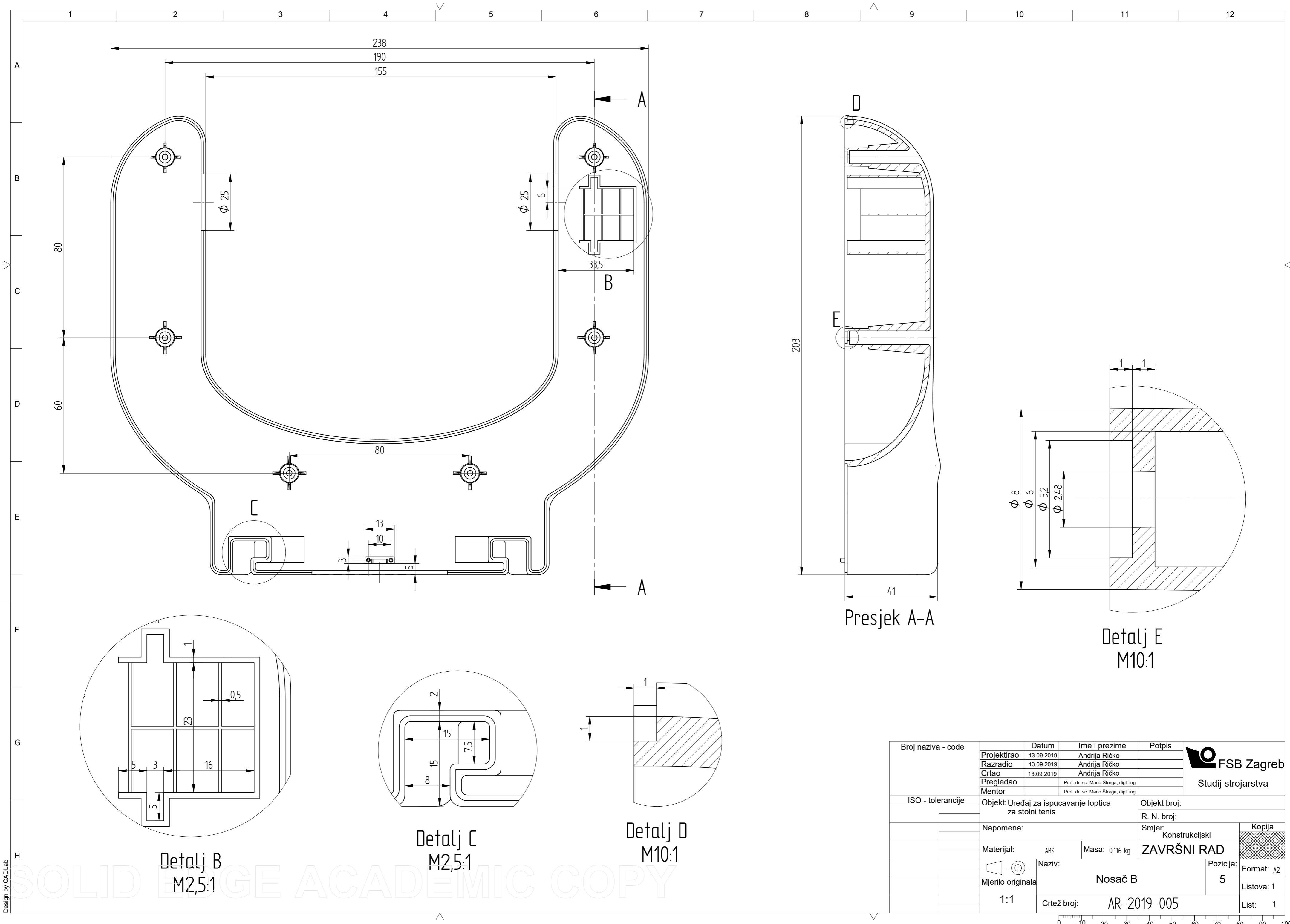
Detalj B  
M2,5:1

