Automatizirana sortirnica za grah

Maletić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Ivan Maletić

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Ivan Maletić

Zagreb, 2023.

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

Ovim putem zahvaljujem prof. dr. sc. Tomislavu Stipančiću na mentorstvu, korisnim savjetima i potpori te asistentu mag. ing. Leonu Korenu radi izrade 3D printanih pozicija.

Zahvaljujem također svojoj obitelji na podršci tijekom studentskih dana, kao i svojoj djevojci na pomoći i vjeri u mene.

Ivan Maletić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

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

DIPLOMSKI ZADATAK

Student: **Ivan Maletić** JMBAG: 0035216743

Naslov rada na hrvatskom jeziku: **Automatizirana sortirnica za grah**

Naslov rada na engleskom jeziku: **Automated bean sorting facility**

Opis zadatka:

Proces ručne obrade i sortiranja plodova graha je zamoran i ponavljajući posao zbog velike količine te malih dimenzija ploda. Zbog tih razloga razvoj automatiziranih sustava za sortiranje zrna graha je poželjan. U radu je potrebno razraditi i realizirati ideju linije za automatizirano sortiranje zrna graha koja koristi računalni model dubokog učenja za klasifikaciju zrna po kvaliteti.

Sumarno, u okviru rada je potrebno:

- predvidjeti te opisati probleme koji se mogu pojaviti kod razvoja i realizacije linije za automatizirano sortiranje plodova graha,
- prikupiti podatke te kreirati bazu slika koja prikazuje zrna graha u različitim pozicijama te uvjetima na sceni,
- kreirati računalni model dubokog učenja za klasifikaciju zrna graha po kvaliteti,
- osmisliti i izraditi sustav za optički nadzor te izuzimanje pojedinih zrna graha iz skupine (sustav treba uključivati komponente kao što su pokretna traka, držač kamere, vodilica za spust zrna i sl.),
- odabrati te implementirati električne komponente i prigone sustava,
- osmisliti model upravljanja komponentama sustava,
- izraditi projektnu dokumentaciju odabranog rješenja koristeći CAD programsku podršku.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
4. svibnja 2023.	6. srpnja 2023.	17. – 21. srpnja 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Izv. prof. dr. sc. Tomislav Stipančić Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

SAŽETAK

U sklopu ovog diplomskog rada opisan je postupak razvoja linije za automatizirano sortiranje plodova graha. Linija je zamišljena na način da se iz nesređenog stanja (hrpe) zrna graha odvajaju pojedinačno te nakon toga, s određenim razmakom, dolaze na konvejer. Iznad konvejera nalazi se kamera koja kontinuirano snima stanje te šalje podatke na računalo. Na računalu se obrađuje signal s kamere i koristi se kao ulaz u neuronsku mrežu koja prepoznaje 3 klase - dobar, loš i ništa. Klasa dobar predstavlja sliku na kojoj se nalazi zrno graha koje je ispravnog oblika, boje i veličine. Klasa loš predstavlja sliku na kojoj se nalazi zrno graha s bilo kojom vrstom oštećenja - otkinut dio zrna uslijed djelovanja sile prilikom prijašnje obrade, promjena boje uslijed djelovanja vlage, premala veličina i slično. Klasa ništa predstavlja sliku bez prisutstva zrna graha. U normalnom režimu rada stroja to su sva moguća stanja. Izlaz neuronske mreže određuje poziciju servo motora čiji nastavak usmjerava zrno prema odgovarajućem izlazu. Kroz rad su detaljno prikazani procesi razvoja svake komponente sustava te su dani razlozi za odabir kupovnih komponenti.

Ključne riječi: grah, automatizacija procesa, neuronska mreža, umjetna inteligencija, 3D modeliranje

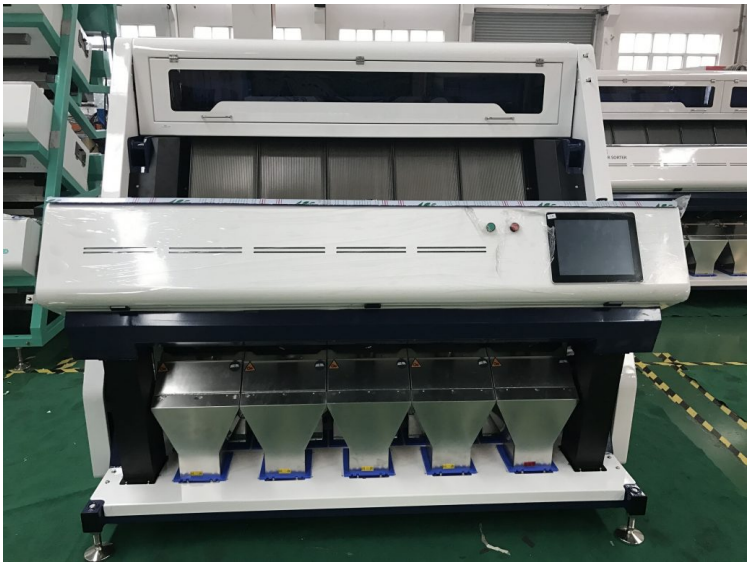
SUMMARY

This masters thesis describes the process of developing a line for automated sorting of bean grains. The line is designed in such a way that individual grains of beans are separated from an unsorted state (pile) and then, at a certain distance, they come onto a conveyor. Above the conveyor, there is a camera that continuously records the state and sends the data to a computer. The camera signal is processed on the computer and used as input to a neural network that recognizes 3 classes - good, bad, and nothing. The class good represents an image containing a bean grain that is of correct shape, color, and size. The bad class represents an image containing a bean grain with any type of damage - a broken part of the grain due to previous excessive force, color change due to moisture, undersized grain, and so on. The nothing class represents an image without the presence of a bean grain. In normal machine operation mode, these are all possible states. The output of the neural network determines the position of a servo motor whose extension directs the grain towards the corresponding output. The thesis provides a detailed presentation of the development processes of each system component and gives reasons for choosing commercial components.

