Automatsko izuzimanje objekata temeljem oblika

Pavlović, Vid

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Vid Pavlović

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Vid Pavlović

Zagreb, 2023.

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

Zahvaljujem se dr.sc. Tomislavu Stipančiću na pomoći pri izradi završnog rada, savjetima o usavršavanju ovog rada i korištenje laboratorijske opreme.

Zahvaljujem se dr.sc. Danijelu Pavkoviću na dozvoli za korištenje pokretne trake.

Zahvaljujem se obitelji i prijateljima na velikoj podršci tijekom izrade ovog rada i tijekom studiranja.

Zahvaljujem zaposlenima i studentima u Laboratoriju za projektiranje izradbenih i montažnih sustava na pomoći i savjetima za završni rad.

Vid Pavlović

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23 -**	

ZAVRŠNI ZADATAK

Student: **Vid Pavlović** JMBAG: **0035228634**

Naslov rada na hrvatskom jeziku: **Automatsko izuzimanje objekata temeljem oblika**

Naslov rada na engleskom jeziku: **Automatic separation of objects based on shape**

Opis zadatka:

Računalni modeli temeljeni na algoritmima umjetne inteligencije omogućavaju automatsko prepoznavanje objekata koristeći vizijske senzore smještene u stvarnu okolinu. Prepoznavanje je moguće temeljiti na različitim značajkama objekata te na različitim metodama iz područja strojnog vida. Razvijenu programsku podršku je potom moguće koristiti u sklopu brojnih primjena u sklopu automatizacije proizvodnih sustava.

U radu je potrebno:

- proučiti metode i algoritme za detekciju objekata u stvarnom vremenu,
- upoznati se s OpenCV programskom bibliotekom otvorenog koda te koristiti prikladne metode u svrhu prepoznavanja objekata na temelju oblika,
- proučiti karakteristike postojećeg postava temeljenog na Arduino dostavnom sustavu koji se sastoji od kamere, servo motora pokretne trake, servo motora za sortiranje predmeta, te laserskog odašiljača i prijemnika,
- razviti i implementirati programsku podršku koja uključuje brojač predmeta, sortiranje predmeta prema obliku te omogućuje povezivanje komponenti koristeći serijsku komunikaciju,
- modelirati prikladno kupolno osvjetljenje koje anulira utjecaj reflektiranog svjetla te tako pospješuje rad vizijske aplikacije,
- dati kritički osvrt na razvijenu aplikaciju.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:

30. 11. 2022.

Zadatak zadao:

Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:

1. rok: 20. 2. 2023.
2. rok (izvanredni): 10. 7. 2023.
3. rok: 18. 9. 2023.

Predviđeni datumi obrane:

1. rok: 27. 2. – 3. 3. 2023.
2. rok (izvanredni): 14. 7. 2023.
3. rok: 25. 9. – 29. 9. 2023.

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

100-01-001	Profili nosača
100-01	Nosač
100-02-002	Kupola
100-02-003	Nosač kupole
100-02-004	Poklopac kupole
100-02-005	Držač kamere
100-02-006	Držač za C270
100-02	Nosač za kameru i kupolno svjetlo

POPIS OZNAKA

Oznaka	Jedinica	Opis
$\mathbf{A}$	-	Matrica s elementima ulazne slike
G	-	Funkcija Gaussovog filtra
$\mathbf{G}$	-	Gradijent rubova
$\mathbf{G}_x$	-	Gradijent u horizontalnom smjeru
$\mathbf{G}_y$	-	Gradijent u vertikalnom smjeru
$\mathbf{K}$	-	Gaussov filter
status	-	Pomoćna varijabla
x, y	Piksel (px)	Koordinate sustava
x_{centroid}	Piksel (px)	Horizontalna koordinata centroida
x_{granica}	Piksel (px)	Horizontalna koordinata granice za okidanje
θ	°	Kut smjera gradijenta
σ	-	Standardna devijacija Gaussove funkcije

SAŽETAK

Računalni i strojni vid postaju jedne od najkorištenijih područja umjetne inteligencije u industriji. Temeljni princip računalnog vida sastoji se od prepoznavanja rubova, te detekcije kontura i oblika. Cilj ovog rada je predstaviti osnove računalnog vida, Cannyev algoritam za prepoznavanje rubova , uvod u mikrokontrolere i vrste osvjetljenja u primjeni računalnog vida. Aplikacija za prepoznavanje oblika napravljena je pomoću Python programskog jezika i OpenCV biblioteke. Sustav sortiranja upravljan je pomoću Arduino mikrokontrolera. Konstruirano je kupolno svjetlo i nosač za kameru pomoću tehnologije 3D printanja i aluminijskih profila. Sustav je povezan tako da Python program prepoznaje oblik koji pošalje Arduino mikrokontroleru, koji će predmet sa pokretne trake sortirati na prikladno mjesto. Za svaki dio sustava analizirane su prednosti i nedostaci.

Ključne riječi: Canny algoritam, detekcija kontura, OpenCV, Arduino, servo motori, pokretna traka, serijska komunikacija, nosač za kameru i osvjetljenje, kupolno svjetlo

SUMMARY

Computer and machine vision are becoming one of the most used areas of artificial intelligence in industry. The fundamental principle of computer vision consists of edge recognition and contour and shape detection. The aim of this paper is to present the basics of computer vision, Canny's algorithm for edge recognition, an introduction to microcontrollers and types of lighting in the application of computer vision. The shape recognition application was made using the Python programming language and the OpenCV library. The sorting system is controlled by an Arduino microcontroller. The dome light and camera stand was constructed using 3D printing technology and aluminum profiles. The system is configured so that the Python program recognizes the shape and sends it to the Arduino microcontroller, which will sort the object from the conveyor belt into the appropriate place. Advantages and disadvantages were analyzed for each part of the system.

Key words: Canny algorithm, contour detection, OpenCV, servo motors, conveyor belt, serial communication, camera and lighting mount, diffuse dome light

1. UVOD

U današnjoj industriji stavlja se naglasak na automatizaciju i brzinu rada. Rad čovjeka zamjenjuje se strojevima. Umjesto da čovjek upravlja procesom velike brzine, i to nekoliko sati dnevno, strojevi imaju mogućnost raditi istu stvar, bez prestanka. Spajanjem računalnog ili strojnog vida s industrijskim procesom, više nema potrebe da ljudi nadgledaju i upravljaju tim procesom.

1.1. Tema završnog rada

Tema ovog završnog rada je prepoznavanje oblika pomoću programske biblioteke OpenCV. Program za prepoznavanje oblika napisan je u programskom jeziku Python. Korištenjem algoritma za prepoznavanje rubova, program mora uspješno prepoznati oblike na pokretnoj traci. Pokretna traka upravljana je Arduino mikrokontrolerom. Kako bi program nakon uspješne detekcije oblika poslao informaciju mikrokontroleru, koristi se USB serijska komunikacija. Iako sustav pokretne trake ima puno komponenti i namjena, ovaj sustav sastoji se od nekoliko servo motora, laserskog odašiljača i prijemnika. Ostali dijelovi nisu toliko bitni za temu ovog završnog rada. Arduino mikrokontroler mora na temelju primljene informacije sortirati predmet na njegovu ispravnu lokaciju. Za što bolju detekciju rubova, konstruiran je nosač za kameru s kupolnim svjetlom.

1.2. Struktura rada

Rad je strukturiran tako da su u početku objašnjene osnove računalnog vida. U drugom poglavlju razrađuju se korištene biblioteke za razvoj aplikacije, Cannyev algoritam za prepoznavanje rubova, kamera korištena u radu, te kratki uvod u računalni vid. U trećem poglavlju opisan je program za detekciju rubova, prepoznavanje oblika i logika brojača. Četvrto poglavlje sadrži kratki opis sustava pokretne trake, Arduino mikrokontrolera i princip serijske USB komunikacije. Peto poglavlje razrađuje logiku i program mikrokontrolera. Šesto poglavlje sadrži opis konstrukcije nosača i osvjetljenja, kao i princip kupolnog svjetla. Na samom kraju, priložen je zaključak i kritički osvrt na cijeli sustav.

2. DETEKCIJA OBJEKTA TEMELJOM OBLIKA

Računalni vid je polje umjetne inteligencije koje omogućuje računalima i sustavima da izvedu značajne informacije iz digitalnih slika, videozapisa i drugih vizualnih ulaza i poduzmu radnje ili daju preporuke na temelju tih informacija. Ako umjetna inteligencija omogućuje računalima da razmišljaju, računalni vid im omogućuje da vide, promatraju i razumiju. Računalni vid funkcionira gotovo isto kao i ljudski vid, osim što ljudi imaju prednost. Ljudski vid ima prednost doživotnog konteksta za treniranje kako razlikovati objekte, koliko su udaljeni, kreću li se i postoji li nešto pogrešno na slici. Računalni vid trenira strojeve za obavljanje ovih funkcija, ali to mora učiniti u mnogo kraćem vremenu s kamerama, podacima i algoritmima umjesto mrežnice, optičkih živaca i vidnog korteksa. Budući da sustav osposobljen za pregled proizvoda ili promatranje proizvodnog sredstva može analizirati tisuće proizvoda ili procesa u minuti, uočavajući neprimjetne nedostatke ili probleme, može brzo nadmašiti ljudske sposobnosti [1].

Računalni vid vrlo je raširen u današnjem svijetu, gdje mobilni uređaji sadrže algoritme za prepoznavanje korisnika i internet tražilice sadrže algoritme prepoznavanja slika i pronalaska sličnih slika i/ili slika s istim značenjem.

2.1. Strojni vid

Iako se smatraju istom tehnologijom, računalni vid i strojni vid različiti su termini koji se koriste za tehnologije koje se poklapaju. Računalni vid u širokom pogledu odnosi se na snimanje i automatizaciju analize slike s naglaskom na funkciju analize slike u širokom rasponu teorijskih i praktičnih primjena. Strojni vid tradicionalno se odnosi na korištenje računalnog vida u industrijskoj ili praktičnoj primjeni, gdje je potrebno izvršiti određene funkcije ovisno o analizi koja je odrađena od strane vizualnog sustava [2].

Strojni vid je sve raširenji u današnjoj industriji. Strojni vid ima razne primjene, od kontrole kvalitete, sustava pozicioniranja i orijentiranja robotskih manipulatora, te kao i u ovom radu, sortiranje predmeta.

2.2. Cannyev algoritam za detekciju rubova

Kako bi uspješno prepoznali oblik na slici, potrebno je predmete sa slike odvojiti od pozadine. Postoje mnogi algoritmi koji detektiraju rubove. U ovom radu koristiti će se Cannyev algoritam. Cannyev algoritam za detekciju rubova jedan je od popularnijih algoritma korišten u računalnom vidu. Razvio ga je John F. Canny 1986. godine. To je višekoračni algoritam za detekciju rubova. Algoritam se sastoji od sljedećih koraka:

1.) Odstranjivanje i filtriranje šuma sa slike pomoću Gaussovog filtra
2.) Pronalazak intenziteta gradijenta slike
3.) Primjena ne-maksimalnog suzbijanja
4.) Određivanje praga histerezom

U nastavku detaljnije su opisani koraci Cannyevog algoritma.

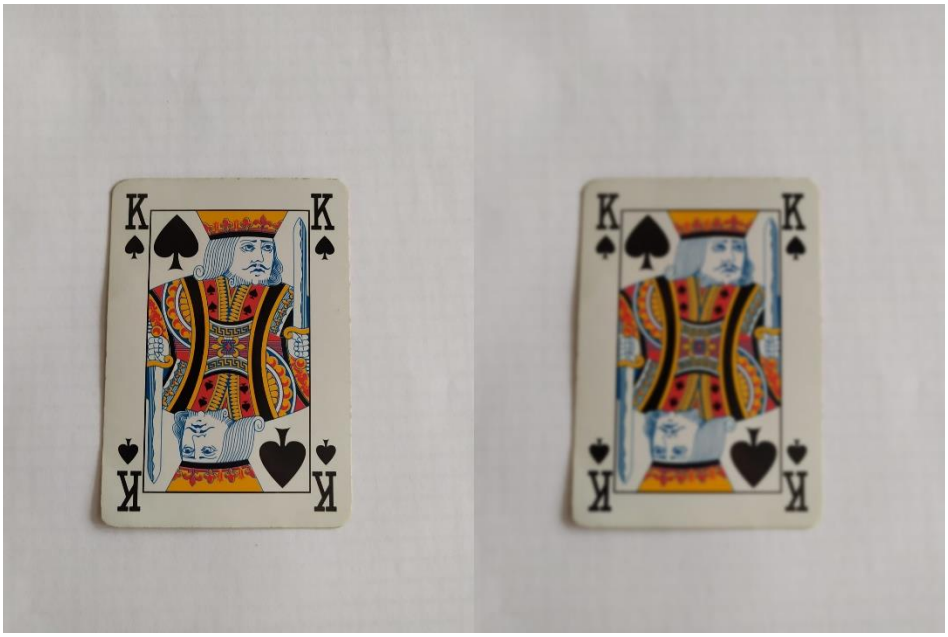
2.2.1. Odstranjivanje i filtriranje šuma sa slike pomoću Gaussovog filtra

Detekcija rubova osjetljiva je na šum u slikama. Zato se primjenjuje Gaussov filter, kako bi se smanjio šum i kako bi se odstranile neželjene informacije sa slike. Nakon primjene Gaussovog filtra, dobivamo masku koju stavljamo na originalnu sliku. Ta maska sadrži puno manje informacija, pa je time i rad Cannyevog algoritma puno brži. Gaussova funkcija koristi se za izračun transformacije koja se primjenjuje na svaki piksel, te se tako dobiva maska. Gaussova funkcija u dvije dimenzije može se opisati sljedećom jednadžbom:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{1}$$

gdje je *x* udaljenost od središnjeg piksela u horizontalnom smjeru, *y* je udaljenost od središnjeg piksela u vertikalnom smjeru, a σ je standardna devijacija Gaussove funkcije.

Slika 1. prikazuje primjer Gaussovog filtra



Slika 1. Primjer Gaussovog filtra

Na slici 1. može se vidjeti normalna slika (lijevo) i slika nakon Gaussovog filtra (desno).

Primjer Gaussovog filtra **K** veličine 3x3 sa standardnom devijacijom $\sigma = 1$ glasi:

$$\mathbf{K} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2}$$

Veličina matrice filtra pridodaje utjecaj daljnjih piksela od središnjeg piksela.

2.2.2. Pronalazak intenziteta gradijenta slike

Rub na slici može biti orijentiran u različitim smjerovima, tako da algoritam Cannyev koristi četiri filtra za otkrivanje vodoravnih, okomitih i dijagonalnih rubova na zamućenoj slici. Operator detekcije ruba (kao što je Roberts, Prewitt ili Sobel) vraća vrijednost za prvu derivaciju u vodoravnom smjeru ($\mathbf{G}_x$) i okomitom smjeru ($\mathbf{G}_y$) [3].

Koristeći Sobelov operator moguće je odrediti gradijent, odnosno usmjerenu promjenu intenziteta elemenata slike, u horizontalnom i vertikalnom smjeru. U izrazima (3) i (4), $\mathbf{G}_x$ predstavlja gradijent u horizontalnom smjeru, dok $\mathbf{G}_y$ predstavlja gradijent u vertikalnom smjeru, dok je **A** matrica koja sadrži podatke slike, te znak $*$ predstavlja operaciju konvolucije.

$$\mathbf{G}_{\mathrm{x}(i,j)} = \mathbf{A}_{(i,j)} * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \tag{3}$$

$$\mathbf{G}_{\mathrm{y}(i,j)} = \mathbf{A}_{(i,j)} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \tag{4}$$

Prema predznacima u izrazu (5), može se vidjeti kako pozitivan rezultat ukazuje na prelazak iz tamnijeg u svjetlije područje gledajući sliku od lijeva prema desno po redovima ili prema od gore prema dolje po stupcima. Sukladno tome, negativan rezultat ukazuje na prelazak iz svjetlijeg u tamnije područje, odnosno prelazak iz područja manjeg u područje većeg intenziteta.

$$\mathbf{G}_{(i,j)} = \sqrt{\mathbf{G}_{\mathrm{x}(i,j)}^2 + \mathbf{G}_{\mathrm{y}(i,j)}^2}\,, \tag{5}$$

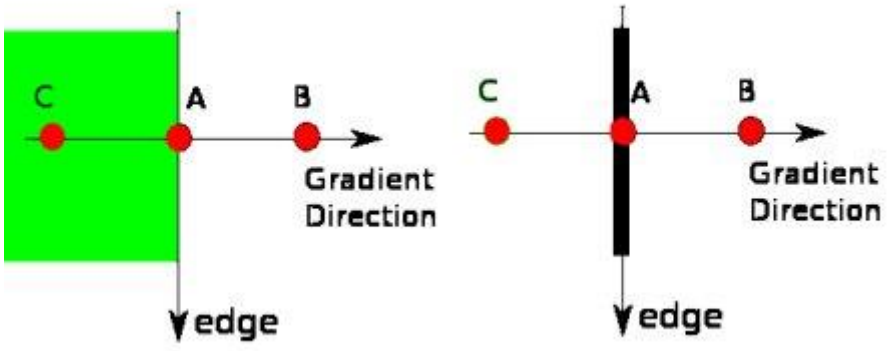
$$\theta(x,y) = \tan^{-1}\left(\frac{\mathbf{G}_{\mathrm{y}(i,j)}}{\mathbf{G}_{\mathrm{x}(i,j)}}\right). \tag{6}$$

Nakon pronalaska gradijenta, uz izraz (6) računa se kut θ kojeg je potrebno zaokružiti na jedan od četiri smjera u kojem piksel može imati susjeda.