Detalj  
M10:1

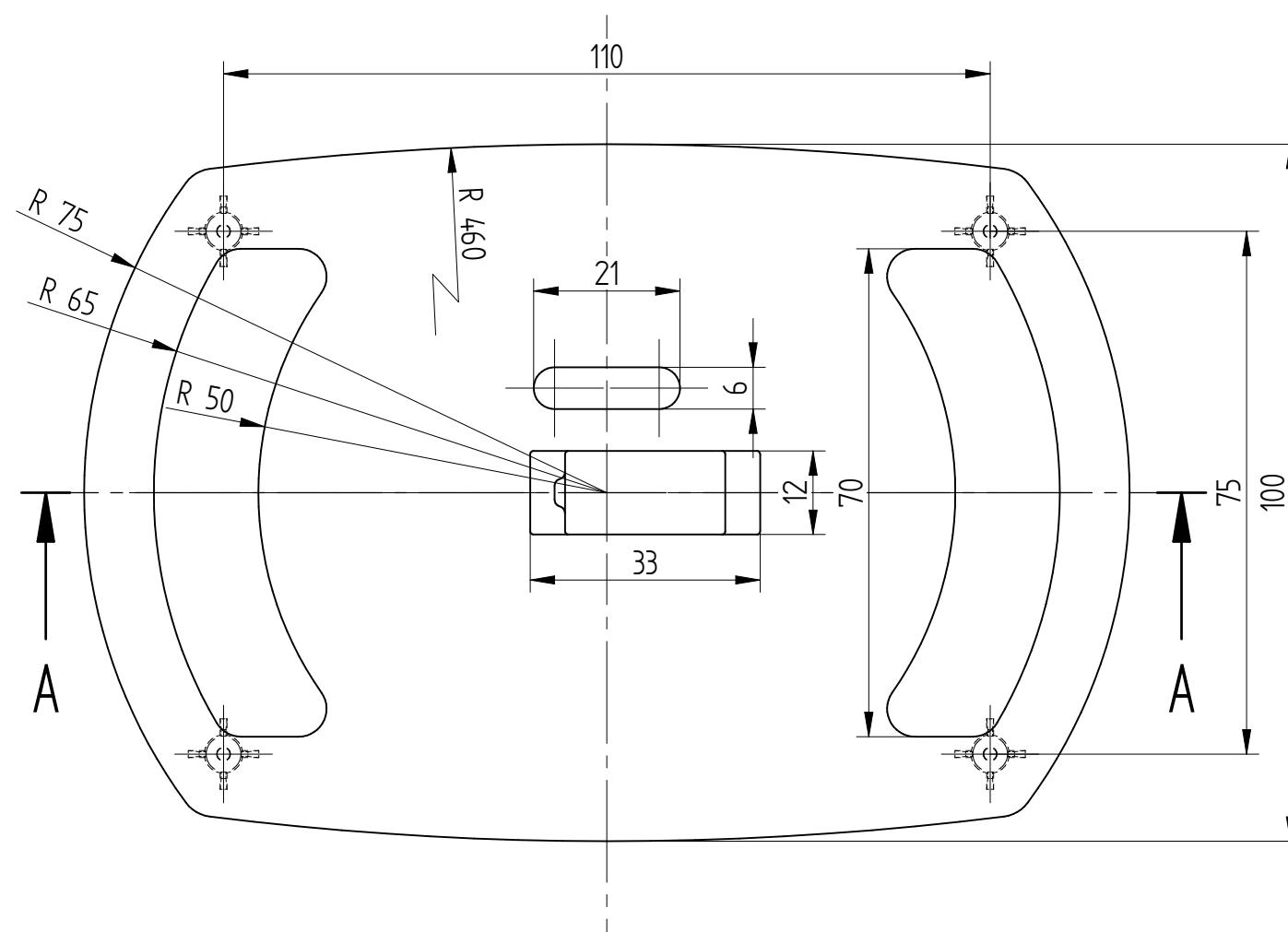