Keywords: beans, process automation, neural network, artificial intelligence, 3D modeling.

1 UVOD

Motivacija za izradu automatizirane sortirnice za grah proizlazi iz želje za automatizacijom manualnog procesa klasifikacije zrna graha koja se vrši nakon žetve u velikom postotku hrvatskih poljoprivrednih gospodarstava koja se bave proizvodnjom graha. Projektiranje sortirnice bazirano je na tehnologiji aditivne proizvodnje koja pruža izradu kompleksne geometrije pozicija bez otpadnog materijala. Također, korišteno je područje dubokog učenja koje, u ovom slučaju, zamjenjuje ljudsku procjenu kvalitete zrna, a podatci za nadzirano učenje dani su u formi slika i oznaka odgovarajućih objekata. Cilj rada je dizajnirati robustan te pouzdan stroj koji će neprekidno raditi bez ljudske podrške. Uz kvalitetan, odnosno točan sustav računalnog vida te bez uskog grla u protoku stroja, potencijalni budući vlasnici stroja postigli bi veću konkurentnost na tržištu zbog umanjenja vremena sortiranja te zbog nestanka potrebe za radnom snagom na poslovima vezanim uz sortiranje. Automatizirana linija također može, uz preinake na određenim pozicijama i u modelu dubokog učenja biti korištena i za sortiranje različitih vrsta orašastog voća poput badema, oraha, lješnjaka, kikirikija, ali i zrna drugih vrsta poput kave ili slanutka. Na tržištu postoje automatizirane sortirnice koje imaju izrazito visok kapacitet sortiranja (nekoliko tona zrna po satu), no njihove cijene kreću se od 25 000 do 50 000 € što ih čini prikladnijima za tvorničke primjene. Na sljedećoj slici prikazana je automatizirana sortirnica kapaciteta sortiranja većeg od 2,5 t/h kineske tvrtke *Metak*.



Slika 1: Industrijska automatizirana sortirnica [1]

Poljoprivredna gospodarstva u Hrvatskoj često nemaju motivaciju za takvom investicijom budući da je prosječan prinos graha približno 3 tone po hektaru, a prema podacima Gospodarskog lista prosječno gospodarstvo posjeduje 5 hektara zemlje, te ako se uzme u obzir prosječna otkupna cijena otkupa od 3,30 € te profitna marža od 36%, dolazi se do podatka da je neto zarada prosječnog gospodarstva koje se bavi proizvodnjom graha približno 18 000 €. Kupnja industrijske sortirnice koja bi uklonila troškove rada ljudi za poslove sortiranja koji iznose 11% ukupnih troškova uzgajanja kulture. Budući da u slučaju kupnje sortirnice profitna marža raste na 44% što donosi približno 21 780 € godišnje, dobiva se podatak da je potrebno 6,6 godina za povrat investicije za najjeftiniju varijantu sortirnice. Cilj ovog rada je konstruirati niskobudžetnu, ali preciznu i pouzdanu sortirnicu kojoj će period povrata investicije biti znatno manji.

2 ADITIVNA PROIZVODNJA

Budući da je zahtjev za veliku većinu pozicija u ovome radu da budu izrađene pomoću 3D printanja, u ovom poglavlju prolaze se glavni aspekti tehnologije aditivne proizvodnje. Aditivna proizvodnja (eng. *additive manufacturing*) naziv je za proizvodnu granu koja se bavi izradom proizvoda nanošenjem čestica materijala u slojevima. Velike prednosti u usporedbi s ostalim načinima proizvodnje su manje otpadnog materijala prilikom izrade proizvoda, izrada složenih geometrija koje su teško ili nemoguće izvedive tradicionalnim proizvodnim postupcima te manje uređaja i osoblja koji sudjeluju u cjelokupnom procesu proizvodnje budući da ne isti ne zahtjeva planiranje toka procesa, specifičnu opremu za rad s materijalima, transport između radnih mjesta i slično. Valja dodati kako troškovi izrade prilikom povećanja kompleksnosti proizvoda za konvencionalne postupke rastu mnogo znatnije nego što je to prilikom aditivne proizvodnje.