Kut smjera ruba zaokružen je na jedan od četiri kuta koji predstavljaju okomitu, vodoravnu i dvije dijagonale($0°, 45°, 90°$ *i* $135°$) [3].

2.2.3. Ne-maksimalna supresija

Nakon dobivanja veličine i smjera gradijenta, vrši se potpuno skeniranje slike kako bi se uklonili svi neželjeni pikseli koji možda ne čine rub. Za to se na svakom pikselu provjerava da li je piksel lokalni maksimum u njegovom susjedstvu u smjeru gradijenta [4]. Slika 2. prikazuje primjer ne-maksimalne supresije.

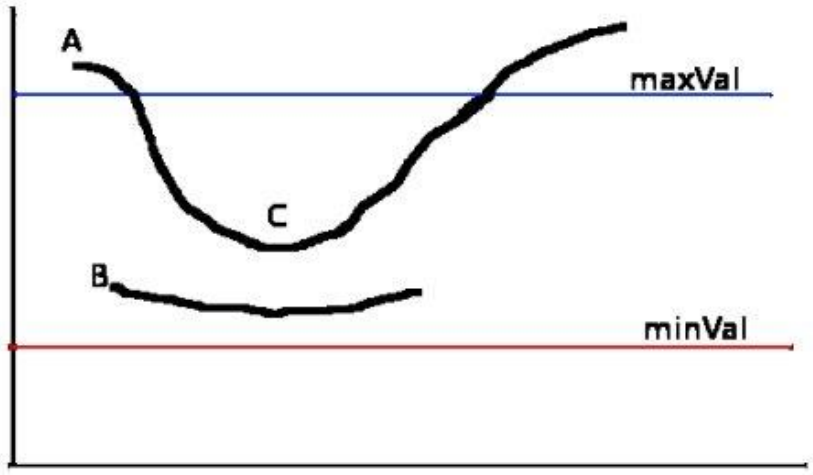


Slika 2. Ne-maksimalna supresija piksela [4]

Na slici 2. može se vidjeti da je točka A na rubu (u okomitom smjeru). Smjer gradijenta normalan je na rub. Točke B i C su u smjerovima gradijenta. Tako se točka A provjerava s točkom B i C da se vidi čini li lokalni maksimum. Ako je tako, uzima se u obzir za sljedeću fazu, u protivnom se potiskuje (stavlja na nulu). Ukratko, rezultat koji se dobiva je binarna slika s "tankim rubovima" [4].

2.2.4. Određivanje praga histerezom

Ova faza odlučuje koji su sve rubovi jaki rubovi, a koji nisu. Za ovu fazu potrebne su dvije vrijednosti praga, minVal i maxVal. Svi rubovi s gradijentom intenziteta većim od maxVal su jaki rubovi, a oni ispod minVal sigurno su slabi rubovi, pa se odbacuju. Oni koji se nalaze između ova dva praga klasificirani su kao rubovi ili nerubovi na temelju njihove povezanosti. Ako su spojeni na piksele jakog ruba, smatraju se dijelom rubova [4]. Slika 3. prikazuje primjer određivanja slabih i jakih rubova.



Slika 3. Određivanje jakih i slabih rubova [4]

Na slici 3. može se vidjeti da je rub A je iznad maxVal, pa se smatra jakim rubom Iako je rub C ispod maxVal, povezan je s rubom A, tako da se i on smatra valjanim rubom i dobivamo tu punu krivulju. Ali rub B, iako je iznad minVal i nalazi se u istom području kao rub C, nije povezan ni s jednim jakim rubom, tako da se to odbacuje. Stoga je vrlo važno odabrati minVal i maxVal u skladu s tim kako bismo dobili točan rezultat [4].

U ovom radu vrijednosti jakih i slabih rubova određuju se na početku rada aplikacije kako bi se algoritam detekcije rubova prilagodio za različite uvjete osvjetljenja. Za rad algoritma bitno je dobro osvjetljenje sustava, kako ne bi došlo do pogrešne detekcije, te kako bi sustav bio robustan na vanjske poremećaje u osvjetljenju.

2.3. Definicija kontura i rubova

Potrebno je razumjeti razliku između rubova, granica i kontura. Rubovi predstavljaju diskontinuitete u svjetlini ili boji. Granice predstavljaju sjecište dviju površina različite svjetline ili boje. Konture se mogu objasniti kao spoj svih točaka iste boje ili intenziteta, koje spajaju kontinuiranu krivulju duž granice. Konture su važan alat za analizu i detekciju objekta na slici, te su često korištene u industriji. U algoritmima računalnog vida, konture se stvaraju pomoću rubova i granica.

2.4. Korištene aplikacije i biblioteke

2.4.1. Python

Python je interpretirani, objektno orijentirani programski jezik visoke razine s dinamičkom semantikom. Njegove visoke razine ugrađenih podatkovnih struktura, u kombinaciji s dinamičkim pisanjem i dinamičkim vezanjem, čine ga vrlo atraktivnim za brzi razvoj aplikacija. Pythonova jednostavna sintaksa koju je lako naučiti naglašava čitljivost i stoga smanjuje troškove održavanja programa. Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Python tumač i opsežna standardna biblioteka dostupni su u izvornom ili binarnom obliku bez naknade za sve glavne platforme i mogu se besplatno distribuirati [5].

U ovom radu korišten je Python verzije 3.6., zbog kompatibilnosti s korištenim bibliotekama. Također, kreiran je novi virtualni okoliš, koji koristi samo najpotrebnije biblioteke, kako bi Python tumač (*eng. Interpreter*) i program radili većom brzinom.

2.4.2. Korištene biblioteke

2.4.2.1. OpenCV

OpenCV (Open Source Computer Vision) biblioteka je softvera za računalni vid i strojno učenje otvorenog koda. OpenCV je napravljen kako bi osigurao zajedničku infrastrukturu za aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima. Knjižnica ima više od 2500 optimiziranih algoritama, što uključuje opsežan skup klasičnih i najsuvremenijih algoritama računalnog vida i strojnog učenja. Ovi se algoritmi mogu koristiti za otkrivanje i prepoznavanje lica, identificiranje objekata, klasificiranje ljudskih radnji u videozapisima, praćenje pokreta kamere, praćenje pokretnih objekata, izdvajanje 3D modela objekata, stvaranje 3D oblaka točaka iz stereo kamera, spajanje, pronalaženje sličnih slika iz baze podataka slika, uklanjanje crvenih očiju sa slika snimljenih bljeskalicom, praćenje pokreta očiju, prepoznavanje krajolika i postavljanje oznaka za prekrivanje s proširenom stvarnošću itd. [6].

OpenCV biblioteka odlična je za početak s radom u području računalnog vida, kao i za kompleksnije operacije s računalnim vidom komponirane s algoritmima umjetne inteligencije. U ovom radu koriste se osnovne i jednostavne funkcije za detekciju i prepoznavanje kontura. Sintaksa je jednostavna, te je dokumentacija na internetu dobro objašnjena i popraćena slikama i primjerima.

2.4.2.2. NumPy

NumPy je temeljni paket za znanstveno računalstvo u Pythonu. To je Python biblioteka koja pruža višedimenzionalne objekte niza, razne izvedene objekte (kao što su maskirani nizovi i matrice) i asortiman rutina za brze operacije na nizovima, uključujući matematičke, logičke, manipulaciju oblikom, sortiranje, odabir, diskretne Fourierove transformacije, osnovna linearna algebra, osnovne statističke operacije, slučajna simulacija i još mnogo toga [7].

NumPy je jedan od najpopularnijih biblioteka za rad s nizovima i vise dimenzijskim poljima. OpenCV biblioteka zahtjeva NumPy biblioteku za rad algoritma, pošto se slike opisuju s više dimenzijskim poljima, te su potrebni kompleksni proračuni tih polja.

2.4.2.3. Time

Time biblioteka koristi se za pristupanje i konverziju vremena. Također, biblioteka može stvarati vremenska kašnjenja, koja su razlog zašto se ta biblioteka koristi u ovom radu.

2.4.2.4. pySerial

Biblioteka pySerial omogućava pristup i rad s kanalima serijske komunikacije, kao i slanje podataka preko istih. Također, omogućava pristup postavkama kanala serijske komunikacije. Serijska komunikacija bitna je kako bi se informacije s Pythona poslale na Arduino mikrokontroler. Više o serijskoj komunikaciji govorit će se u kasnijim poglavljima.

2.5. Kamera korištena u radu

Kamera korištena u ovom radu je Logitech C270. Karakteristike kamere su sljedeće:

- Maksimalna rezolucija: 1280x720 piksela
- Broj sličica u sekundi pri maksimalnoj rezoluciji: 30 fps
- Vrsta fokusa: Fiksni fokus
- Tip leće: plastična
- Kut dijagonale vidnog polja: 55° [8].

Slika 4. Logitech C270 [8]

Iako je kamera potrošačkog tipa i namijenjena je za video pozive, nudi nisku rezoluciju i malen broj sličica u sekundi, dovoljno je dobra za ovaj rad. Velika rezolucija nije potrebna jer se radi s običnom detekcijom kontura, gdje nije potrebna prevelika preciznost. Kod primjene kamere za mjerenja i rješavanje problema distorzije slike, kamera bolje kvalitete bila bi prikladnija. Broj sličica u sekundi je zadovoljavajući, jer je brzina pokretne trake vrlo mala, a i sam broj sličica u sekundi ovisi o brzini izvršavanja Python koda.

3. ANALIZA PYTHON KODA I REZULTATI

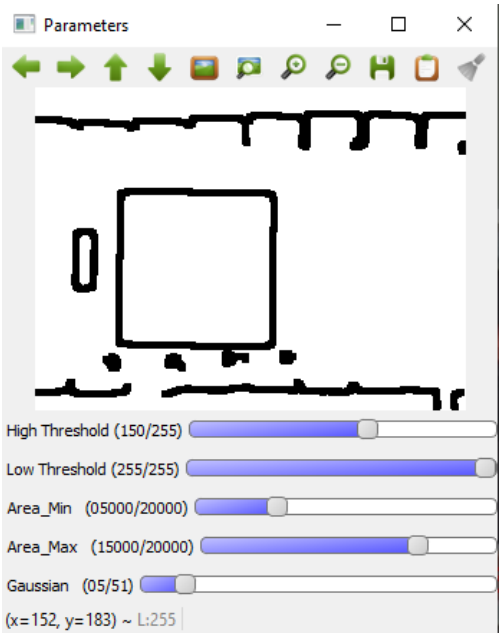
3.1. Priprema slike i programa

Kako bi se koristile funkcije programskih biblioteka, prvo se moraju pozvati u programu. Korištene biblioteke su OpenCV, pySerial, Time i NumPy. U početku se definiraju varijable kao što su rezolucija, koordinate za izrezak glavne slike, početne vrijednosti brojača i statusne varijable. Pomoću funkcije serial.Serial() definiraju se postavke kanala i pokreće se serijska komunikacija. Oznaka kanala mora biti ista kao i kanal na koji je spojen Arduino mikrokontroler, te brzina prijenosa podataka(*eng. Baudrate*), mora biti iste vrijednosti kao što je deklariran u kodu Arduino mikrokontrolera. Funkcija cv2.VideoCapture() služi za učitavanje video datoteka, sekvenci slika ili dohvaća slike s web kamere. Pomoću funkcije cv2.VideoCapture.set() postavljaju se postavke rezolucije i broja sličica u sekundi.

Deklarirana je funkcija koja služi za slanje podataka preko serijske komunikacije. Funkcija prima argumente broja rubova kontura, te kad je pozvana, šalje tu informaciju na Arduino mikrokontroler.

3.2. Prozor s klizačima za konfiguraciju parametra sustava

Zbog promjenjivih uvjeta osvjetljenja i same konstrukcije pokretne trake, definiran je prozor sa klizačima. Pomoću funkcije cv2.createTrackbar() stvaraju se klizači na željenom prozoru, te se definiraju njihove gornje i donje granične vrijednosti i nazivi klizača. Stvoreni su klizači za maksimalnu i minimalnu vrijednost površina kontura, kako bi lakše filtrirali željene predmete od ostatka slike. Slika 5. prikazuje prozor s klizačima.



Slika 5. Prozor s klizačima parametra sustava

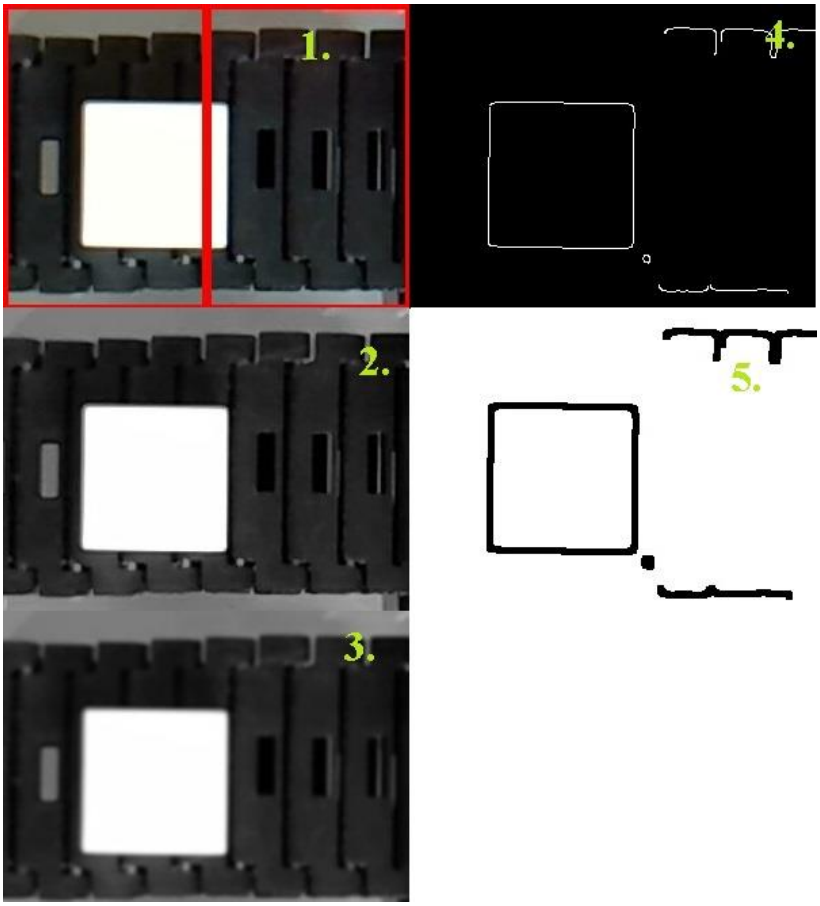
Zbog konstrukcije pokretne trake, javlja se puno šuma i program detektira neželjene rubove i konture. Također, stvoreni su klizači za definiranje gornje i donje granice praga gradijenta intenziteta Cannyevog algoritma. Zadnji klizač je za veličinu matrice Gaussovog filtra. Pomoću stvorenih klizača, jednostavno se mogu namjestiti parametri pripreme slike za detekciju oblika, pogotovo kod promjene osvjetljenja u okolini i promjene veličine oblika za sortiranje na pokretnoj traci.

Nakon definiranja varijabli i klizača, program počinje s while() petljom. While() petlja s argumentom istina (*eng. True*) ponavljati će se beskonačno. Tom petljom osvježava se slika i dobiva se prikaz s web kamere u realnom vremenu. Za prekid petlje definirana je tipka na kraju koda. Prekidom petlje se gase svi prozori, zaustavlja prijenos s web kamere i gasi se kanal serijske komunikacije prema Arduino mikrokontroleru. Unutar while() petlje definirane su vrijednosti s klizača kako bi se osvježavale svaki put kad se petlja ponovi.