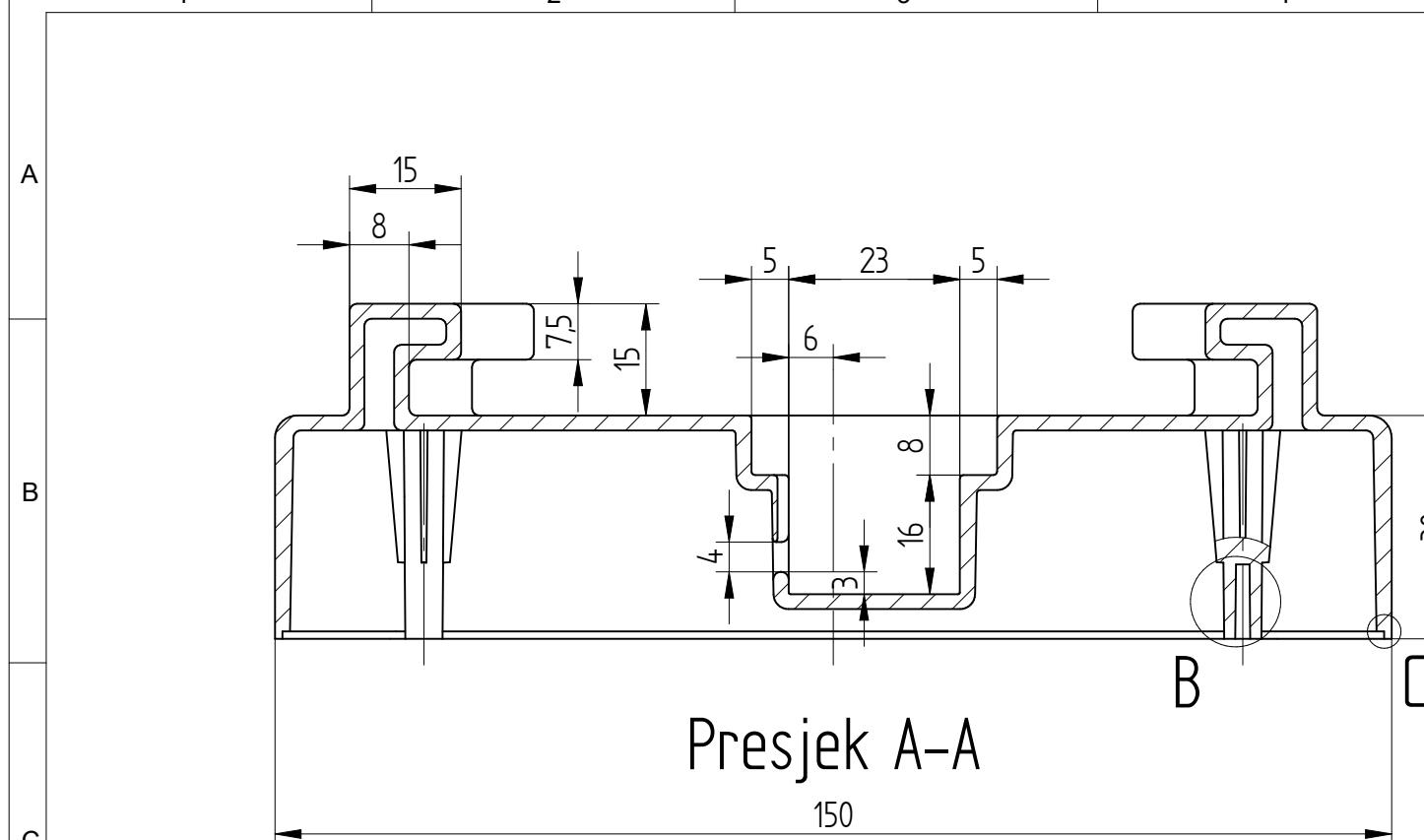
Detalj C  
M2,5:1