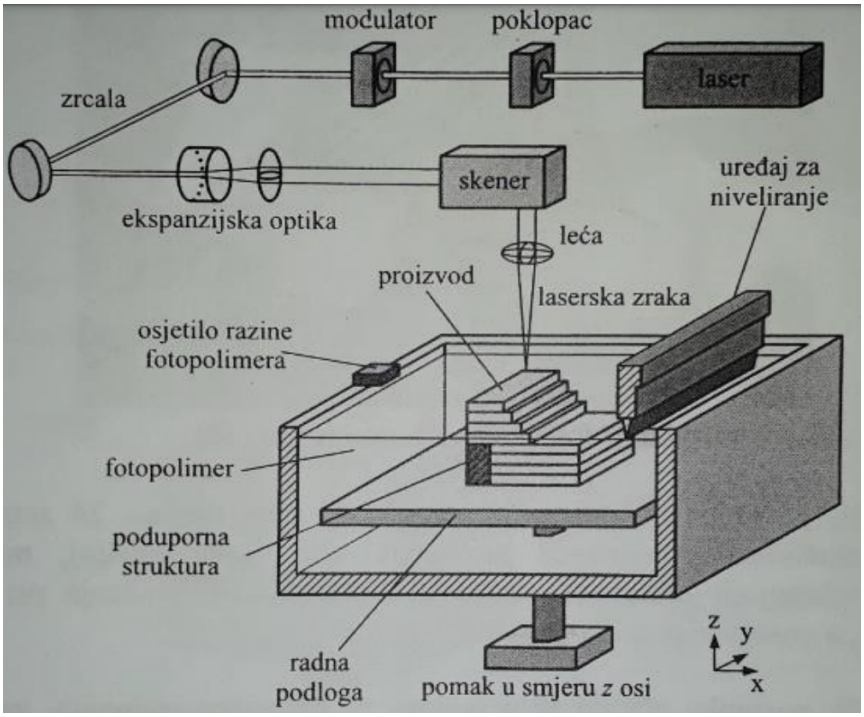
2.1 Procesi unutar aditivne proizvodnje

Unutar aditivne proizvodnje postoje razni postupci, no svi se zasnivaju na postepenoj slojevitoj gradnji proizvoda. Naime, 3D model proizvoda (najčešće u STL formatu koji proizvod prikazuje kao mrežu povezanih trokuta) ubacuje se u softver za rezanje (eng. *slicer*) te se tamo tvore slojevi jednakih debljina (u velikoj većini manji od milimetra) koji naslagani jedan na drugi čine prvotni proizvod. Godine 2009. uvedena AMF datoteka (eng. *Additive Manufacturing File*), koja uz STL format postaje standard za postupke aditivne proizvodnje i čini osnovu za rezanje u slojeve. AMF datoteka predstavlja jedan ili više objekata raspoređenih u vektore. Svaki je objekt opisan kao skupina ne preklopljenih volumena koji su opisani kao mreža trokuta koja povezuje skupinu točaka. Tehnologije aditivne proizvodnje mogu se općenito podijeliti na postupke koji upotrebljavaju materijal u čvrstom stanju, kapljevinu i prah. Postupci kod kojih se koristi čvrsti materijal su taložno

očvršćivanje (eng. *Fused Deposition Modeling*, FDM) i proizvodnja laminiranih objekata (eng. *Laminated Object Manufacturing*, LOM). Postupci koji upotrebljavaju kapljevite materijale su stereolitografija (eng. *Stereolithography*, SLA) i *PolyJet* postupak, dok su postupci koji koriste prah taljenje pomoću snopa elektrona (eng. *Electron Beam Melting*, EBM) i selektivno lasersko sinteriranje (eng. Selective Laser Sintering, SLS).

2.1.1 Stereolitografija

Tehnologija stereolitografije temelji se na fotolitografskim metodama s UV polimerizacijom. Proizvodi napravljeni postupkom stereolitografije nastaju polimeriziranjem niskoviskozne polimerne kapljevine sloj po sloj. Sljedeća slika prikazuje shemu stereolitografskog postupka izrade.

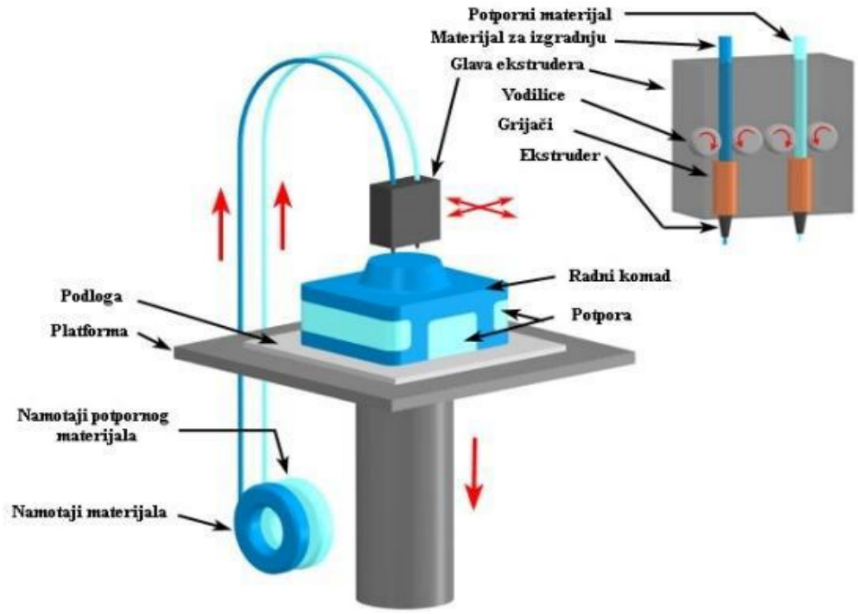


Slika 2: Stereolitografski postupak [2]

Osnovni dijelovi SLA uređaja su laser (uz njegovu opremu), zrcalo za usmjeravanje zraka, posuda s fotopolimerom te podloga s mogućnošću kretanja u smjeru okomite osi. Princip rada temelji se na generiranju i usmjeravanju laserske UV zrake koja se potom pomoću pomičnih zrcala usmjerava na različite horizontalne ravnine fotopolimera. Molekule fotopolimera selektivno očvršćuju i srašćuju prilikom zračenja te time prianjaju na prethodni sloj. Početni sloj najčešće se nanosi na metalnu radnu podlogu. Pri završetku nanošenja sloja, radna podloga se pomiče po vertikalnoj osi u iznosu jedne debljine sloja. Proizvodi načinjeni stereolitografskom metodom odlikuju se visokom razlučivošću, odnosno velikom razinom detelja radi male visine slojeva, no proizvodi moraju proći naknadnu obradu nakon završavanja postupka kao što je kupka u alkoholu za odstranjivanje viška polimera, uklanjanje potpora te dodatno otvrđivanje.