3.3. Priprema slike i prepoznavanje rubova

Kako bi se izbjegle dodatne pogrešne detekcije i smanjio broj nepotrebnih informacija, izrezan je dio slike s kamere koji obuhvaća samo dio pokretne trake, umjesto cijelog sustava. Slika s kamere pretvara se u sivu sliku pomoću funkcije cv2.cvtColor(), kako bi se smanjio broj informacija i povećala brzina programa. Potom se na sivu sliku primjenjuje Gaussov filter s

ulaznim argumentom veličine matrice Gaussovog filtra koja je upravljana preko prijašnje definiranih klizača. Filtrirana slika onda ulazi u funkciju kao argument za Cannyev algoritam detekcije rubova pomoću funkcije cv2.Canny(). Ostali argumenti funkcije cv2.Canny() su vrijednosti gornje i donje granice praga histereze. Sliku rezultata Cannyevog algoritma prosljeđujemo u funkciju cv2.dilate() kako bi se podebljali detektirani rubovi. Slika 6. prikazuje korake pripreme slike s različitim filtrima i algoritmima.

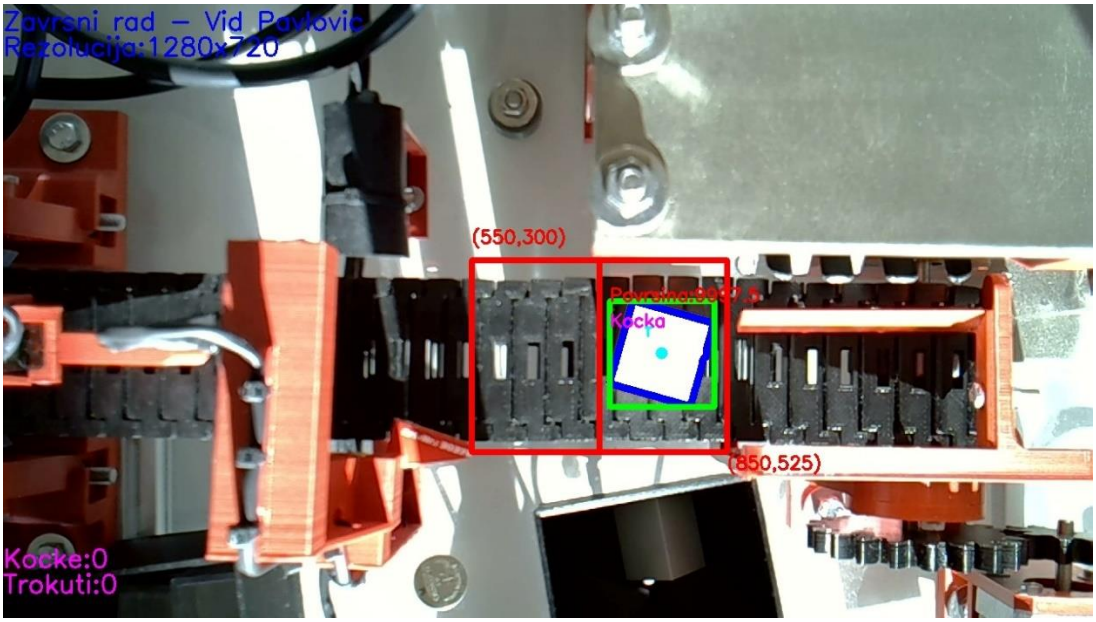


Slika 6. Koraci pripreme slike

Na slici 6. mogu se vidjeti prijašnje objašnjeni koraci pripreme slike. Dio slike označen s brojem 1. prikazuje izrezani dio glavnog ekrana, koji ne sadrži nijedan primijenjen filter. Dio slike označen s brojem 2. prikazuje sivu sliku. Odmah poslije, s oznakom 3., može se vidjeti slika s primijenjenim Gaussovim filtrom. Dio slike s oznakom 4. prikazuje sliku nakon Cannyevog algoritma za detekciju rubova. Na zadnjem dijelu slike može se vidjeti zadnji korak pripreme slike, a to je slika s podebljanim rubovima. U programu napravljen je inverz slike s podebljanim rubovima, čime se postiže da pozadina bude bijela, a rubovi crni, te se olakšava postupak namještanja parametra.

Na glavni ekran stavljene su informacije kao što je rezolucija, stanje brojača pojedinih predmeta, granice izrezane slike i njene koordinate. Tekst se stavlja na sliku pomoću funkcije cv2.putText(), dok se linije i kvadrati crtaju pomoću funkcija cv2.line() i cv2.rectangle(). U navedenim funkcijama ulazni argumenti moraju biti slike i/ili prijenosi s web kamere na kojima se želi staviti tekst, kao i koordinate na kojima će se postaviti u pikselima.

Ishodište koordinatnog sustava u OpenCV biblioteci nalazi se u gornjem lijevom kutu, gdje je horizontalna os usmjerena prema gornjem desnom kutu, a vertikalna os usmjerena prema donjem lijevom kutu. Slika 7. prikazuje glavni prozor aplikacije.



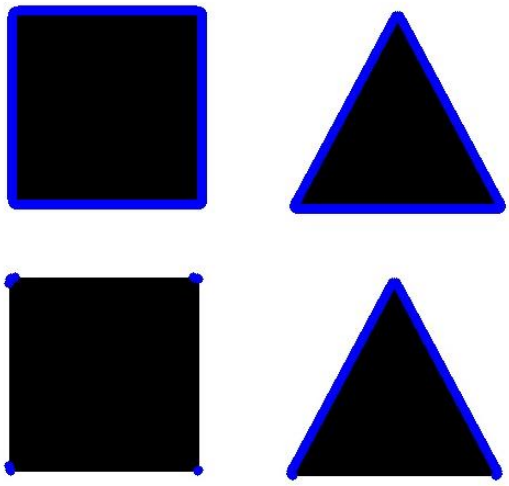
Slika 7. Glavni prozor

Radi jednostavnosti upotrebe, od svih prozora koji su koraci pripreme slike, prikazuju se samo glavni prozor, koji je sučelje za prikaz stanja sustava i informacija, te prozor s klizačima, u kojem također stoji slika s proširenim rubova.

3.4. Detekcija kontura

Kako bi program detektirao konture nakon primjene Cannyevog algoritma za detekciju rubova, koristi se funkcija cv2.findContours(). Funkcija prima tri argumenta. Prvi argument je slika s koje se detektiraju rubovi. Drugi argument je način pronalaska kontura, a treći argument je metoda aproksimacije kontura. Funkcija vraća popis kontura i njihovu hijerarhiju. Svaka kontura je Numpy niz koordinata graničnih točaka objekta u pikselima.

Kao prvi argument stavljena je slika koja je rezultat Cannyevog algoritma s podebljanim rubovima koja je rezultat funkcije cv2.dilate(). Kao drugi argument korištena je metoda cv2.RETR_EXTERNAL koja vraća samo vanjske konture. U ovom programu bitno je samo detektirati jednu konturu, a to je predmet na traci koji sortiramo, pa hijerarhija kontura nije toliko bitna. Treći argument je cv2.CHAIN_APPROX_NONE , koji vraća niz koordinata graničnih točaka kontura. Uz cv2.CHAIN_APPROX_NONE, koji vraća sve točke s rubova, postoji i argument cv2.CHAIN_APPROX_SIMPLE, koji vraća niz koordinata samo krajnjih točka rubova, pa je zbog manjeg broja točaka i koordinata algoritam brži. Zbog konstrukcije pokretne trake koja je izvedena kao lanac od karika napravljeni tehnologijom 3D printanja, javljaju se i rubovi tih karika, pa kako bi se lakše namjestili parametri postavka slike, cv2.CHAIN_APPROX_NONE je najbolji izbor za ovaj slučaj. Slika 8. prikazuje usporedbu dva argumenta.



Slika 8. Usporedba cv2.CHAIN_APPROX_NONE i cv2.CHAIN_APPROX_SIMPLE

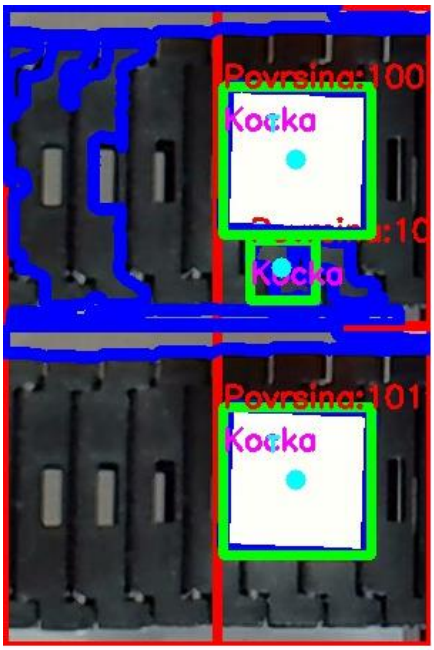
Na slici 8. može se vidjeti iscrtavanje detektiranih kontura argumentom cv2.CHAIN_APPROX_NONE (gore) i cv2.CHAIN_APPROX_SIMPLE (dolje).

Za svaku od pronađenih kontura se traži površina tih kontura s funkcijom cv2.contourArea(). Potom se ta površina uspoređuje sa minimalnim i maksimalnim vrijednostima površina koje su određene pomoću klizača. Time se uspješno filtriraju neželjene detektirane konture. Kada program pronađe konturu koja zadovoljava kriterij površine, kontura se iscrtava na ekranu pomoću funkcije cv2.drawContours(). Funkcija cv2.arcLength() služi za računanje opsega

kontura , te ovisno o argumentu funkcije, ona može računati opseg zatvorene ili otvorene konture. Ako je kontura zatvorena, funkcija vraća vrijednost faktora koji služi za aproksimaciju kontura. Aproksimacija kontura postiže se s funkcijom cv2.approxPolyDP(). Kao drugi argument funkcije stavlja se faktor točnosti aproksimacije, koji je jedak umnošku opsega konture i postotka duljine krivulje. Prvi argument je određena kontura, a treći argument određuje da li je kontura zatvorena ili otvorena.

Nakon aproksimacije konture, traže se koordinate točaka pravokutnog okvira koji omeđuje konturu s funkcijom cv2.boundingRect(). Veličina vektora nakon aproksimacije kontura određuje oblik kontura. Za trokut, vektor sadrži tri točke krajnjih rubova, dok za pravokutne oblike, vektor sadrži četiri točke. Tim veličinama dodatno možemo filtrirati nepoželjne detektirane konture.

Kad program detektira konturu oblika trokuta ili pravokutnika, na glavni prozor iscrtava se okvir oko te konture, zajedno s nazivom konture i površinom te konture. Slika 9. prikazuje primjer uspješne i neuspješne detekcije konture, kao i njihova iscrtavanja na glavnom prozoru.



Slika 9. Primjer uspješne i neuspješne detekcije

Na slici 9. (gore) može se vidjeti kako je program krivo detektirao neželjene konture. Razlog tome je detekcija rubova zbog konstrukcijske izvedbe pokretne trake. Nakon primjene uvjeta minimalne i maksimalne površine, program je detektirao manji broj kontura, koji će se dodatno

filtrirati u daljnjim koracima algoritma. Primjer uspješne detekcije može se vidjeti na slici 9. (dolje).

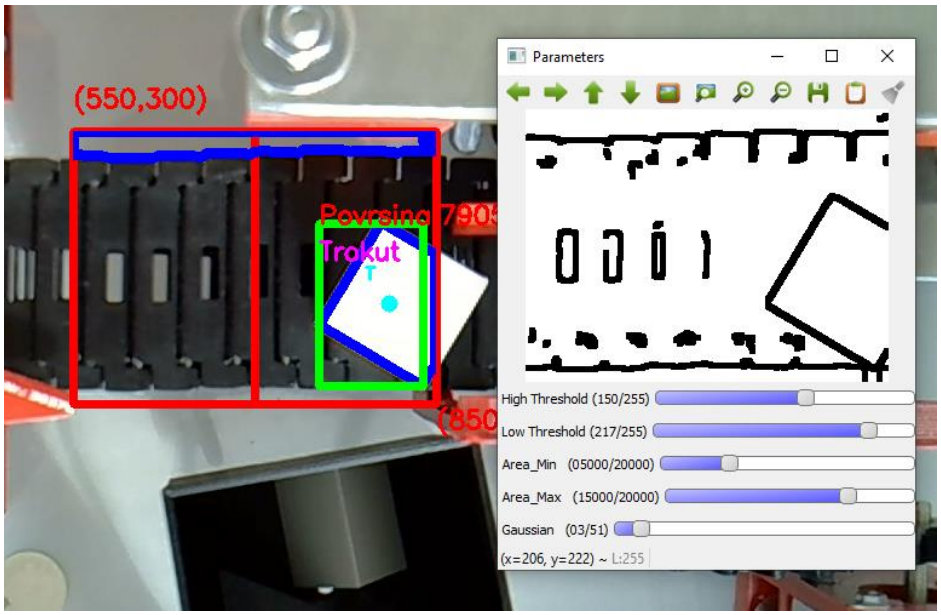
3.5. Brojač i odašiljanje podataka na Arduino mikrokontroler

Kako bi se pratio broj sortiranih predmeta, implementiran je brojač u Python-u. Time se izbjegava korištenje povratne informacije preko Arduino mikrokontrolera i rasterećuje se kanal za serijsku komunikaciju. Logika brojača se ostvaruje sa zamišljenim padajućim bridom, to jest, s promjenom varijable s većom vrijednošću na nižu.

U programu definirana je pomoćna varijabla naziva *status* koja se mijenja s obzirom na poziciju centroida konture. Centroid konture dobiva se s funkcijom cv2.moments(), koja vraća 24 različita momenta konture. Za ovaj slučaj, traže se koordinate centroida konture. S tim vrijednostima postavljaju se uvjeti za slanje podatka na Arduino mikrokontroler i inkrementiranje vrijednosti brojača. U sredini slike postavljena je vertikalna linija s poznatom horizontalnom koordinatom, koja se može uočiti na slici 7. unutar crvenog pravokutnika. S obzirom da se u orijentaciji kamere u ovom radu, pokretna traka kreće s desne na lijevu stranu, kad se detektira željena kontura desno od vertikalne linije, pomoćna varijabla *status* poprima vrijednost 1. Na prvu sličicu u kojoj je koordinata centorida konture lijevo od koordinate vertikale linije, pomoćna vrijednost postavlja se na vrijednost 0, i na tu promjenu poziva se funkcija za serijsku komunikaciju, koja šalje podatak o obliku predmeta. Također, inkrementira se stanje brojača predmeta. Ovisnost vrijednosti pomoćne varijable *status* za različite vrijednosti *x* koordinate centroida glasi:

$$status = \begin{cases} 1, & x_{\text{centroid}} > x_{\text{granica}} \\ 0, & x_{\text{centroid}} \leq x_{\text{granica}} \end{cases}, \tag{7}$$

gdje je x_{centroid} horizontalna koordinata centroida u koordinatnom sustavu OpenCV biblioteke, a x_{granica} je horizontalna koordinata vertikalne okidne linije u koordinatnom sustavu OpenCV biblioteke. Ovom metodom izbjegava se slanje podataka na svaku promjenu sličice, to jest svaku iteraciju while() petlje, što bi uzrokovalo bespotrebno opterećivanje kanala serijske komunikacije. Također, pošto je izrezan dio vidnog polja kamere, ako se na rubovima vidi samo pola konture, moglo bi doći do pogrešne detekcije konture. Slika 10. prikazuje još jedan primjer krive detekcije, koji je riješen primjenom logike padajućeg brida.



Slika 10. Primjer pogrešne detekcije

Na slici 10. može se vidjeti da je kontura koja se mora prepoznati zapravo pravokutnik, dok je algoritam prepoznao trokut. Razlog tome je što je izrezani dio zaslona zasebna slika, te ta slika u potpunosti ne uočava sve rubove i bridove pravokutnika. Kada bi se informacija o obliku konture odašiljala svaki put dok je kontura u vidom polju izrezanog dijela zaslona, došlo bi i do krivog sortiranja predmeta. Također, pošto pokretna traka nema sustav zatezanja, pri kretanju pokretne trake, dolazi do vibracija, što otežava algoritam detekcije. Ako algoritam ne bi prepoznao objekt u jednoj sličici desno od linije za okidanje, te bi ponovo prepoznao predmet lijevo od linije za okidanje, pomoću logike padajućeg brida, informacija o predmetu bi se i dalje proslijedila Arduino mikrokontroleru.