Detalj E  
M5:1

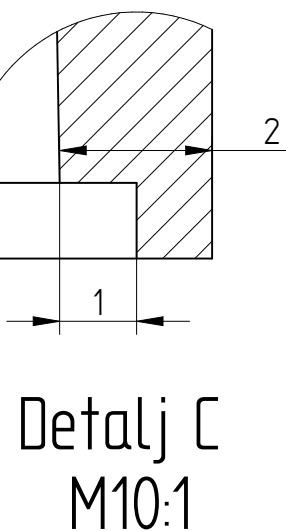
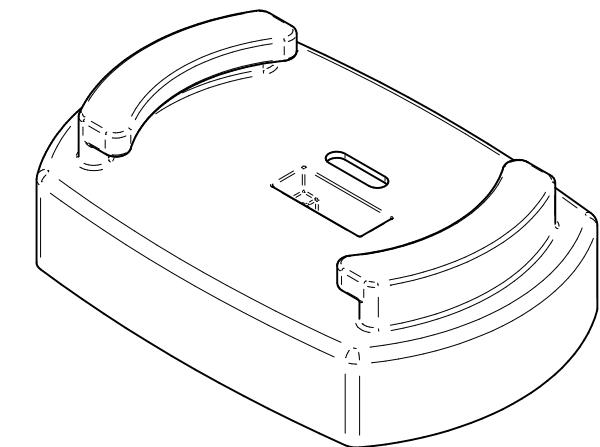
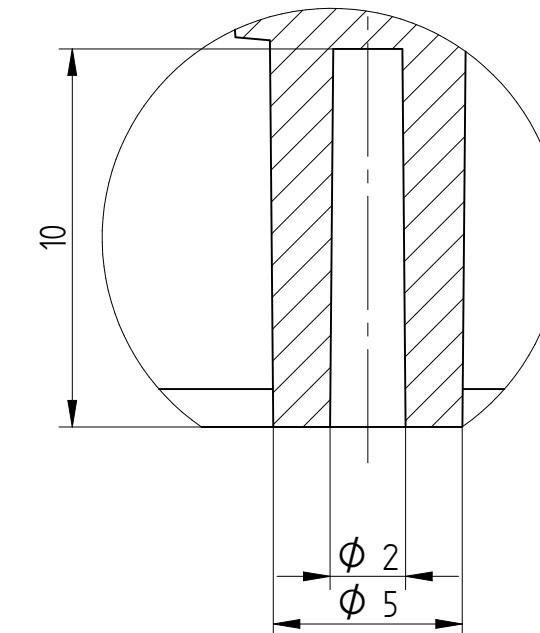
Broj naziva - code		Datum	Ime i prezime	Potpis	 <b>FSB Zagreb</b> Studij strojarstva
		Projektirao	13.09.2019	Andrija Ričko	
		Razradio	13.09.2019	Andrija Ričko	
		Črtao	13.09.2019	Andrija Ričko	
		Pregledao		Prof. dr. sc. Mario Štorga, dipl. ing.	
		Mentor		Prof. dr. sc. Mario Štorga, dipl. ing.	
ISO - tolerancije		Objekt: Uredaj za ispučavanje loptica za stolni tenis		Objekt broj:	
				R. N. broj:	
	Napomena:			Smjer: Konstrukcijski	Kopija
	Materijal:	ABS	Masa: 0,113 kg	<b>ZAVRŠNI RAD</b>	
	 	Naziv: <b>Nosač A</b>			Pozicija:
	Mjerilo originala				4
	1:1	Crtež broj: <b>AR-2019-004</b>			List: 1



1 2 3 4 5 6 7 8



Detalj B  
M5:1



Broj naziva - code	Datum	Ime i prezime	Potpis
Projektirao	13.09.2019	Andrija Ričko	
Razradio	13.09.2019	Andrija Ričko	
Crtao	13.09.2019	Andrija Ričko	
Pregledao		Prof. dr. sc. Mario Štorga, dipl. ing.	
Mentor		Prof. dr. sc. Mario Štorga, dipl. ing.	