2.1.2 Taložno očvršćivanje

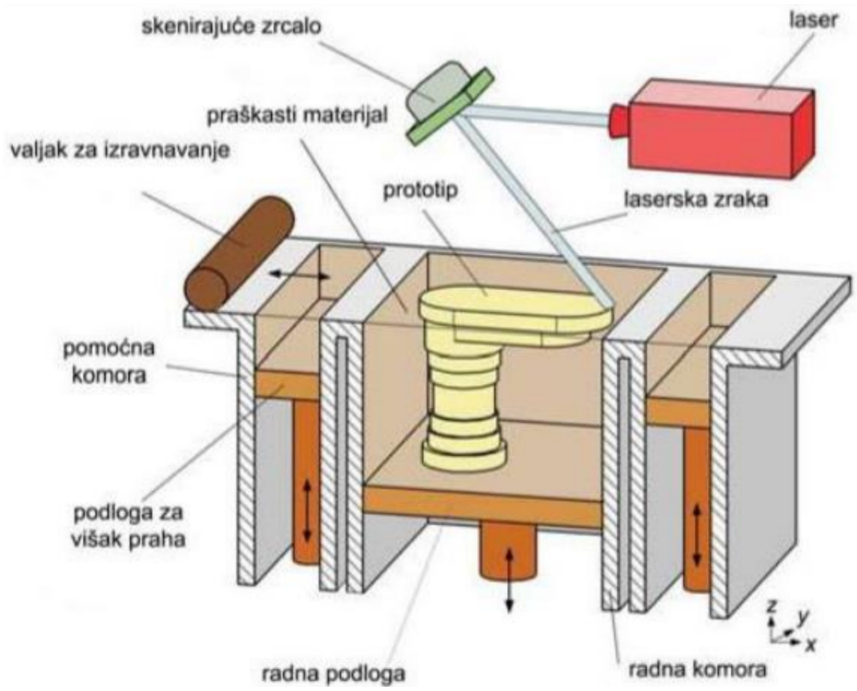
Prilikom izrade proizvoda pomoću postupka taložnog očvršćivanja (FDM) koristi se rastaljeni polimerni materijal koji prolazi kroz ekstruder čijom pozicijom upravlja numerički kontrolirani centar. Postoje i vrste uređaja koje, osim pozicijom ekstrudera, upravljaju i pozicijom podloge. Budući da se materijal počinje hladiti nakon izlaska iz ekstrudera postoji zahtjev za potporom na mjestima gdje nema materijala, a u narednim slojevima će ih biti. U protivnom, rastaljeni materijal jednostavno će propasti umjesto slijeganja. Postoje ekstruderi koji za potporu koriste materijal druge vrste, ali i oni koji koriste materijal iste vrste, no različito nanesene geometrije. Najčešći materijali koji se koriste prilikom postupka su PLA, ABS te PET-G, a pripremljeni su u obliku žice koja naknadno prolazi postupak taljenja. Na sljedećoj slici prikazani su osnovni elementi i princip rada prilikom taložnog očvršćivanja.



Slika 3: Princip rada uređaja prilikom FDM postupka [2]

2.1.3 Selektivno lasersko sinteriranje

Tehnika je bazirana na selektivnom sjedinjenju praškastih materijala u čvrsti oblik, pod utjecajem laserskog zračenja. Laserska zraka CO_2 usmjerava se na materijal, koji uslijed izloženosti visokoj temperaturi sinterira. Pod utjecajem visoke temperature, povećava se adhezija između čestica praha, te se na taj način prah grupira u veću krutinu točno određenog oblika. Suvišni prah u svakom sloju služi kao podrška tijekom procesa izrade. Na sljedećoj slici prikazan je princip rada uređaja prilikom SLS postupka.



Slika 4: Princip rada uređaja prilikom SLS postupka [2]

Ovak tehnologija ima najveći raspon dostupnih materijala za korištenje, budući da se mnogo metala može sinterirati. To se posebno odnosi na čiste metale koji se proizvode u izoliranim i sterilnim uvjetima, ali i mnogi nemetali se mogu sinterirati kao što je staklo ili neki organski polimeri. Također, zbog iznimnih svojstava može se koristiti titan što je izuzetno korisno za zrakoplovnu i svemirsku industriju te medicinske primjene poput implantata. Prednost ovakve tehnologije je to što daje bolja mehanička svojstva modela kao i brži ispis u odnosu na ostale tehnologije poput stereolitografije ili FDM-a. Glavni nedostatak je u kvaliteti površine predmeta. Potrebna je zaštitna atmosfera jer neki materijali koji ispuštaju otrovne plinove te cijeli sustav zauzima velik prostor.

3 IZRADA MODELA DUBOKOG UČENJA

U sklopu ovog poglavlja opisan je postupak izrade modela za prepoznavanje kvalitete zrna graha na osnovu slike. Za izradu modela potrebno je prvotno sakupiti slike te na njima označiti odgovarajuće klase u odgovarajućim područjima. Generirane oznake potrebno je izmijeniti u format prikladan za učenje modela. Kao baza modela koristi se pred-trenirani VGG model koji je opisan u jednom od narednih podpoglavlja. Nakon procesa učenja provedena je validacija rezultata.