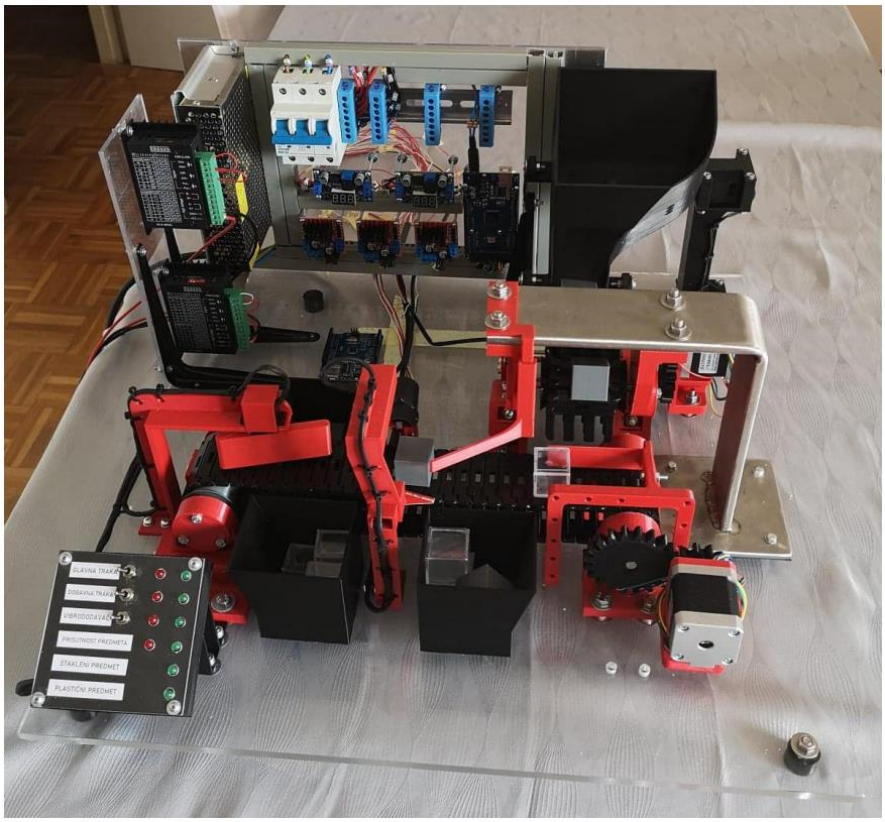
Pri zatvaranju programa, kod uvjeta za prekid while() petlje, zatvara se kanal za serijsku komunikaciju te prestaje komunikacija s web kamerom.

Prilog I. sadrži kod Python programa.

4. ELEKTROMEHANIČKI DIO SUSTAVA

4.1. Sustav pokretne trake

Cijeli sustav pokretne trake inicijalno je bio sustav za sortiranje prozirnih i neprozirnih predmeta, o kojem više u [9]. U ovom radu usredotočuje se samo na glavnu dobavnu traku, Arduino mikrokontroler, laserski prijemnik i odašiljač te servo motor za sortiranje predmeta. Jedine preinake u sustavu su promjena koda Arduino mikrokontrolera, dok neki dijelovi sustava nisu korišteni.

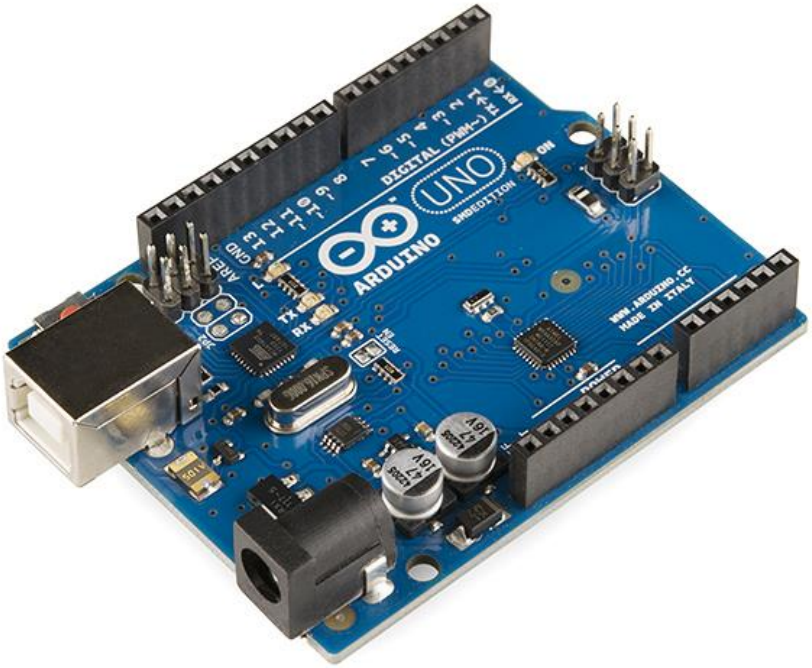


Slika 11. Sustav za sortiranje predmeta [9]

4.2. Arduino mikrokontroler

Arduino je hardverska i softverska tvrtka otvorenog koda, projekt i zajednica korisnika koja dizajnira i proizvodi jednopločne mikrokontrolere i komplete mikrokontrolera za izgradnju digitalnih uređaja. Dizajni Arduino ploča koriste razne mikroprocesore i kontrolere. Ploče su opremljene setovima digitalnih i analognih ulazno/izlaznih (I/O) pinova koji se mogu spojiti na različite ploče za proširenje ('štitovi') ili matične ploče (za izradu prototipova) i druge sklopove. Ploče imaju sučelje serijske komunikacije, uključujući univerzalnu serijsku sabirnicu (USB) na

nekim modelima, koja se također koriste za učitavanje programa. Mikrokontroleri mogu se programirati korištenjem programskih jezika C i C++, te korištenjem standardnog API-ja koji je također poznat kao Arduino programski jezik, inspiriran jezikom za obradu i korišten s modificiranom verzijom IDE-a za obradu. Uz korištenje tradicionalnih alatnih lanaca prevoditelja, projekt Arduino pruža integrirano razvojno okruženje (IDE) [10].



Slika 12. Arduino Uno mikrokontroler

Za upravljanje sustava korišten je Arduino Mega 2560 mikrokontroler, na koji su spojene gotovo sve komponente bitne za sortiranje predmeta. Arduino Uno mikrokontroler služi za upravljanje glavne dobavne trake. Elektronička shema sustava koja sadrži bitne dijelove za ovaj rad može se pronaći u Prilogu 3, kako bi se lakše referencirali ulazno-izlazni pinovi u programu Arduino mikrokontrolera.

4.3. Serijska komunikacija

Univerzalna serijska sabirnica (*eng. Universal Serial Bus,* USB) je vrsta komunikacije računala s vanjskim uređajima i periferijom. Podaci se razmjenjuju velikom brzinom, koja ovisi o verziji i standardu USB-a. USB je zamijenio prijašnja serijska i paralelna sučelja u računalima. Razmjena podataka je serijska, što znači da se podatak šalje sekvencijalno, bit po bit, nasuprot paralelne komunikacije, kod koje se podaci šalju paralelno u cjelini. Veličina podatka tipično iznosi 8 bitova. Cilj USB tehnologije je rasterećivanje glavne sabirnice računala od kartica za proširenje, kao i potrebe za ponovno pokretanje računala prilikom spajanja i odvajanja vanjskih uređaja.

Arduino Mega 2560 mikrokontroler sadrži USB komunikaciju verzije 2.0, te Type-B konektor. USB komunikacija verzije 2.0 omogućava brzine prijenosa i do 480 Mbit/s, te se napaja preko 5V izvora napona. Tablica 1. prikazuje brzine prijenosa za različite verzije standarda USB komunikacije.

Tablica 1. Brzine prijenosa po verzijama USB standarda

Brzina prijenosa [Mbit/s]	Podržano od verzije
1,5	USB 1.0
12	USB 1.1
480	USB 2.0
4800	USB 3.0

5. ANALIZA ARDUINO PROGRAMA I REZULTATI

5.1. Korištene biblioteke

5.1.1. Servo

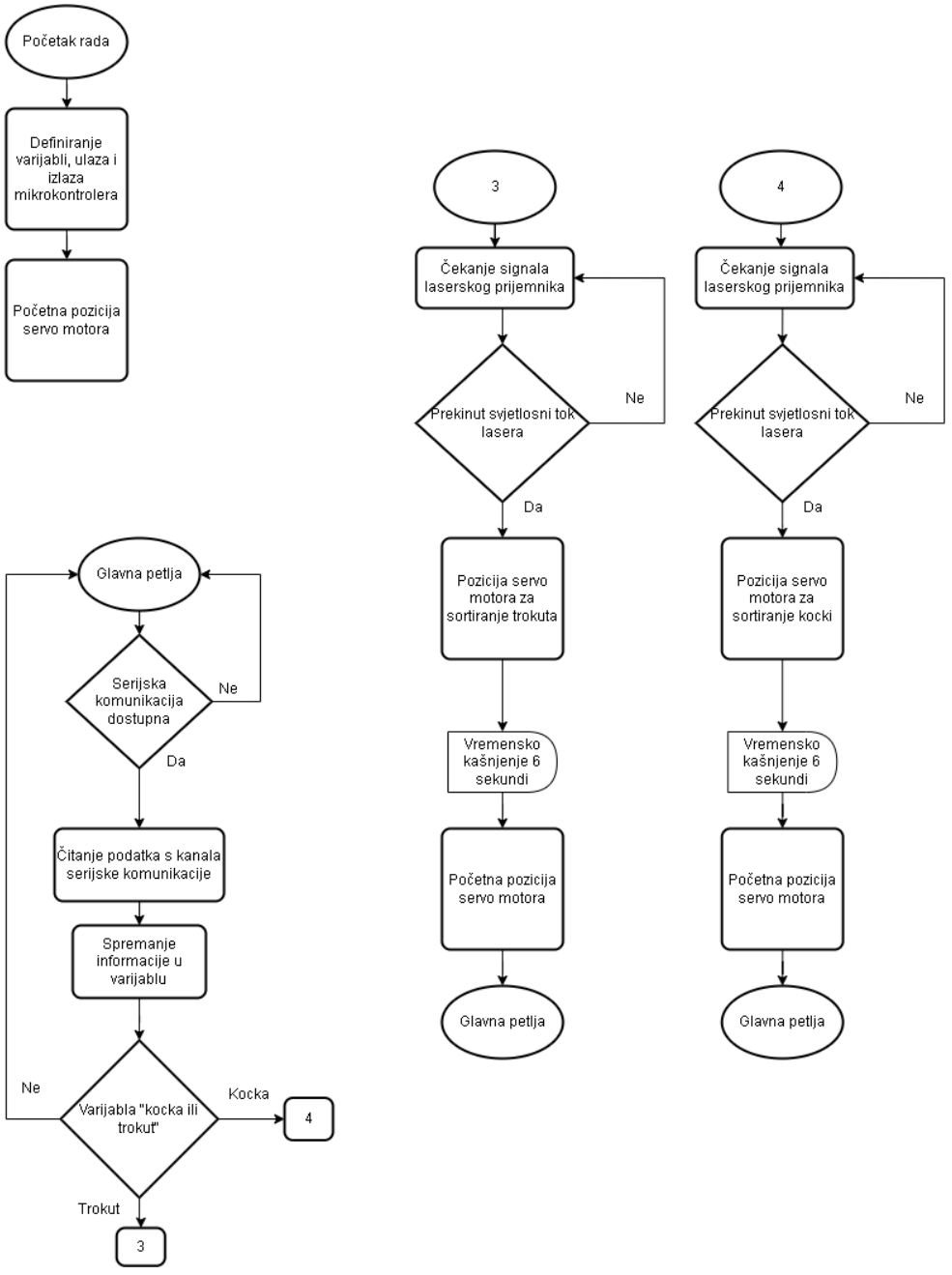
Servo biblioteka omogućava Arduino mikrokontrolerima da upravljaju servo motorima hobističke namjene. Biblioteka olakšava pozicioniranje standardnih servo motora kod kojih se upravlja kut zakreta i servo motora kod kojih se upravlja brzinom vrtnje.

5.2. Program Arduino Mega 2560 mikrokontrolera

Na samom početku programa potrebno je definirati potrebne varijable, pinove i pozvati korištene biblioteke. U početku programa pozvana je biblioteka Servo.h za upravljanje servo motorima. Deklarirana je pomoćna varijabla potrebna za serijsku komunikaciju. Također, deklarirane su varijable koje u sebi pohranjuju broj oznake ulazno-izlaznih pinova, radi lakšeg snalaženja u kodu. U glavnoj funkciji setup(), definiraju se korišteni pinovi te njihov režim rada. Pomoću funkcije iz biblioteke za upravljanje servo motorima, deklariran je pin na koji je spojen servo motor. Pin za laserski odašiljač definiran je kao izlazni pin, te se u početku stavlja u stanje visokog napona. Pin za laserski prijemnik definiran je kao ulazni pin. Inicijalizirana je serijska komunikacija s brzinom prijenosa informacija istog iznosa kao i u Python kodu. Deklarirane su funkcije kocka() i trokut(). Ovisno o podatku koji Arduino mikrokontroler primi s računala, pokretati će se jedna od tih funkcija za ispravno sortiranje predmeta.

U glavnoj funkciji loop(), koja se ponavlja beskonačno, stoji glavni dio koda za upravljanje sustavom. Funkcija počinje sa if() uvjetom, koji provjerava da li je kanal za serijsku komunikaciju zauzet. Ako bi kanal za serijsku komunikaciju bio zauzet, Arduino mikrokontroler ne bi bio u mogućnosti primati informacije s računala. Provjera stanja serijske komunikacije dobiva se pomoću funkcije Serial.available(), koja vraća dvije vrijednosti, istina ili laž. Ako je uvjet dostupnosti serijske komunikacije zadovoljen, Arduino mikrokontroler sve informacije iz kanala serijske komunikacije sprema u pomoćnu varijablu. Nakon toga slijede dva if() uvjeta. Ako pomoćna varijabla sadrži broj 4 , koji je doznaka da se radi o pravokutnom obliku, onda program zove funkciju kocka(). U toj funkciji, čeka se prekid toka svjetlosti između laserskog odašiljača i prijemnika. Potom, servo motor postavlja se na definirani otklon, te se uključuje unutarnja LE dioda, koja služi samo kao informacija da je u tijeku postupak sortiranja. Potom se stvara vremensko kašnjenje od 6000 ms pomoću funkcije delay(), kako bi predmet imao vremena doći do servo motora, te klizati pod kutem zakrilca dok ne upadne u

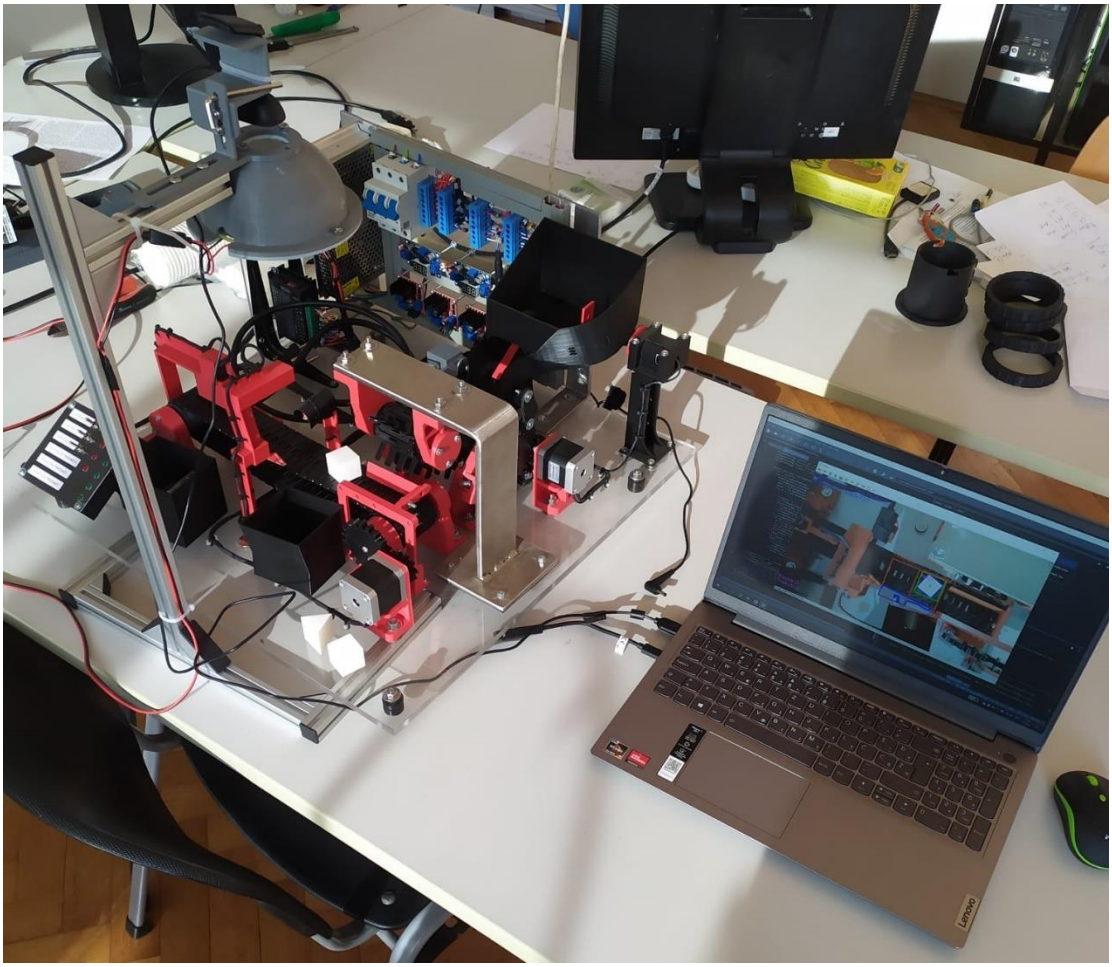
predviđenu kutiju. Nakon toga, koračni motor postavlja se na inicijalni otklon, isključuje se unutarnja LE dioda i brišu se podaci iz pomoćne varijable. Potom funkcija loop() ponovno kreće ispočetka. Drugi if() uvjet u glavnoj petlji loop() glasi, ako pomoćna varijabla sadrži broj 3, poziva se funkcija trokut(), koja ima istu strukturu koda kao i funkcija kocka(). Jedina razlika je otklon koračnog motora. Slika 13. prikazuje dijagram rada mikrokontrolera.