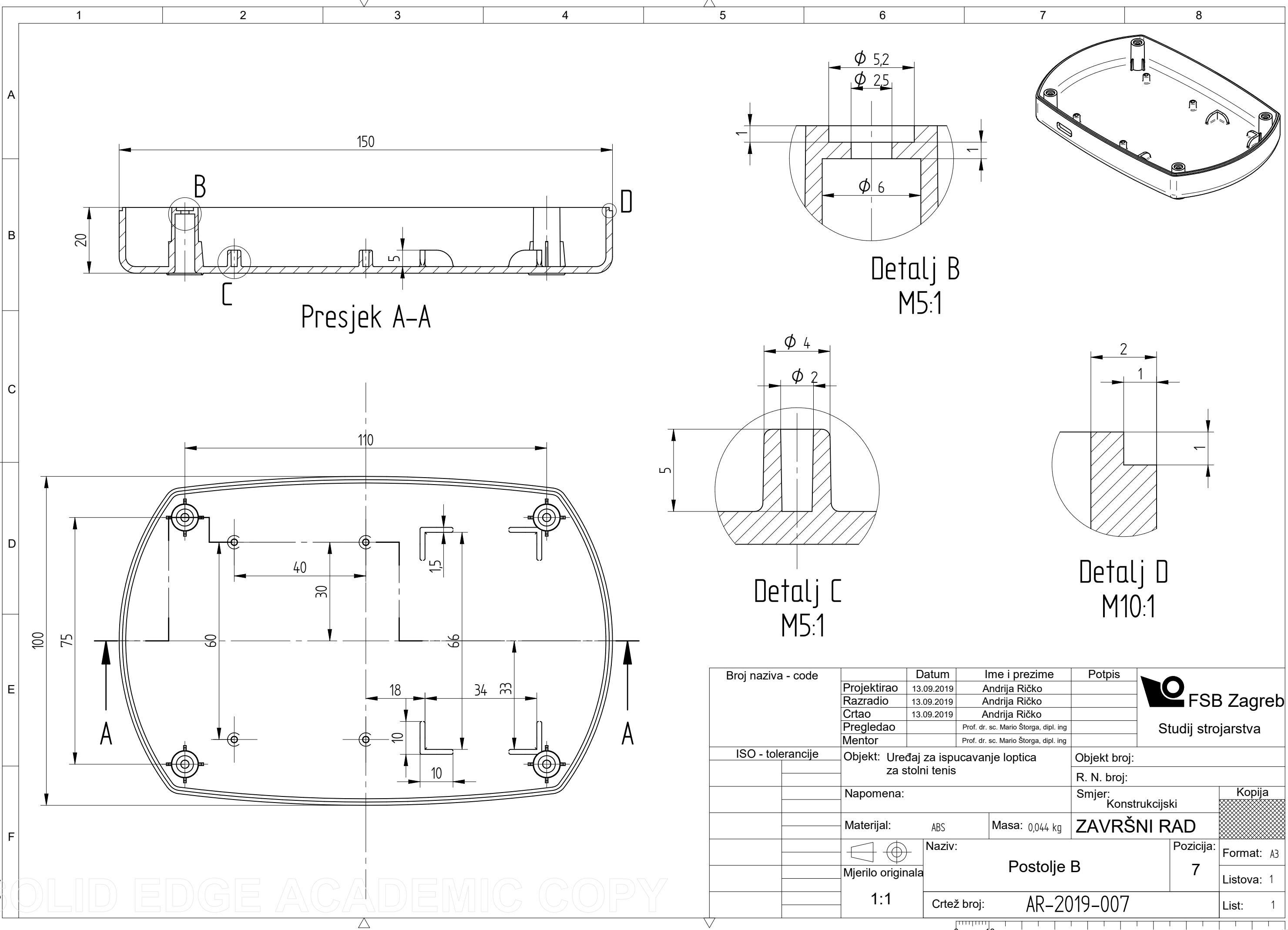
ISO - tolerancije	Objekt: Uredaj za ispučavanje loptica za stolni tenis	Objekt broj:
		R. N. broj:
	Napomena:	Smjer: Konstrukcijski
	Materijal: ABS	Masa: 0,075 kg
		ZAVRŠNI RAD