3.1 Prikupljanje i označavanje podataka

Prikupljanje kvalitetnih podataka prvi je korak ka preciznom i robusnom modelu dubokog učenja. Duboko učenje zahtijeva velike količine podataka kako bi model mogao naučiti složene značajke i obrasce u podacima. Više podataka omogućuje modelu da stekne općenitije razumijevanje problema i bolje generalizira na novim primjerima. Bez dovoljno podataka, model može biti prenaučen na dostupnim primjerima, što rezultira lošom sposobnošću generalizacije na nove podatke. U slučaju ovog diplomskog zadatka, podaci su pruženi u obliku slika. Slika u *RGB* formatu pretvorena je u 3 matrice veličine 224x224, gdje svaka matrica prikazuje jedan kanal boja - crveni, zeleni te plavi. Idealno je da se pruže različiti konteksti okruženja u kojima je slika prikupljena - različite boje i vrste pozadina, različita osvjetljenja, različiti kutevi snimanja itd. Slike su prikupljene pomoću mobilnog telefona budući da se na internetskim stranicama ne može pronaći dovoljan broj slika (naročito klase loš) zrna željene vrste Lipa-3. Prikupljanje podataka s pravilno postavljenim oznakama (labelama) omogućuje modelu da razumije koji primjeri pripadaju kojim klasama ili ciljnim vrijednostima. Labele pružaju superviziju tijekom obuke, omogućujući modelu da usporedi svoje predikcije s pravim vrijednostima i prilagodi svoje parametre (težine i

pomake) kako bi minimizirao pogrešku. Sljedeće dvije slike prikazuju primjer zrna iz klase dobar te zrna iz klase loš.

Slika 5: Primjer slike zrna graha iz klase dobar

Slika 6: Primjer slike zrna graha iz klase loš

U klasu loš smješteno je svako zrno graha koje ima bilo kakvu vrstu oštećenja - različitu boju, deformiran oblik, krhotine površine ili cijelog zrna, premalu veličinu, kratere na površini itd. Prikupljeno je 100 slika vrste dobar i loš te 50 slika klase ništa, a svaka od njih kasnije je 3 puta rotirana za 90 stupnjeva (kako bi model lakše generalizirao) što dovodi do veličine seta podataka za trening od 1000 slika. Slike su označavane u softveru labelImg. Nakon što se slika učita označi se pravokutnik sa željenim objektom i nakon toga se označi željena klasa. Rezultati označavanja spremaju se u XML formatu vidljivom na sljedećoj slici.

```xml
<annotation>
    <folder>img</folder>
    <filename>0204_105851crop.jpg</filename>
    <path>C:\Users\ivanm\Desktop\FSB\Projekt i diplomski rad\200 pojedinacnih\dobar\train\img\0204_105851crop.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>2604</width>
        <height>2604</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>dobar</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>773</xmin>
            <ymin>850</ymin>
            <xmax>1800</xmax>
            <ymax>1447</ymax>
        </bndbox>
    </object>
</annotation>

```

Slika 7: XML datoteka s podacima označavajućeg pravokutnika

Na prethodnoj slici uokvireni su elementi koji prikazuju označenu klasu te koordinate označavajućeg pravokutnika zapisane u pikselima od kojih se može generirati isti. Navedeni podaci biti će kasnije izlaz iz neuronske mreže pa ih je potrebno izdvojiti kako bi se mogli koristiti za treniranje mreže. To je napravljeno pomoću skripte 1 u prilogu - učita se XML datoteka, pročitaju se vrijednosti elemenata imena slike, koordinata pravokutnika te klase te se zapišu u CSV format (tip datoteke gdje su vrijednosti odvojene zarezom - eng. *Comma Separated Values*). Svaki redak predstavlja jednu sliku s njenim podacima i njegov izlged je kako slijedi:

```
0204_105847crop.jpg,711,664,1803,1437,0
0204_105851crop.jpg,773,850,1800,1447,0
0204_105901crop.jpg,868,293,1678,1266,0
0204_105903crop.jpg,727,550,1673,1272,0
0204_105906crop.jpg,1089,912,1886,1685,0
0204_105908crop.jpg,1022,920,1930,1537,0
0204_105911crop.jpg,1224,974,2111,1596,0
0204_105932crop.jpg,697,845,1730,1631,0
0204_105947crop.jpg,584,482,1503,1255,0
0204_105951crop.jpg,1443,534,2292,1639,0
0204_105955crop.jpg,1065,985,2070,1691,0
0204_105959crop.jpg,1414,153,2224,1055,0
0204_110050crop.jpg,786,437,1719,1045,0
0204_110053crop.jpg,724,496,1627,1072,0
0204_110055crop.jpg,700,282,1662,1069,0
0204_110057crop.jpg,1149,401,1859,1166,0
0204_110100crop.jpg,1251,547,2297,1199,0
```

Slika 8: CSV datoteka s podacima označavajućeg pravokutnika i klase

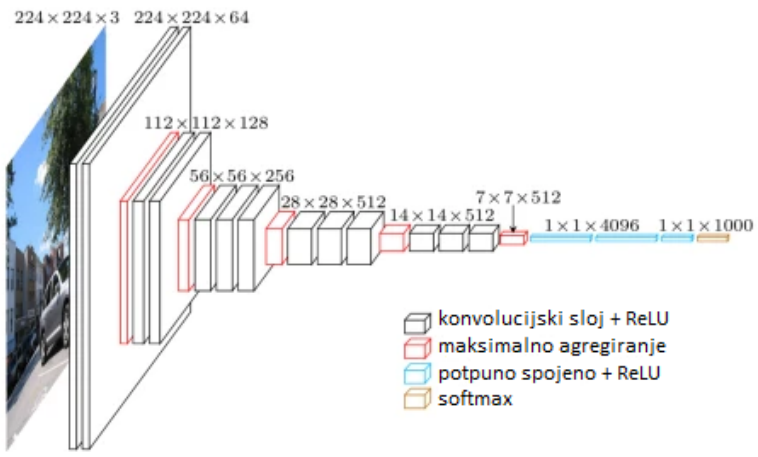
Ovime je zaključeno prikupljanje svih potrebnih podataka za treniranje mreže.