Slika 13. Dijagram toka rada Arduino mikrokontrolera

Na slici 13. može se vidjeti da cijeli sustav radi slijedno, te ako jedan od uvjeta nije ispunjen, program se ponovno ponavlja.

Nedostatak povratne veze mogao bi utjecati na neke dijelove sustava, na primjer, ako bi predmetu trebalo duže od 6 sekundi da padne u predviđenu kutiju, servo motor mogao bi predmet izbaciti s pokretne trake. Prikladno rješenje za taj problem bilo bi da se iznad kutija za predmete postavi ultrazvučni senzor za udaljenost, ili laserski odašiljač i prijemnik, koji bi detektirali da je predmet ubačen u kutiju. Slika 14. prikazuje cijeli sustav.



Slika 14. Prikaz cijelog sustava

6. KONSTRUKCIJA OSVJETLJENJA

Kako bi sustav bio prikladno osvjetljen, konstruirano je kupolno svjetlo koje eliminira odbijanje vanjskih zraka svijetlosti u kameru. Osvjetljenje je vrlo bitno u vizijskim sustavima, jer smanjuje mogućnost krive detekcije zbog utjecaja okoliša.

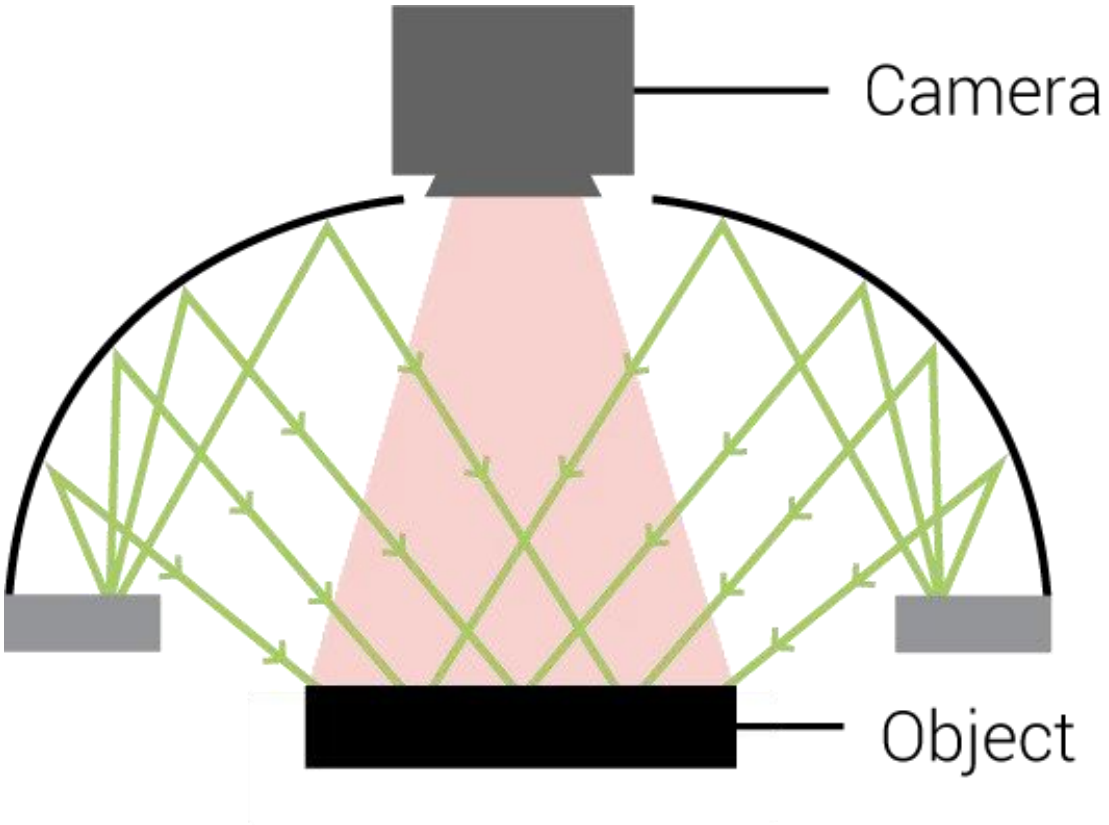
6.1. Nosač kamere i osvjetljenja

Konstruiran je nosač za kameru i osvjetljenje koristeći aluminijske profile tvrtke ITEM . Nosač se sastoji od aluminijskih profila dimenzije 20x20 mm rezanih na razne duljine. Sami profili su vrlo lagani, ali dovoljno čvrsti da drže kameru i osvjetljenje. Aluminijski profili spojeni su kutnicima, umjesto nekim drugim načinom pričvršćivanja, kako bi se povećala krutost samog nosača. Na krajeve profila spojeni su plastični poklopci kako ne bi došlo do ozljede prilikom rukovanja nosača. Nosač je konstruiran u programu Solidworks 2020. Slika 15. prikazuje model nosača iz programa Solidworks.

Slika 15. Model nosača

6.2. Difuzno kupolno svjetlo

Difuzno osvjetljenje u aplikacijama strojnog vida može se klasificirati kao potpuno osvjetljenje svijetlog polja za razliku od djelomičnog ili usmjerenog osvjetljenja svijetlog polja [11]. Kupolno svjetlo vrlo je korisno kod osvjetljavanja sjajnih i zakrivljenih površina. Objekti koji se detektiraju u ovom sustavu su izrađeni tehnologijom 3D printanja. Zbog toga, objekti, kao i sama pokretna traka, jako odbijaju svjetlost iz okolnih izvora svjetlosti. Temeljni princip kupolnog svjetla je da se svjetlost velike površine iz kupolnog svjetla odbija o zakrivljenu površinu, te da pada na površinu iz više kutova. Slika 16. prikazuje princip kupolnog svjetla.

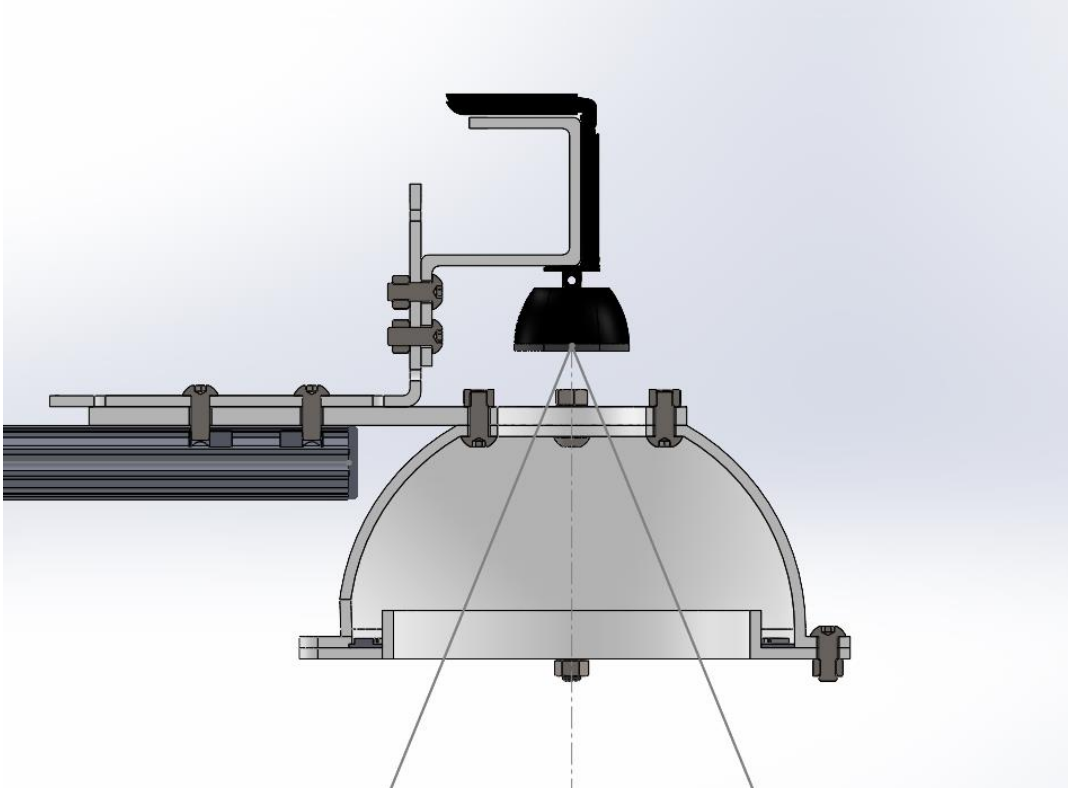


Slika 16. Princip kupolnog svjetla [11]

Upotrebom kupolnog svjetla postiže se uniformno osvjetljen predmet sa smanjenim utjecajem odbijanja okolne svjetlosti. Iza konstrukcije kupolnog svjetla stoje brojni proračuni odbijanja zraka izvora svjetlosti i zakrivljenosti kupole. Najveća pažnja posvećena u konstrukciji kupolnog svjetla je stavljena na kutu vidnog polja kamere, primjenjivosti svjetla na drugim vrstama kamera i povezivanje dijelova svjetla rastavljivim spojevima.

Kupolno svjetlo izrađeno je tehnologijom 3D printanja. Materijal od kojeg je izrađeno kupolno svjetlo je ABS . Svjetlo se sastoji od šest dijelova. Prvi i najvažniji dio je kupola. Unutarnja

zakrivljena površina kupole mora biti glatka, kako bi se svjetlost što bolje odbijala. Drugi zahtjev kod konstrukcije kupole je da unutarnja zakrivljena površina mora biti bijele boje, kako se svjetlost ne bi upila, nego odbila. Drugi dio kupole je poklopac kupole, koji drži LED prsten na mjestu. Poklopac je spojen s kupolom pomoću vijčanih spojeva. Treći dio kupole je LED prsten. LED prsten radi na naponu od 12 V, te troši struju od približno 1.5 A. LE diode na LED prstenu su hladno bijele boje. Četvrti dio kupole je nosač kupole. Nosač kupole preuzima težinu kupole, poklopca kupola i LED prstena, pa debljina nosača kupole iznosi 5 mm, dok su svi ostali dijelovi debljine 3 mm. Kupola je povezana s nosačem pomoću četiri M5 vijka norme ISO-7380. Vijci te norme imaju zakrivljenu glavu, pa su pogodi za spajanje kako ne bi došlo do stvaranja sjene, te kako bi se svjetlost odbijala o što veću površinu. Peti i šesti dio kupolnog svjetla služe za kameru. Konstruirani su s namjerom da se kamera može pomicati u dvije ravnine, kako bi se kamera s drugim vidnim poljem prilagodila kupolnom svjetlu. Prilog 4. sadrži tehničku dokumentaciju za kupolno svjetlo. Slika 17. prikazuje presjek modela kupolnog svjetla s naznačenim vidnim kutom kamere.



Slika 17. Presjek modela kupolnog svjetla

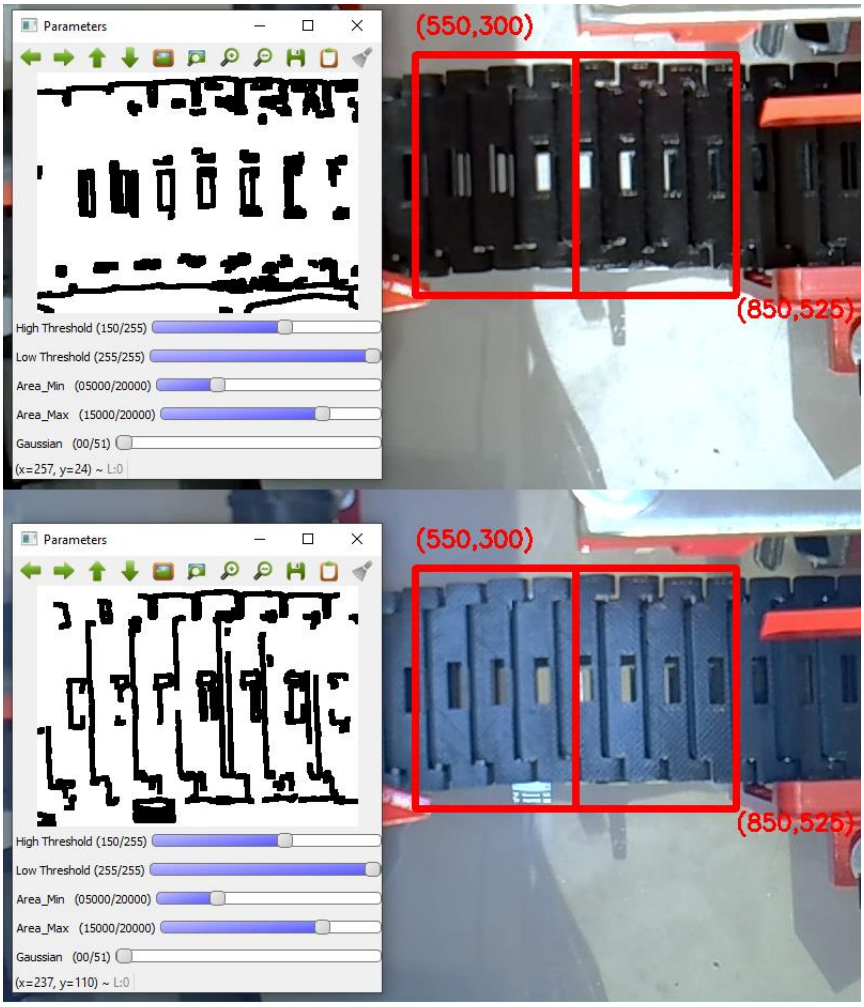
Na slici 17. može se vidjeti da ako dođe do zamjene kamere, te kamera ima drugačiji tip potpore, samo je potrebno prilagoditi držač za tu potporu, dok ostatak nosača ostaje nepromijenjen.

Slika 18. Nosač s kupolnim svjetlom i kamerom

Na slici 18. može se vidjeti stvarni nosač s osvjetljenjem. Vodiči povezani su s nosačem kako ne bi stvarali sjenu, te da ne ulaze u vidno polje kamere, kako ne bi došlo do pogrešne detekcije pri radu aplikacije. Vodiči za LED prsten spojeni su na prekidač kako bi se svjetlo moglo jednostavno paliti i gasiti. LED prsten napaja se preko 12 V ispravljača napona.

6.3. Usporedba osvjetljenja i rezultati

Kamere potrošačkog tipa rade loše u uvjetima niskog i visokog osvjetljenja, za razliku od industrijskih kamera. Slika 19. prikazuje glavni ekran i prozor slike nakon Cannyevog algoritma bez upotrebe kupolnog osvjetljenja (gore) i sa uključenim kupolnim svjetlom (dolje).



Slika 19. Usporedba slike s i bez primjene osvjetljenja

Na slici 19. može se vidjeti kako se pri uvjetima okolišnog svjetla zrake svjetlosti odbijaju na nekim dijelovima (gore), dok se sa kupolnim svjetlom dobije slika bez točkastog odbijanja svjetlosti.

7. ZAKLJUČAK

U ovom radu napravljena je aplikacija za detekciju i prepoznavanje kontura, te sortiranje predmeta tih kontura pomoću elektromehaničkog sustava pokretne trake upravljane Arduino mikrokontrolerom. U sklopu teorijske podloge, objašnjene su osnove računalnog vida, korištene aplikacije i biblioteke, osnove o Arduino mikrokontoleru i princip kupolnog svjetla. Najveći izazov u ovom radu bio je prepoznavanje oblika na pokretnoj traci. Primjenom objašnjenih algoritma i alata, predmeti su uspješno detektirani i prepoznati, te sustav ispravno sortira te predmete. Iako je glavni problem izdvajanje predmeta od pokretne trake koja nije ravna, može se vidjeti da se različitim metodama taj problem može riješiti. Konstruiranim osvjetljenjem riješen je problem odbijanja svjetlosti ili slučaja slabog okolnog osvjetljenja.

Detekcija kontura i prepoznavanje oblika važan je dio suvremene industrije i raznih pogona. Biblioteka OpenCV pruža jednostavan uvod u svijet računalnog i strojnog vida, te je jednostavno razumjeti i implementirati algoritme koje ta biblioteka pruža.