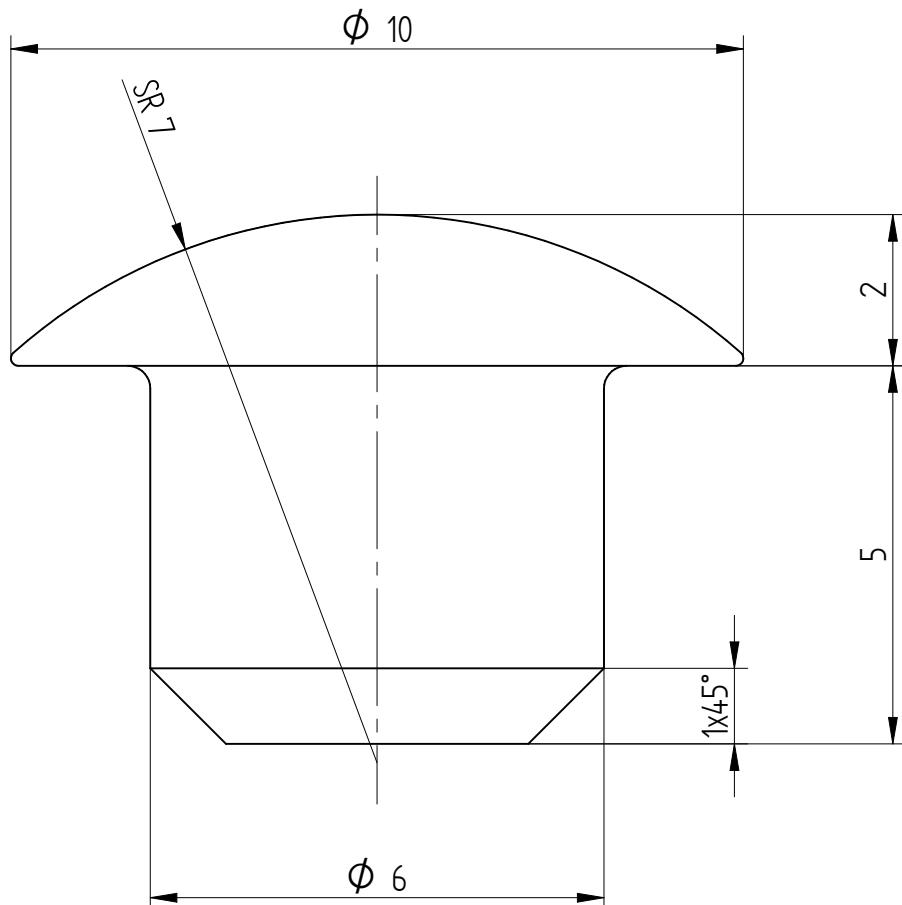
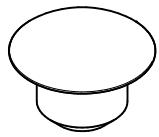
  

Mjerilo originala	Naziv:	Pozicija:
1:1	Postolje A	Format: A3
		6

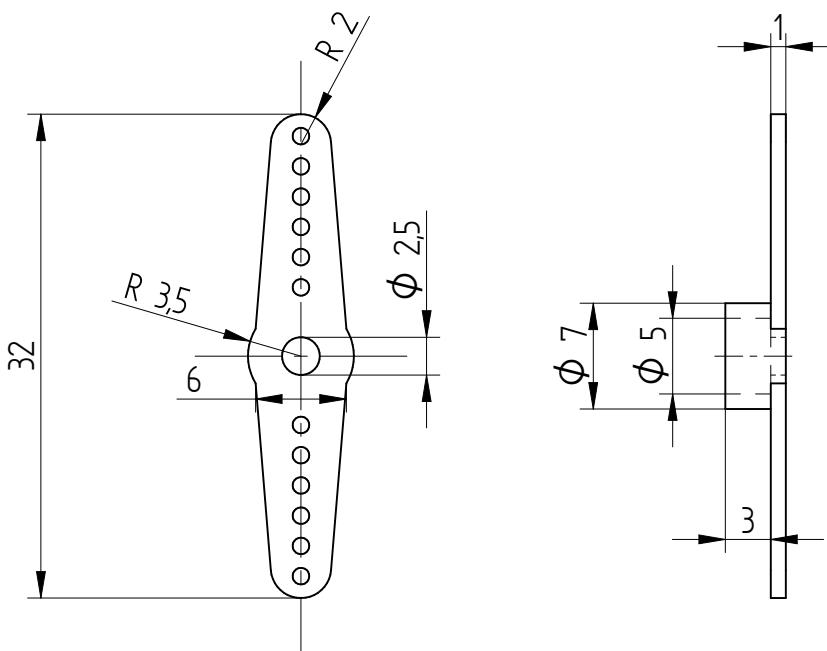
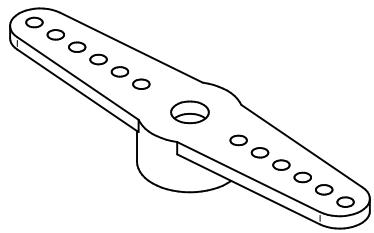
  