3.2 VGG model

VGG (*Visual Geometry Group*) model je model konvolucijske neuronske mreže koji je predstavljen od strane znanstvenika s Oxforda. Model postiže preciznost od 92.7 % na skupu podataka *ImageNet* koji sadrži 14 milijuna slika s približno 1000 klasa. Sastoji se od ukupno 16 slojeva, točnije od 13 konvolucijskih slojeva i 3 potpuno povezana sloja. Za konačnu klasifikaciju koristi se *softmax* aktivacijska funkcija čija je jednadžba prikazana sljedećim izrazom:

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)} \tag{1}$$

gdje je $S(y)_i$ vrijednost *softmax* aktivacije nad i-tim elementom u izlaznom sloju, $exp(y_i)$ vrijednost standardne eksponencijalne funkcije i-tog elementa, a $\sum_{j=1}^{n} \exp(y_j)$ zbroj pojedinačnih vrijednosti standardnih eksponencijalnih funkcija nad cijelim izlaznim slojem. Na sljedećoj slici prikazana je arhitektura sustava.



Slika 9: Aritektura VGG modela [8]

3.2.1 Konvolucijski sloj

Svaki konvolucijski sloj sastoji se sastoji od više filtara ili jezgri (eng. *kernel*), pri čemu svaki filter generira jednu dvodimenzionalnu aktivacijsku mapu. Izlaz sloja sastoji se od svih aktivacijskih mapa, što rezultira više dvodimenzionalnih matrica, gdje svaka matrica predstavlja jednu dubinu izlaza. Tijekom faze unaprijednog prolaza (eng. *forward pass*) provodi se operacija konvolucije nad ulazom u sloj te filtrom. Rezultat konvolucije je dvodimenzionalna aktivacijska mapa koja prikazuje odziv filtera na svakoj prostornoj poziciji. Tijekom učenja mreže elementi filtra biti će podešeni na način da prepoznaju vizualne značajke unutar slike kao što su rubovi, teksture, boje, oblici i slično. U konkretnom primjeru VGG modela, koriste se matrice dimenzija 3x3 kao filtri. Da bi se odredila veličina izlaza konvolucijskog sloja, potrebno je definirati korak pomaka filtra (eng. *stride*). Stride određuje koliko će se filtar pomaknuti po širini ili visini tijekom konvolucije. Širina izlaznog sloja W_y računa se na način da se od širine ulaznog sloja W_x oduzme veličina kvadratnog filtra F te se njihova razlika podijeli s korakom pomaka filtra S te se kvocijentu

doda 1:

$$W_y = \frac{W_x - F}{S} + 1 \tag{2}$$

Ukoliko je visina ulaza H_x jednaka širini ulaza W_x tada je i visina izlaza H_y jednaka širini izlaza W_y. Operaciju konvolucije najlakše je opisati algebarski, stoga je izrazom 2 opisana veza između ulaza i izlaza. Ulaz je zapravo izlaz iz prethodnog sloja pa je označen s $y_d^{l-1}(i,j)$, gdje i i j određuju poziciju unutar matrice, d označava dubinu, dok l označava o kojem sloju se radi. Težine unutar filtra označene su oznakom w. Veličina filtra označena je s F, f predstavlja aktivacijsku funkciju, dok b oznacava prag. Vrijednosti a i b povećavaju se za vrijednost ranije spomenutog koraka pomaka.

$$y^l(i,j) = \sum_{a=0}^{F-1} \sum_{b=0}^{F-1} \sum_{d=0}^{D-1} f\left(w_d(i,j) \cdot y_d^{l-1}(i+a, j+b) + b\right) \tag{3}$$

3.2.2 Sloj sažimanja

Slojevi sažimanja često se koriste u konvolucijskim neuronskim mrežama nakon nekoliko konvolucijskih slojeva s ciljem smanjivanja rezolucije mapi. Osim smanjivanja rezolucije slojevi sažimanja povećavaju prostornu invarijantnost (neosjetljivost na manje pomake značajki u uzorku) neuronske mreže. Kao i kod konvolucijskog sloja manji dio uzorka grupira se i obrađuje, te rezultira jednom vrijednošću. Postoji nekoliko vrsta sažimanja. Najčešći načini su sažimanje usrednjavanjem te maksimalno agregiranje kojeg koristi VGG 16. Kod sažimanja usrednjavanjem grupirani podaci zamjenjuju se aritmetičkom sredinom grupiranih vrijednosti, dok se kod maksimalne agregacije grupirane vrijednosti zamjenjuju maksimalnom vrijednošcu. Valja napomenuti kako dubina sloja prije sloja sažimanja i nakon sažimanja ostaje ista.

3.2.3 Potpuno povezani sloj

Potpuno povezani sloj je sloj u neuronskoj mreži koji povezuje sve neurone iz prethodnog sloja s neuronima u sljedećem sloju. Svaki neuron u potpuno povezanom sloju je povezan s svakim neuronu u prethodnom sloju. Glavna funkcija potpuno povezanog sloja je kombiniranje i transformacija značajki koje su naučene u prethodnim slojevima kako bi se donijela konačna predviđanja ili klasifikacije. Svaki neuron u potpuno povezanom sloju izračunava težinsku sumu svih ulaznih vrijednosti iz prethodnog sloja, uključujući i pomak (eng. *bias*) te primjenjuje aktivacijsku funkciju na tu težinsku sumu kako bi generirao izlaz.