Arduino mikrokontroleri savršen su uvod u upravljačke sklopove i sustave. Uz veliku zajednicu korisnika i brojne module, moguće je napraviti jednostavne i napredne sklopove, kao i automatizirane sustave. Također, rad na mikrokontrolerima dobra je priprema za rad na industrijskim logičkim kontrolerima, to jest, PLC računalima.

Doduše, sustav ima par nedostataka. Jedan od nedostataka je što aplikacija za prepoznavanje i detekciju kontura jako ovisi o osvjetljenju sustava, te je ručno potrebno prilagoditi sustav pomoću definiranih klizača. Jedno od mogućih rješenja je napraviti izvor svjetlosti na sustavu koji će kameri i algoritmu stvarati referencu o intenzitetu svjetlosti, te bi se s time sustav mogao donekle sam kalibrirati.

Drugi problem je izvedba programa mikrokontrolera. Mikrokontroler prima informacije preko serijske komunikacije, te ovisno o tim informacijama, sekvencijalno izvršava naredbe. Problem bi se javio kad bi informacija preko serijske komunikacije došla dok sustav izvršava zadatak, pri čemu bi ju sustav „ignorirao“, i nastavio s izvršavanjem trenutne operacije. Taj problem mogao bi se riješiti primjenom hardverskih prekida(*eng. Interrupt*), gdje bi mikrokontroler zaustavio trenutni program, te pokrenuo rutinu tog prekida. Pošto je u ovom programu korištena serijska komunikacija između računala i mikrokontrolera, hardverski prekidi nisu jednostavno

izvedivi, osim ako bi se računalno spojilo na zaseban mikrokontoler, koji bi odašiljao signal u obliku visokog i niskog napona, te bi tako pokretao hardverski prekid na glavnom mikrokontroleru. Drugo rješenje problema je korištenje mikrokontolera koji sadrži dvije jezgre, pri čemu bi jedna jezgra služila za čitanje podatka s kanala serijske komunikacije, a druga jezgra izvršavala bi naredbe sustava. U tom slučaju sustav bi mogao imati više predmeta na pokretnoj traci, za razliku od trenutnog sustava. Vremenska kašnjenja kod Arduino mikrokontolera riješila bi se korištenjem millis() funkcija, te bi svakom zadatku mogli posvetiti jedan dio vremena. Informacije o nadolazećim predmetima spremale bi se u definiran niz u memoriji mikrokontrolera.

Također, jedan od problema bio bi hlađenje izvora svjetlosti kod kupolnog svjetla. U industrijskim primjenama gdje bi pogodi radili na duže vrijeme, izvor svjetlosti stvarao bi toplinski tok koji bi se učinkovito trebao disipirati u okolinu. Jedno od mogućih rješenja u ovom radu bila bi konstrukcija poklopca kupole, koji bi imao provrte kako bi se na što većoj površini sklop izvora svjetlosti hladio. U ovom slučaju, hlađenje nije potrebno jer sustav nije predviđen za rad na duže vrijeme, ni mehanički ni programski.

LITERATURA

[1] IBM: https://www.ibm.com/topics/computer-vision, (pristupljeno 07.01.2023.)

[2] Živkušić D., Primjena računalnog vida u kontroli kvalitete https://urn.nsk.hr/urn:nbn:hr:128:007595 , (pristupljeno 13.02.2023)

[3] Wikipedia: Canny edge detector, https://en.wikipedia.org/wiki/Canny_edge_detector, (pristupljeno 07.01.2023.)

[4] OpenCV: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html, (pristupljeno 07.01.2023.)

[5] Python: https://www.python.org/doc/essays/blurb/, (pristupljeno 07.01.2023.)

[6] OpenCV: https://opencv.org/about/, (pristupljeno 07.01.2023.)

[7] NumPy: https://numpy.org/doc/stable/, (pristupljeno 07.01.2023.)

[8] Logitech: https://www.logitech.com/en-us/products/webcams/c270-hd-webcam.960-000694.html, (pristupljeno 07.01.2023.)

[9] Jelenić D., Projektiranje i sklapanje umanjene laboratorijske makete transportne trake https://urn.nsk.hr/urn:nbn:hr:235:276246 , (pristupljeno 13.02.2023)

[10] Wikipedia: Arduino, https://en.wikipedia.org/wiki/Arduino, (pristupljeno 07.01.2023.)

[11] Advanced Illumination: https://www.advancedillumination.com/lighting-education/lighting-technique-diffuse-illumination/ , (pristupljeno 07.01.2023.)

PRILOZI

I. Python kod

```
#Pozivanje biblioteka
import cv2
import numpy as np
import serial
import time

#Definiranje varijabli
brojac_kocka = 0
brojac_trokut = 0
status = 0
#Rezolucija
frameWidth = 1280
frameHeight = 720
#Koordinate za izrezak s glavnog prozora
kockax1=550
kockay1=300
kockax2=850
kockay2=525
filterx1 = 700

#Inicijalizacija serijske komunikacije
arduino = serial.Serial(port='COM3', baudrate=9600, timeout=.1)

#Postavke kamere
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(5, 30)

#Fukncija za slanje podataka na Arduino mikrokontroler
def write_read(x):
    arduino.write(bytes(x, 'utf-8'))
```

```
    time.sleep(0.05)
    #data = arduino.readline()
    print("poslano:" + x)
    #return data


def empty(a):
    pass

#Definiranje klizača
cv2.namedWindow("Parameters")
cv2.resizeWindow("Parameters", 640, 240)
cv2.createTrackbar("High Threshold","Parameters",150,255,empty) #80
cv2.createTrackbar("Low Threshold","Parameters",255,255,empty) #43
cv2.createTrackbar("Area_Min","Parameters",5000,20000,empty)
cv2.createTrackbar("Area_Max","Parameters",15000,20000,empty)
cv2.createTrackbar("Gaussian","Parameters",0,51,empty) #14


while True:
    success, img = cap.read()

    #Izrezak s glavnom prozora
    izrezak = img[kockay1: kockay2, kockax1: kockax2]

    #Vrijednosti s klizača
    threshold1=cv2.getTrackbarPos("High Threshold", "Parameters")
    threshold2=cv2.getTrackbarPos("Low Threshold", "Parameters")
    area_min=cv2.getTrackbarPos("Area_Min", "Parameters")
    area_max=cv2.getTrackbarPos("Area_Max", "Parameters")

    #Veličina kernela za Gaussov filter
    gaussian=cv2.getTrackbarPos("Gaussian", "Parameters")
    if gaussian%2 == 0:
        gaussian = gaussian + 1


    #Priprema slike
    imgGray = cv2.cvtColor(izrezak,cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (gaussian, gaussian), 0)
    imgCanny = cv2.Canny(imgBlur,threshold1,threshold2)
    kernel = np.ones((5,5))
    imgDil = cv2.dilate(imgCanny, kernel, iterations=1)
    izrezakDil = 255-imgDil
```

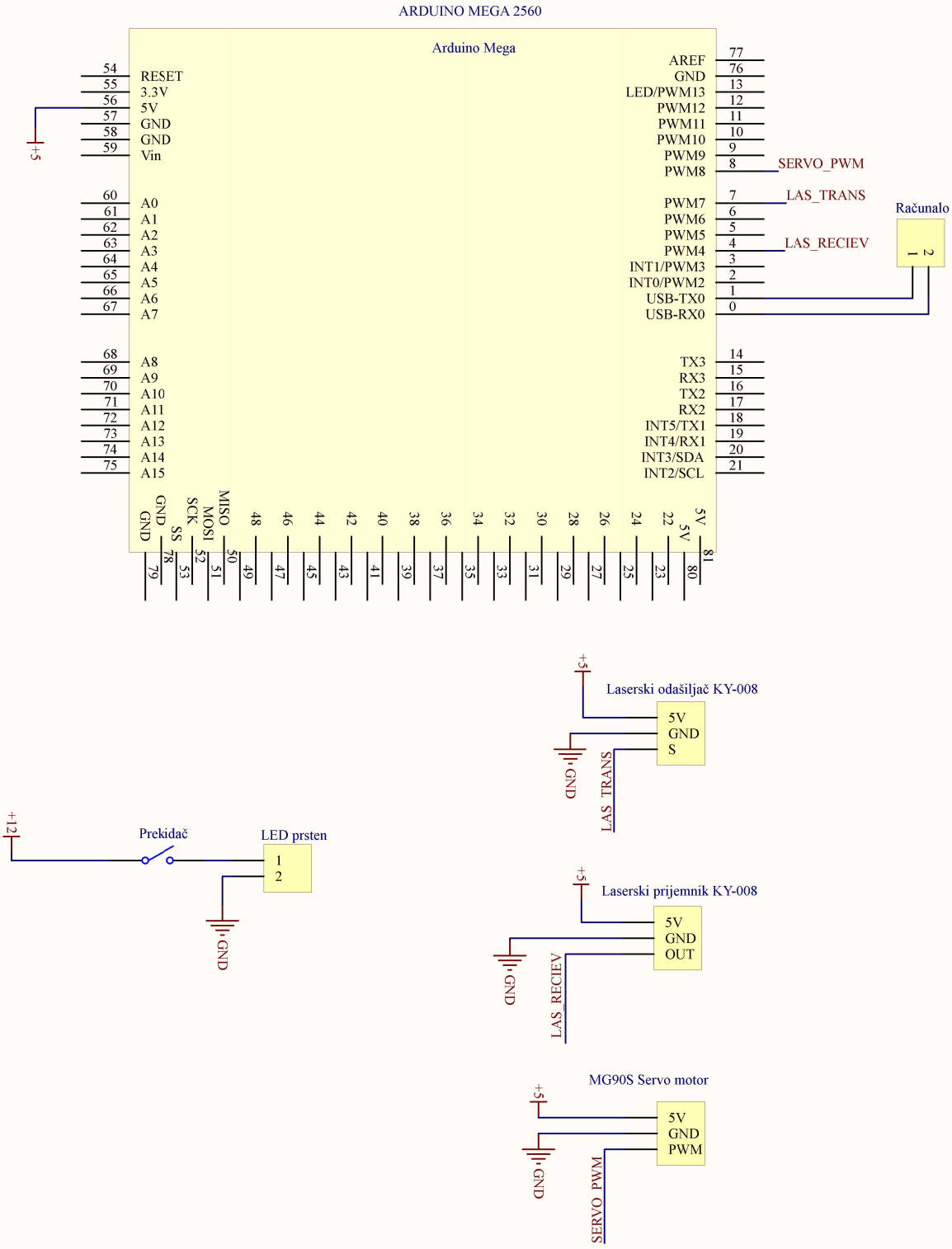
```
    #Generalne informacije za glavni zaslon
    cv2.putText(img,'('+str(kockax1)+','+str(kockay1)+')',(kockax1,kockay1-
20),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0, 0, 255),2)
    cv2.putText(img,'('+str(kockax2)+','+str(kockay2)+')',(kockax2,kockay2+20),cv2.FONT_H
ERSHEY_SIMPLEX,0.7,(0, 0, 255),2)
    cv2.rectangle(img,(kockax1,kockay1),(kockax2,kockay2),(0,0,255),5)
    #Okidna linija za brojač
    cv2.line(img,(filterx1,kockay1),(filterx1,kockay2),(0,0,255),5)
    cv2.putText(img,'Zavrsni rad - Vid Pavlovic',(0,30),cv2.FONT_HERSHEY_SIMPLEX,1,(
255, 0, 0),2)
    cv2.putText(img,'Rezolucija:' + str(frameWidth) + 'x' + str(frameHeight),(0,60),cv2.FONT
_HERSHEY_SIMPLEX,1,(255, 0, 0),2)
    cv2.putText(img,'Kocke:' + str(brojac_kocka),(0,frameHeight - 60),cv2.FONT_HERSHEY
_SIMPLEX,1,(255, 0, 255),2)
    cv2.putText(img,'Trokuti:' + str(brojac_trokut),(0,frameHeight - 30),cv2.FONT_HERSHE
Y_SIMPLEX,1,(255, 0, 255),2)

    #Detekcija kontura
    contours,hierarchy = cv2.findContours(izrezakDil,cv2.RETR_EXTERNAL,cv2.CHAIN_A
PPROX_NONE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area>area_min and area<area_max:
            #Aproksimacija kontura
            cv2.drawContours(izrezak, cnt, -1, (255, 0, 0), 7)
            peri = cv2.arcLength(cnt,True)
            approx = cv2.approxPolyDP(cnt,0.1*peri,True)
            x,y, w, h = cv2.boundingRect(approx)


            if len(approx) == 3 or len(approx) == 4:
                cv2.rectangle(izrezak,(x,y),(x+w,y+h),(0,255,0),5)
                cv2.putText(img, 'Povrsina:' + str(area), (kockax1 + x,kockay1 + y),cv2.FONT_HE
RSHEY_SIMPLEX,0.7,(0, 0, 255),2)
                if len(approx) == 4:
                    cv2.putText(img,'Kocka',(kockax1 + x,kockay1 + y + 30),cv2.FONT_HERSHEY
_SIMPLEX,0.7,(255, 0, 255),2)
                if len(approx) == 3:
                    cv2.putText(img,'Trokut',(kockax1 + x,kockay1 + y + 30),cv2.FONT_HERSHE
Y_SIMPLEX,0.7,(255, 0, 255),2)
                #Teziste centroida
                M = cv2.moments(cnt)
```

```
            if M['m00'] != 0:
                cx = int(M['m10']/M['m00'])
                cy = int(M['m01']/M['m00'])
                cv2.circle(izrezak, (cx, cy), 7, (255, 255, 0), -1)
                cv2.putText(izrezak, "T", (cx - 20, cy - 20),cv2.FONT_HERSHEY_SIMPLEX, 0
.5, (255, 255, 0), 2)
            #Logika brojača
            if (cx + kockax1 > filterx1):
                status = 1
            if (cx + kockax1 < filterx1) and (status == 1):
                status = 0
                write_read(str(len(approx)))
                if len(approx) == 4:
                    brojac_kocka = brojac_kocka + 1
                if len(approx) == 3:
                    brojac_trokut = brojac_trokut + 1


    #Prikaz slika
    cv2.imshow("Parameters", izrezakDil)
    cv2.imshow("Glavni", img)

    #Prekid aplikacije
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
arduino.close()
```

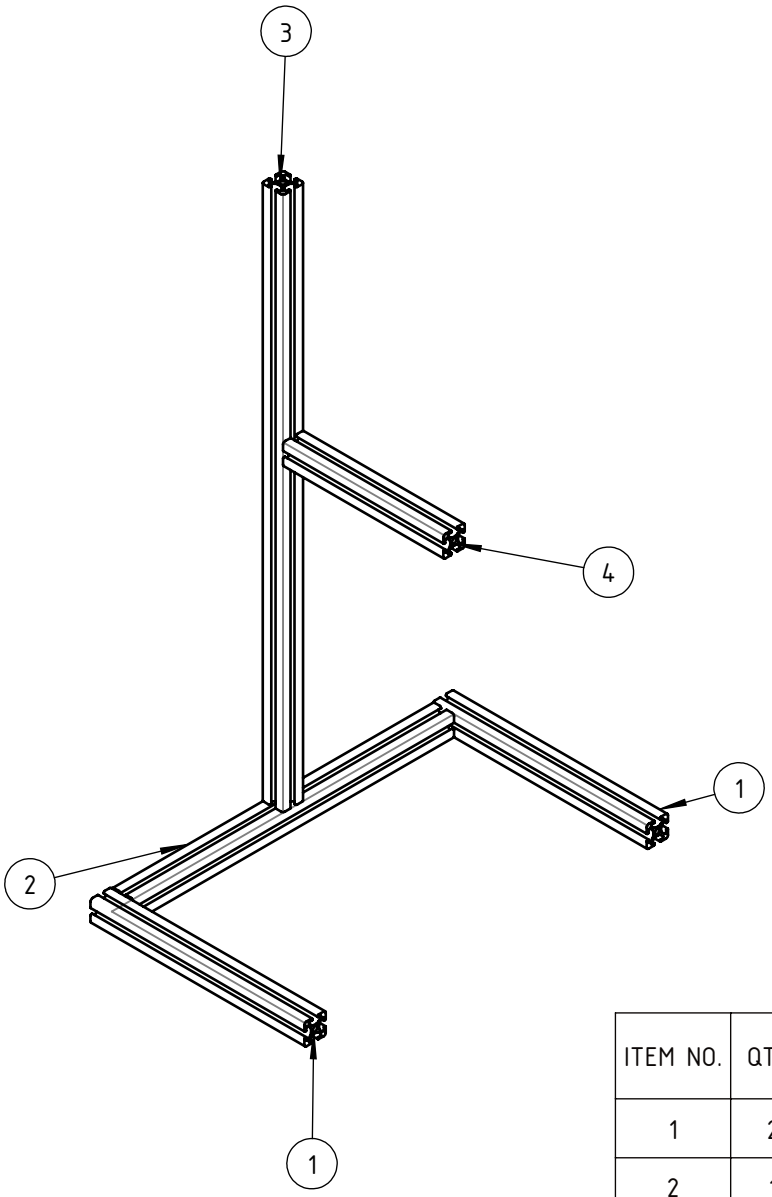
II. Arduino Mega 2560 kod

```
//Pozivanje biblioteka
#include <Servo.h>
Servo servo1;
//Definiranje varijabli
String python;
int laser = 7; //odašiljač
int dioda = 4; //prijemnik
int otklon = 50;//početna pozicija koračnog motora
void setup() {
//Definiranje pinova
servo1.attach(8);
servo1.write(50);

pinMode(LED_BUILTIN, OUTPUT);
pinMode(laser,OUTPUT);
pinMode(dioda,INPUT);
digitalWrite(laser,HIGH);

//Inicijalizacija serijske komunikacije
Serial.begin(9600);
Serial.setTimeout(1);
}

void loop() {
 if (!Serial.available()){
   python = Serial.readString();
   if ( python == "4" ) {
     digitalWrite(LED_BUILTIN, HIGH);
     kocka();;
   }
   if ( python == "3" ) {
     digitalWrite(LED_BUILTIN, HIGH);
     trokut();
   }
 }
}
//Funkcije
void kocka(){
 while (digitalRead(dioda) == HIGH){}
      servo1.write(120);
      delay(6000);
```

```
        servo1.write(50);
        digitalWrite(LED_BUILTIN, LOW);
        python = "0";
}
void trokut(){
        while (digitalRead(dioda) == HIGH){}
        servo1.write(10);
        delay(6000);
        servo1.write(50);
        digitalWrite(LED_BUILTIN, LOW);
        python = "0";
}
```