Crtež broj:	AR-2019-006	List:
		1

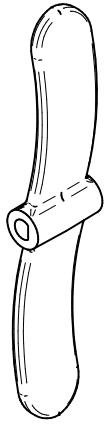
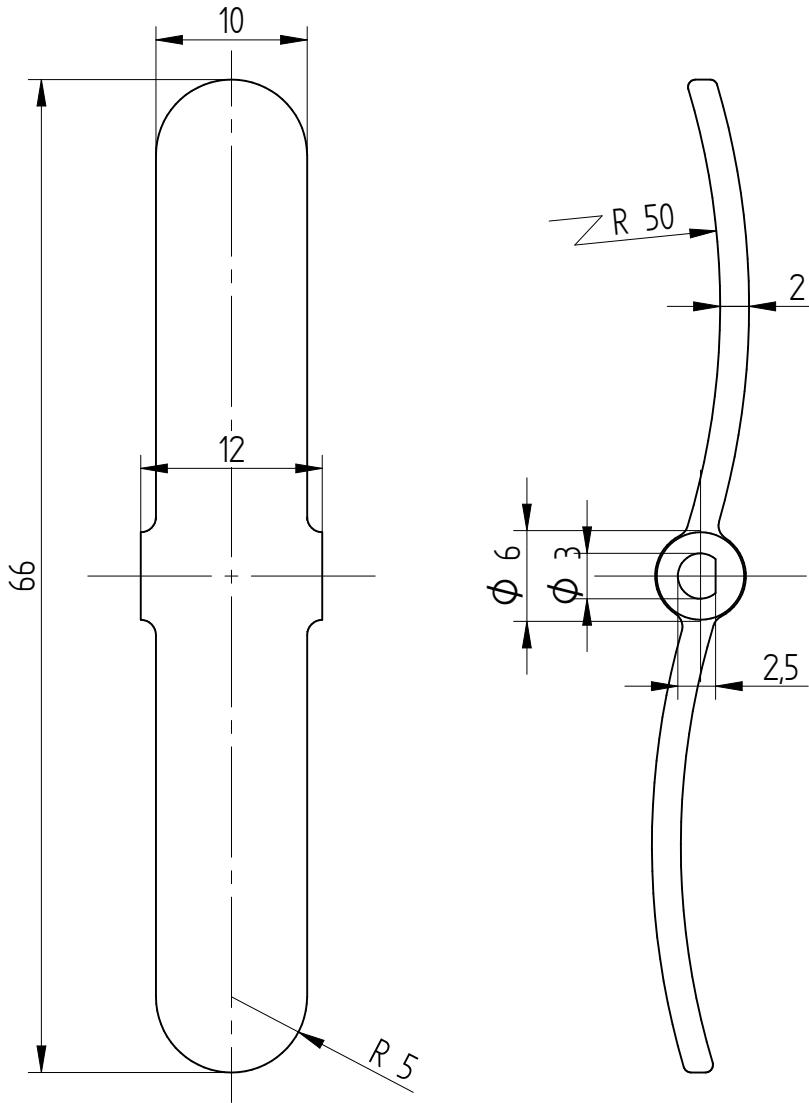