3.3 Proces učenja mreže

Kako bi neuronska mreža mogla rješavati neki konkretan problem potrebno je podesiti njene težine kako bi za pojedini uzorak izlaz neuronske mreže bio ispravan. Najcešći algoritam koji se koristi za učenje neuronskih mreža je algoritam unatražne propagacije (eng. *backpropagation*). On omogućuje računanje gradijenta (derivacija funkcije gubitka u odnosu na težine) na temelju kojeg se ažuriraju težine mreže kako bi se minimizirala greška i poboljšala sposobnost generalizacije. Proces *backpropagation*-a provodi se u četiri koraka. Prvi od njih je unaprijedni prolaz (eng. *forward propagation*) pri kojem se ulazni podaci proslijeđuju kroz mrežu kako bi se generirale izlazne vrijednosti. Svaki sloj izračunava svoj izlaz na temelju težina i aktivacijske funkcije. Sljedeći korak je izračun gubitka (eng. *loss calculation*) - izlaz neuronske mreže uspoređuje se s ciljnim vrijednostima kako bi se izračunala vrijednost funkcije pogreške, odnosno razlika između stvarnog trenutnog izlaza mreže i očekivanog izlaza koji je dan u podatcima za učenje. Često se u procesu učenja kao funkcija gubitka koristi srednja kvadratna pogreške (eng. *Mean Squared Error*). Sljedeći

izraz prikazuje izračun funkcije pogreške za izlazni sloj mreže:

$$E = \frac{1}{2}\sum_{j}\left(y_j - t_j\right)^2 \tag{4}$$

gdje y_j oznacava vrijednost izlaz j-tog neurona, a t_j označava očekivani izlaz j-tog neurona. Gradijent gubitka u odnosu na težine izračunava se unatražnim propagiranjem kroz mrežu. Prvo se izračuna gradijent gubitka u odnosu na izlazni sloj, što je prikazano sljedećim izrazom:

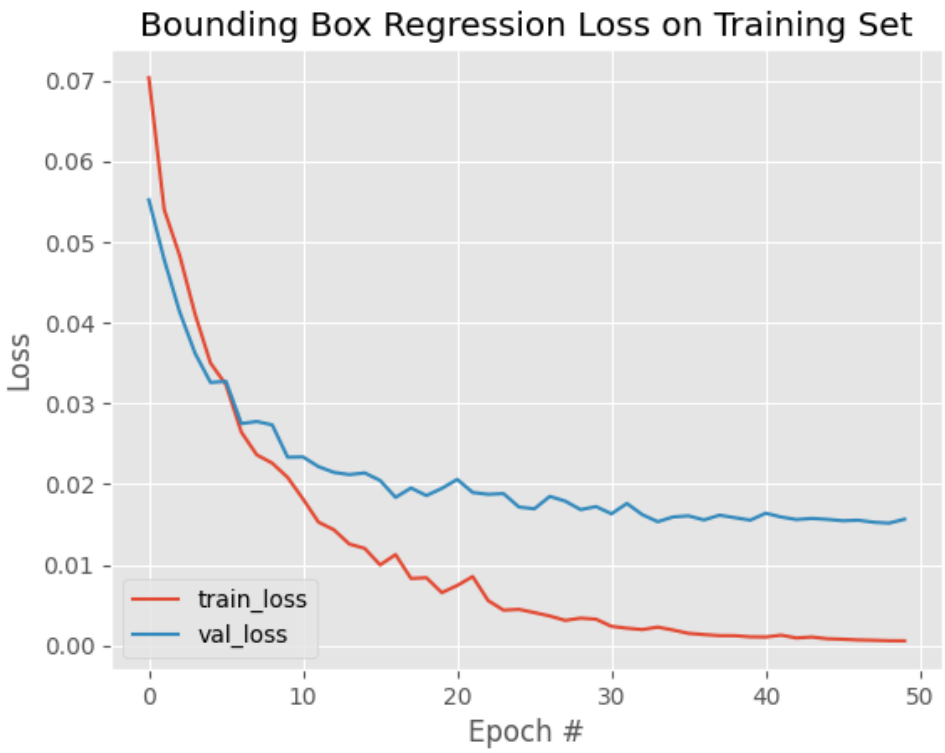
$$\frac{\partial E}{\partial y_j} = \frac{1}{2}\frac{\partial}{\partial y_j}\left(y_j - t_j\right)^2 = \frac{1}{2}\cdot 2\left(y_j - t_j\right) = \left(y_j - t_j\right) \tag{5}$$