Naziv Prilog III - Elektronička shema i kupolno svjetlo	*Fakultet strojarstva i brodogradnje* *Sveučilište u Zagrebu* *Ivana Lučića 5* *10 000 Zagreb*	MiR
Student Vid Pavlović		
Datum: 03.02.2023		
ZAVRŠNI RAD		



ITEM NO.	QTY.	DESCRIPTION	LENGTH
1	2	Profile 5 20x20	200
2	1	Profile 5 20x20	300
3	1	Profile 5 20x20	500
4	1	Profile 5 20x20	150

	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

FSB Zagreb

Studij strojarstva

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Smjer: Mehatronika i robotika

Kopija

Materijal:

Masa:

ZAVRŠNI RAD

Mjerilo originala 1:5

Naziv: Profili nosača

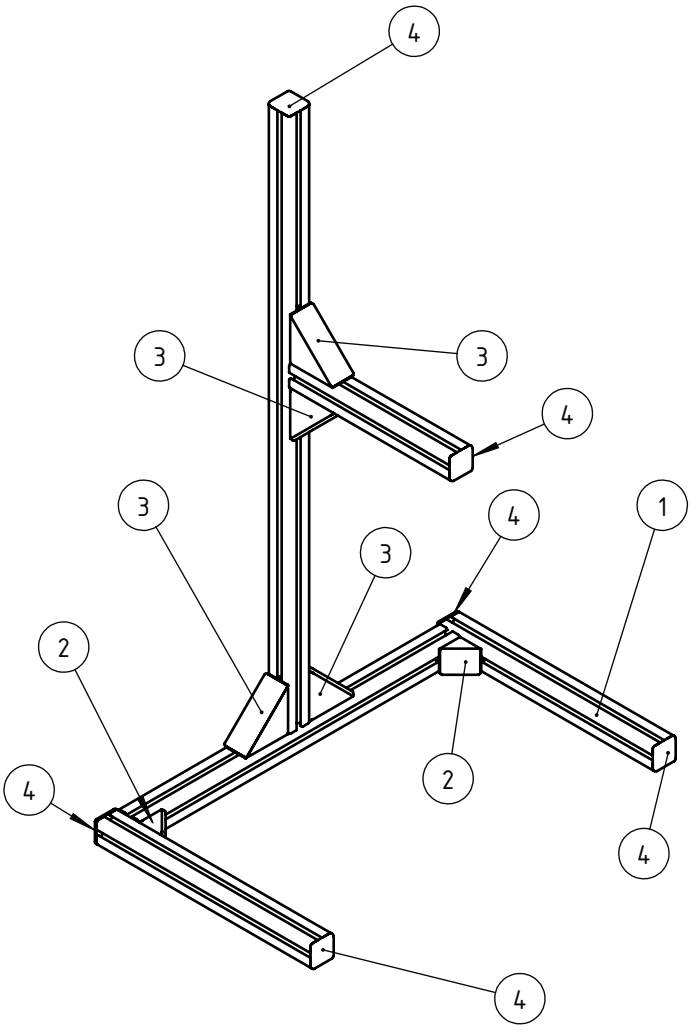
Pozicija: 1

Format: A4

Listova: 1

Crtež broj: 100-01-001

List: 1



ITEM NO.	PART NUMBER	QTY.
1	100-01-001	1
2	item 0042502 Angle Bracket Set 5 20x20 2	2
3	item 0042505 Angle Bracket Set 5 40x40 1	4
4	item 0037009 Cap 5 20x20 1	6

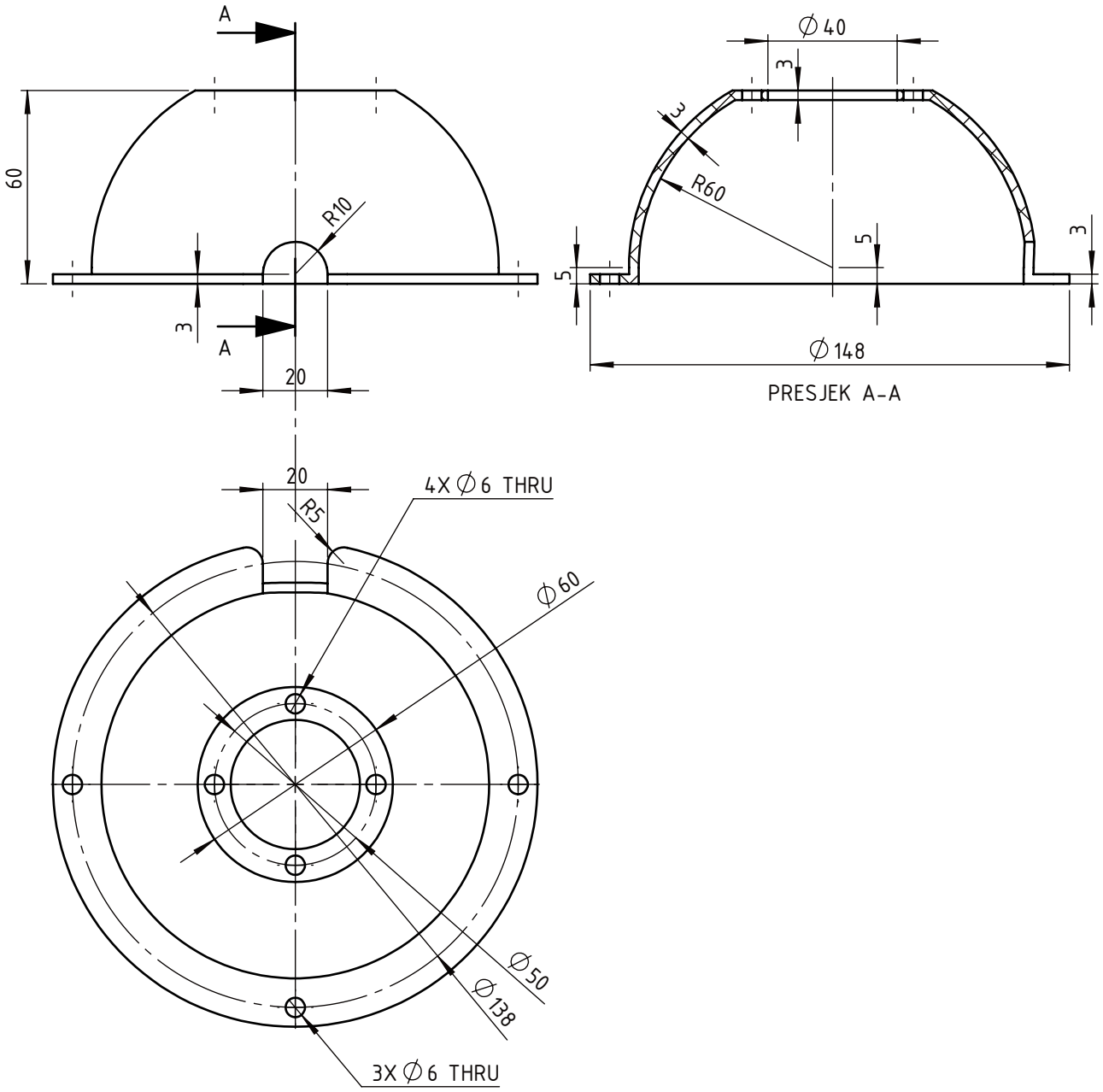
	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

FSB Zagreb

Studij strojarstva

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Smjer: Mehatronika i robotika
Materijal: / Masa:	ZAVRŠNI RAD
Mjerilo originala 1:5	Naziv: Nosač
	Pozicija: 1
Crtež broj: 100-01	Format: A4
	Listova: 1
	List: 1

Kopija

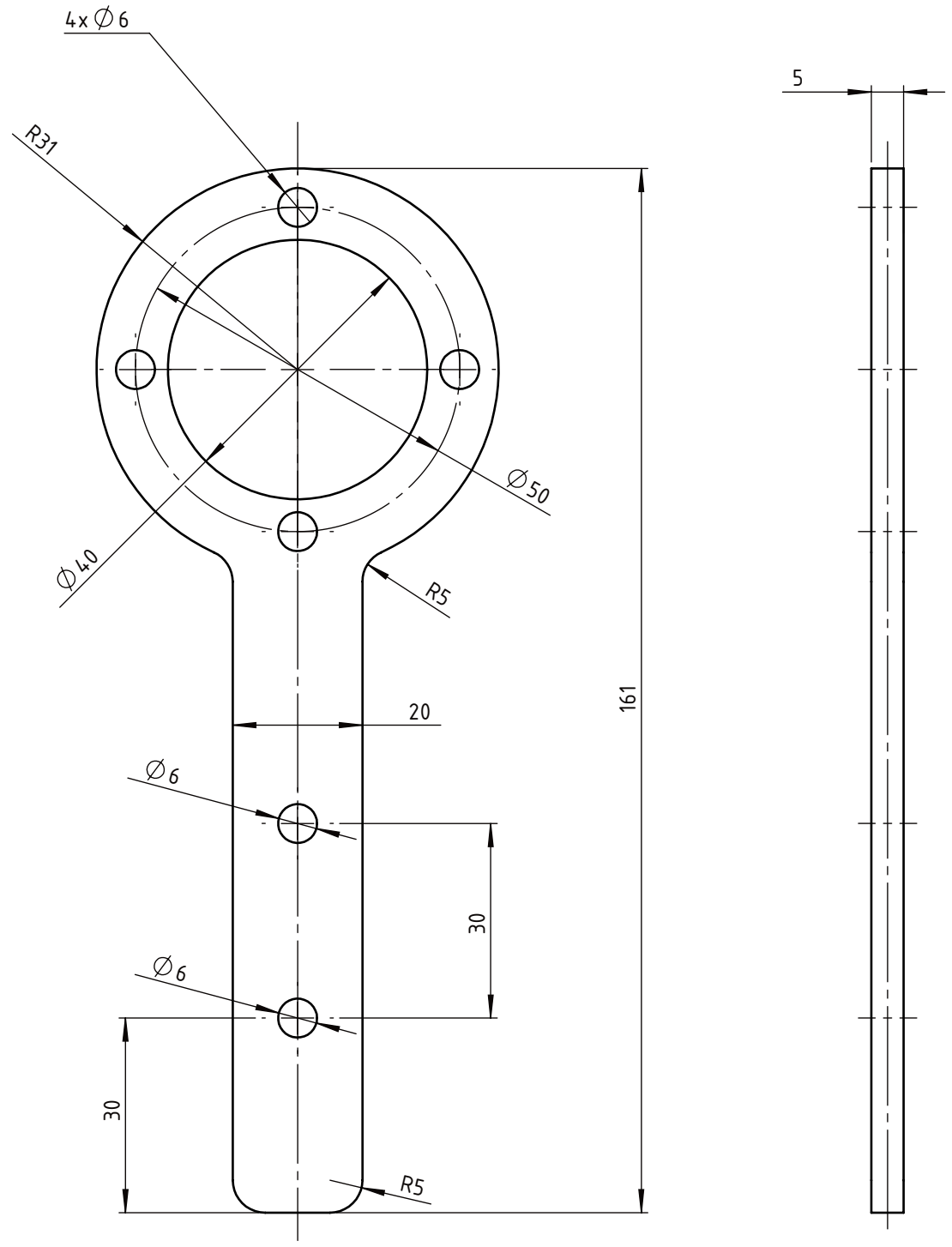


	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

FSB Zagreb

Studij strojarstva

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:	Smjer: Mehatronika i robotika	Kopija
Materijal: ABS	Masa:	ZAVRŠNI RAD
Mjerilo originala 1:2	Naziv: Kupola	Pozicija: 2
	Crtež broj: 100-02-002	Format: A4
		Listova: 1
		List: 1

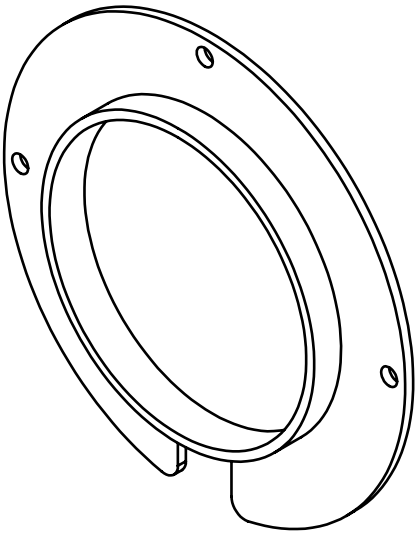


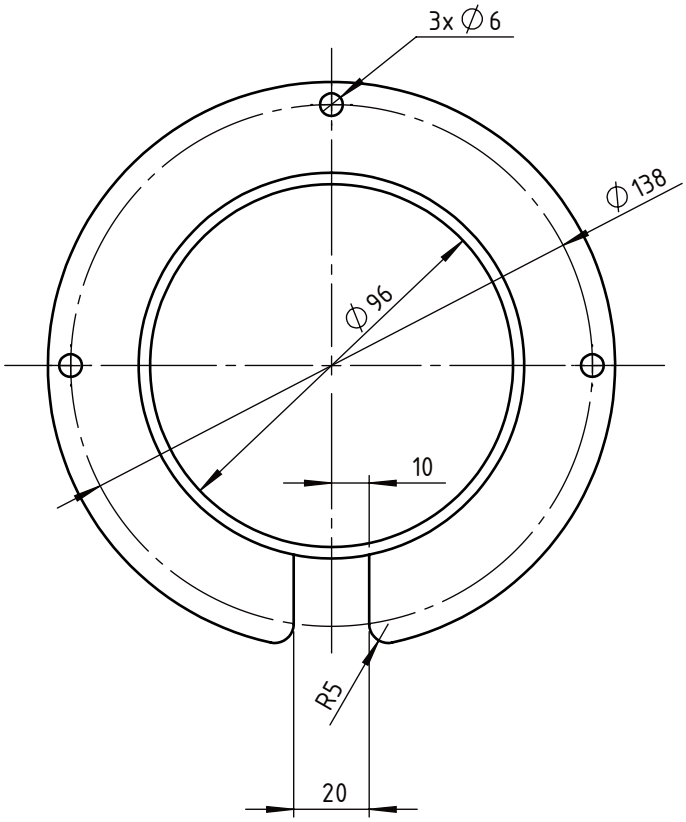
	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

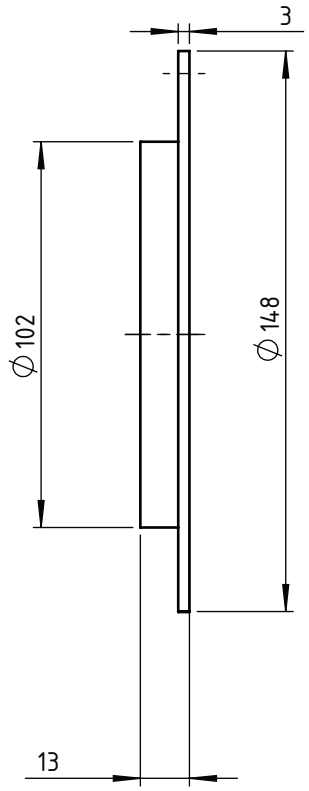
FSB Zagreb

Studij strojarstva

Objekt:		Objekt broj:	
		R. N. broj:	
Napomena:		Smjer: Mehatronika i robotika	Kopija
Materijal: ABS	Masa:	ZAVRŠNI RAD	
Mjerilo originala 1:1	Naziv: Nosač kupole	Pozicija: 3	Format: A4
	Crtež broj: 100-02-003		Listova: 1
			List: 1