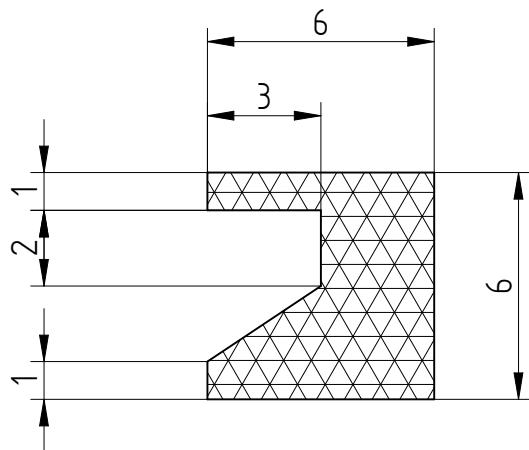
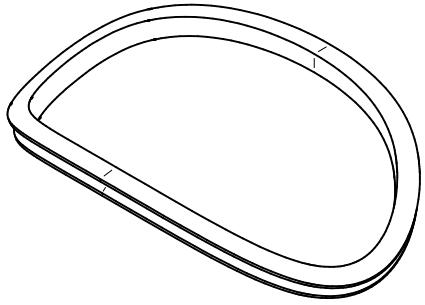
Broj naziva - code		Datum	Ime i prezime	Potpis	 FSB Zagreb
Projektirao		13.09.2019	Andrija Ričko		
Razradio		13.09.2019	Andrija Ričko		
Crtao		13.09.2019	Andrija Ričko		
Pregledao			Prof. dr. sc. Mario Štorga, dipl. ing		
Mentor			Prof. dr. sc. Mario Štorga, dipl. ing		
ISO - tolerancije		Objekt: Uredaj za ispučavanje loptica za stolni tenis	Objekt broj:		
			R. N. broj:		
		Napomena:	ZAVRŠNI RAD	Kopija	
		Materijal: EPDM	Masa: 0 kg		
			Naziv: Gumeni nožica	Pozicija: 8	Format: A4
		Mjerilo originala 10:1	Crtež broj: AR-2019-008		Listova: 1
Design by CADLab					List: 1

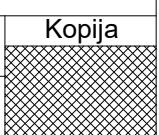


Broj naziva - code	Datum	Ime i prezime	Potpis	
	Projektirao	13.09.2019	Andrija Ričko	
	Razradio	13.09.2019	Andrija Ričko	
	Crtao	13.09.2019	Andrija Ričko	
	Pregledao		Prof. dr. sc. Mario Štorga, dipl. ing	
	Mentor		Prof. dr. sc. Mario Štorga, dipl. ing	
ISO - tolerancije	Objekt: Uredaj za ispučavanje loptica za stolni tenis		Objekt broj:	
			R. N. broj:	
	Napomena:		ZAVRŠNI RAD	Kopija
	Materijal: ABS		0 kg	
 Mjerilo originala	 Naziv: Nastavak za servo motor	Pozicija: 9	Format: A4 Listova: 1	
	Crtež broj:	AR-2019-009	List: 1	



Broj naziva - code		Datum	Ime i prezime	Potpis	 FSB Zagreb	
	Projektirao	13.09.2019	Andrija Ričko			
	Razradio	13.09.2019	Andrija Ričko			
	Crtao	13.09.2019	Andrija Ričko			
	Pregledao		Prof. dr. sc. Mario Štorga, dipl. ing			
	Mentor		Prof. dr. sc. Mario Štorga, dipl. ing			
ISO - tolerancije	Objekt: Uredaj za ispucavanje loptica za stolni tenis			Objekt broj:		
				R. N. broj:		
	Napomena:			ZAVRŠNI RAD	Kopija	
	Materijal: ABS					
	Masa: 0,001 kg					
	 Mjerilo originala	Naziv:  Lopatice	Pozicija:  10	Format:  A4		
	2:1	Crtež broj:  AR-2019-010		Listova:  1		
Design by CADLab				List:  1		



Broj naziva - code		Datum	Ime i prezime	Potpis	 FSB Zagreb	
	Projektirao	13.09.2019	Andrija Ričko			
	Razradio	13.09.2019	Andrija Ričko			
	Crtao	13.09.2019	Andrija Ričko			
	Pregledao		Prof. dr. sc. Mario Štorga, dipl. ing			
	Mentor		Prof. dr. sc. Mario Štorga, dipl. ing			
ISO - tolerancije	Objekt: Uredaj za ispučavanje loptica za stolni tenis			Objekt broj:		
				R. N. broj:		
	Napomena: Potrebna dužina gume je 315mm			ZAVRŠNI RAD	Kopija	
	Materijal: EPDM					
	 Mjerilo originala	Naziv: Gumeni rub			Pozicija:	
					Format: A4	
	 5:1	Crtež broj: AR-2019-011			Listova: 1	
					List: 1	