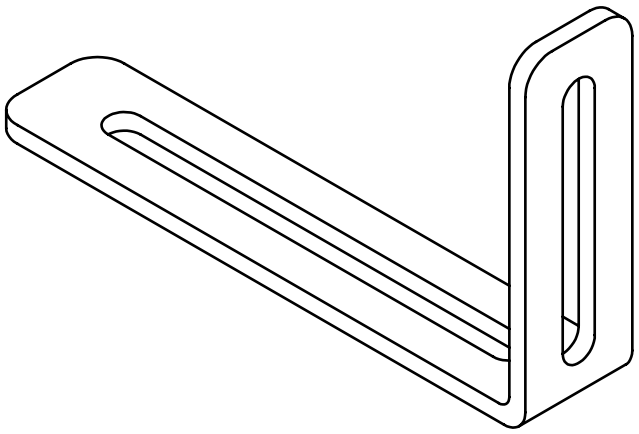


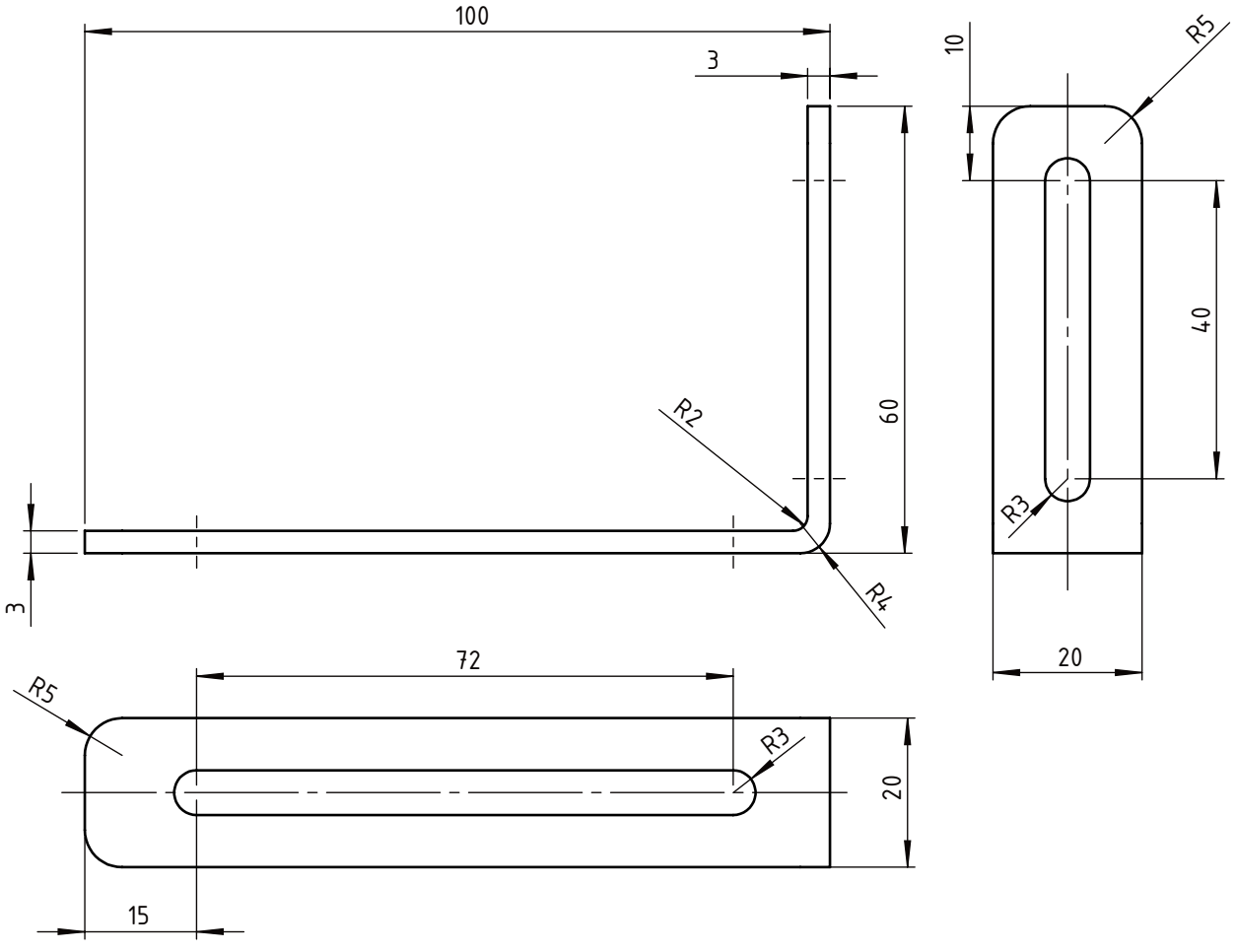
	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

FSB Zagreb

Studij strojarstva

Objekt:		Objekt broj:	
		R. N. broj:	
Napomena:		Smjer: Mehatronika i robotika	Kopija
Materijal: ABS	Masa:	ZAVRŠNI RAD	
Mjerilo originala 1:2	Naziv: Poklopac kupole	Pozicija: 4	Format: A4
			Listova: 1
	Crtež broj: 100-02-004		List: 1



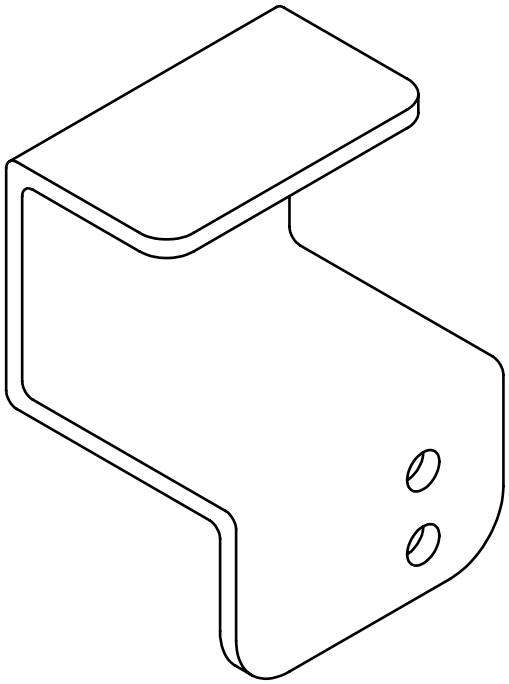


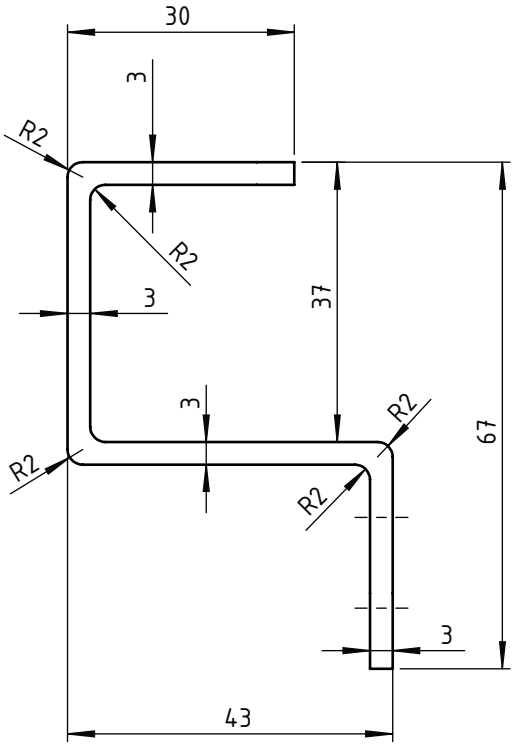
	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

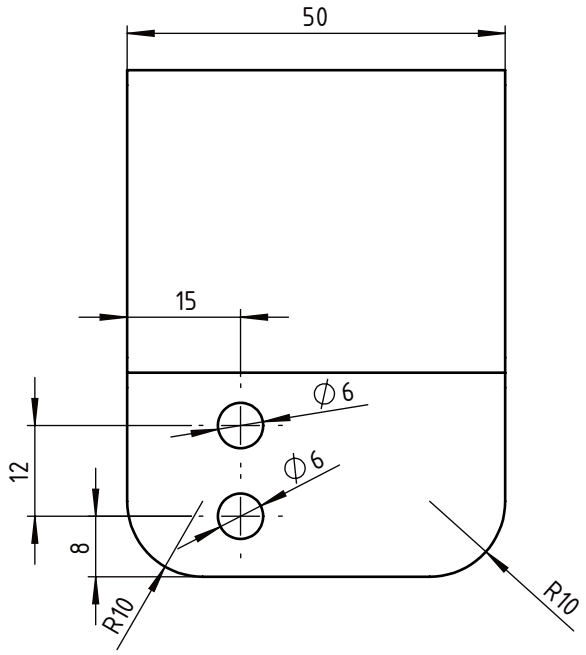
FSB Zagreb

Studij strojarstva

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:	Smjer: Mehatronika i robotika	Kopija
Materijal: ABS / Masa:	ZAVRŠNI RAD	
Mjerilo originala 1:1	Naziv: Držač kamere / Pozicija: 5	Format: A4
	Crtež broj: 100-02-005	Listova: 1 / List: 1





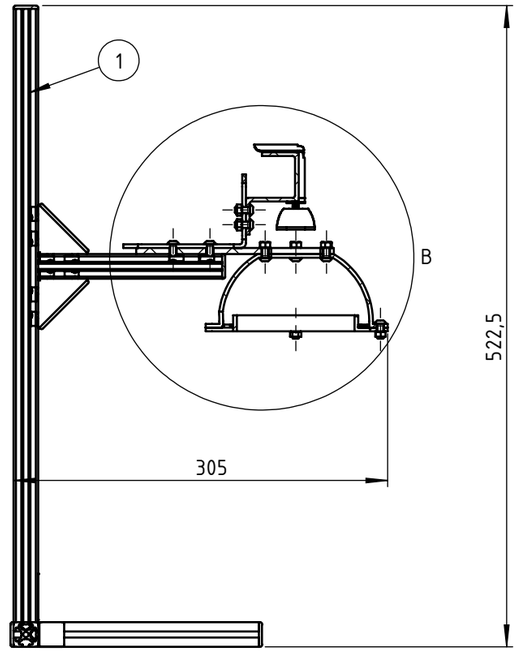


	Datum	Ime i prezime	Potpis
Projektirao		Vid Pavlović	
Razradio		Vid Pavlović	
Crtao		Vid Pavlović	
Pregledao		dr.sc.Tomislav Stipančić	
Voditelj rada		dr.sc.Tomislav Stipančić	

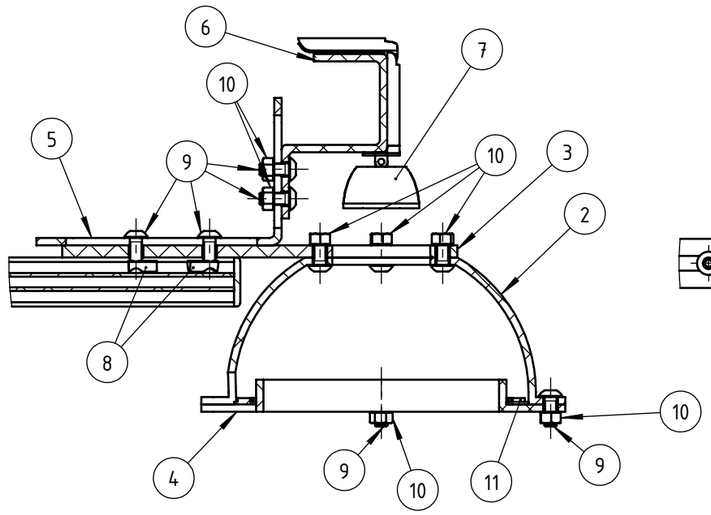
FSB Zagreb

Studij strojarstva

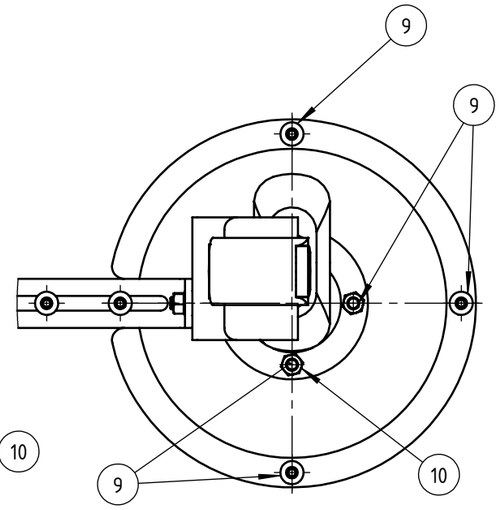
Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:	Smjer: Mehatronika i robotika	Kopija
Materijal: ABS / Masa:	ZAVRŠNI RAD	
Mjerilo originala 1:1	Naziv: Držač za C270	Pozicija: 6
	Crtež broj: 100-02-006	Format: A4 / Listova: 1 / List: 1



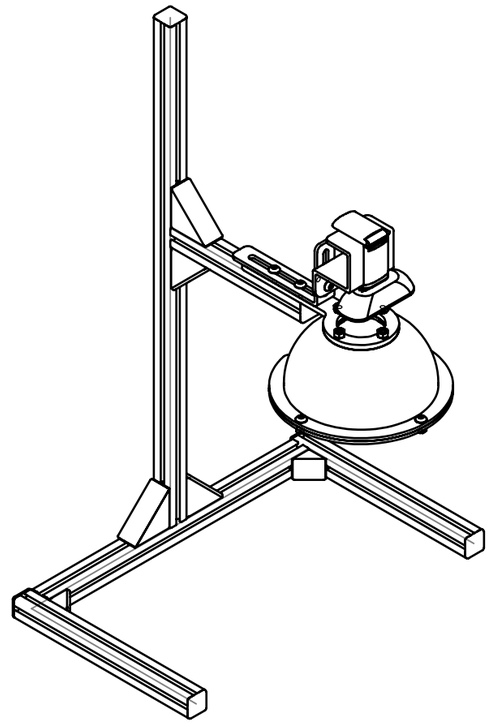
PRESJEK A-A

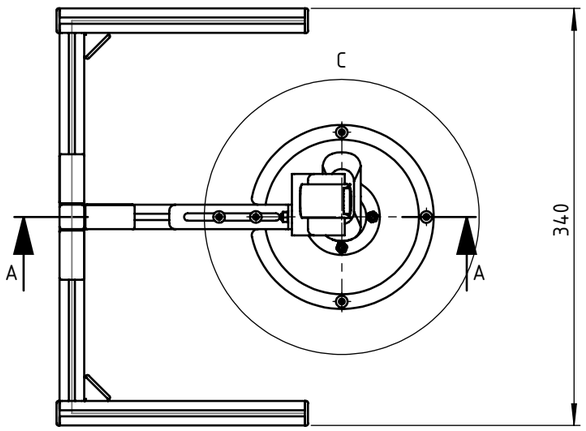


DETALJ B
M 2 : 5



DETALJ C
M 2 : 5





11	LED prsten	1	–	–	∅120x3.5	
10	Matica M5	9	ISO 4032	10.9	–	
9	Vijak M5x12	11	ISO 7380	8.8	–	
8	Matica M5 za ITEM profile 20x20	2	0042511	–	–	
7	Logitech C270	1	–	–	–	
6	Držač za C270	1	100-02-006	ABS	67x43x50	
5	Držač kamere	1	100-02-005	ABS	100x60x20	
4	Poklopac kupole	1	100-02-004	ABS	∅150x13	
3	Nosač kupole	1	100-02-003	ABS	161x62x5	
2	Kupola	1	100-02-002	ABS	∅150x63	
1	Nosač	1	100-01	ABS	340x520x203	
Poz.	Naziv dijela	Kom.	Crtež broj Norma	Materijal	Sirove dimenzije Proizvođač	Masa

Broj naziva - code		Datum	Ime i prezime	Potpis
	Projektirao		Vid Pavlović	
	Razradio		Vid Pavlović	
	Crtao		Vid Pavlović	
	Pregledao		dr.sc.Tomislav Stipančić	
	Voditelj rada		dr.sc.Tomislav Stipančić	

FSB Zagreb
Studij strojarstva

ISO - tolerancije

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Smjer: Mehatronika i robotika

Materijal:

Masa:

ZAVRŠNI RAD

Mjerilo originala
1:5

Naziv: Nosač za kameru i kupolno svjetlo

Pozicija:

Format: A3

Listova: 1

Crtež broj: 100-02

List: 1

0 10 20 30 40 50 60 70 80 90 100

Design by CADLab