Zatim se gradijent propagira unatrag kroz prethodne slojeve primjenjujući lančano pravilo. Na poslijetku se pristupa ažuriranju težina (eng. *weights update*) gdje se, koristeći izračunati gradijent, težine ažuriraju kako bi se minimizirala greška. Najčešće se koristi optimizacijski algoritam kao što je gradijentni spust (eng. *gradient descent*) ili njegove varijante. Za stvaranje i treniranje mreže korištena je besplatna softverska biblioteka otvorenog koda za strojno učenje i umjetnu inteligenciju po imenu *Tensorflow* potpomognuta aplikacijskim programskim sučeljem *Keras*, kao i raznim alatima za manipuliranje slikama, podacima te datotekama. Skripta za treniranje također je vidljiva u prilogu. Prvo je stvorena klasa *Config* u kojoj su definirane sve potrebne putanje, inicijalizirane postavke učenja - stopa učenja, broj epoha te veličina serije. Oblik klase izabran je kako bi kasnije bilo lakše pozivati se na određene stavke. Nakon toga potrebno je stvoriti liste sa slikama, željenim izlazima mreže te imenima slika. Željenih izlaza iz mreže ima 5: početna x koordinata pravokutnika, početna y koordinata pravokutnika, završna x koordinata pravokutnika, završna y koordinata pravokutnika te željena klasa (dobar, ništa ili loš). Podaci su prikupljeni na način da se stvorena CSV datoteka podijelila na retke i odgovarajući elementi pridruženi su varijablama. Valja napomenuti kako su vrijednosti pravokutnika normalizirane pošto je u izlaznom sloju

mreže aktivacijka funkcija sigmoid koja daje izlazne vrijednosti u rasponu [0,1]. Nakon stvaranja lista, skup podataka za treniranje podijeljen je u omjeru 9:1 - 90 % podataka korišteno je za treniranje, a 10 % za testiranje. Nakon toga učitan je VGG 16 model. Budući da VGG 16 model radi izuzetno dobro, njegovi parametri neće biti mijenjani niti trenirani, a na kraj modela dodano je još 4 sloja, svi s aktivacijskom funkcijom *ReLu*, osim zadnjeg koji koristi funkciju sigmoid kako je ranije spomenuto. Nakon toga stvara se objekt novog modela i pristupa se treningu.

3.4 Evaluacija rezultata

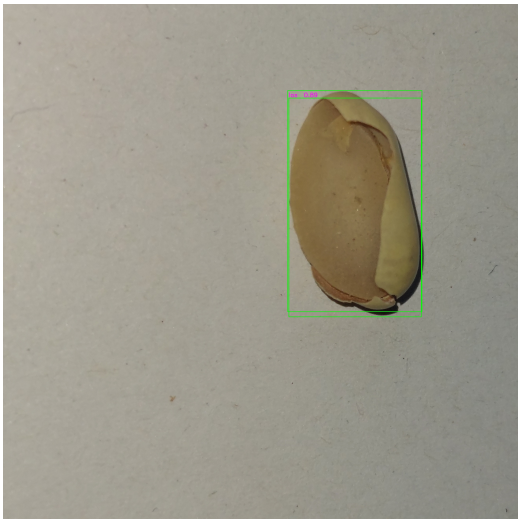
Kako bi se izvukao zaključak o točnosti mreže, koristi se funkcija gubitka, odnosno njena vrijednost kroz epohe. Funkcija gubitka nakon treninga vidljiva je na sljedećoj slici, crvena krivulja prikazuje vrijednost funkcije gubitka tijekom učenja na podacima koji su korišteni za učenje, dok plava prikazuje istu funkciju za podatke koji su korišteni isključivo za validaciju.



Slika 10: Funkcija gubitka unutar treninga na 50 epoha

U ovom slučaju, vidljivo je kako vrijednost funkcije gubitka konvergirala u nižu vrijednost nego na početku čime se može zaključiti da je proces učenja izvršen ispravno, odnosno da je točnost modela nakon procesa učenja veća nego prije prije. Isto tako, pokazaje se kako model bolje prepoznaje podatke koji su već prošli kroz njega u procesu učenja. Model je treniran 50 epoha, a evidentno je kako u samom početku zahvaljujući VGG 16 modelu funkcija gubitka veoma niska. Nakon treniranja provodi se evaluacija na 10 % podataka koji nisu ulazili u trening. Neki od rezultata vidljivi su na sljedećim slikama.

Slika 11: Pravilno predviđena slika klase dobar



Slika 12: Pravilno predviđena slika klase loš

Valja napomenuti kako je zadnji neuron izlaznog sloja mreže decimalni broj u rasponu [0,1] i predstavlja predviđenu klasu te sigurnost pripadnosti klasi. Npr. 0,05 predstavlja izuzetnu sigurnost mreže da se radi o dobrom grahu, dok se izlaz od 0,75 može kategorizirati kao

djelomična sigurnost mreže da se na slici nalazi loš grah. Prilikom evaluacije daju se i lokacije obuhvatnih pravokutnika, a budući da postoje istinite izvorne lokacije obuhvatnih pravokutnika, može se pristupiti računanju srednje prosječne vrijednosti preciznosti - eng. *mean Average precision* (mAP). U računanju srednje prosječne vrijednosti preciznosti potrebno je podijeliti vrijednost presjeka obuhvatnih pravokutnika te njihove unije - što je rezultat veći, mreža je točnija (moguć raspon je [0,1]):

$$IoU = \frac{A \cap B}{A \cup B} \tag{6}$$

Nakon računanja vrijednost IoU uspoređuje se sa željenom vrijednošću praga. Ukoliko je IoU veći od praga kaže se da je to ispravno pozitivni (TP) rezultat, u suprotnom je to lažno pozitivni (FP) rezultat. Na kraju se preciznost (za svaku klasu pri određenoj vrijednosti praga) računa kao:

$$P_p = \frac{TP}{TP + FP} \tag{7}$$

Gdje P_p predstavlja preciznost P pri pragu p. Srednja prosječna vrijednost preciznosti jest onda:

$$mAP = \frac{1}{n} \sum_{p=1}^{n} P_p \tag{8}$$

Kao standardna mjera uzima se preciznost od 50 do 95% i u ovom slučaju iznosi $mAP_{50-95} = 0.61$. Takav rezultat posljedica je većeg broja lažno pozitivnih rezultata na većim pragovima.

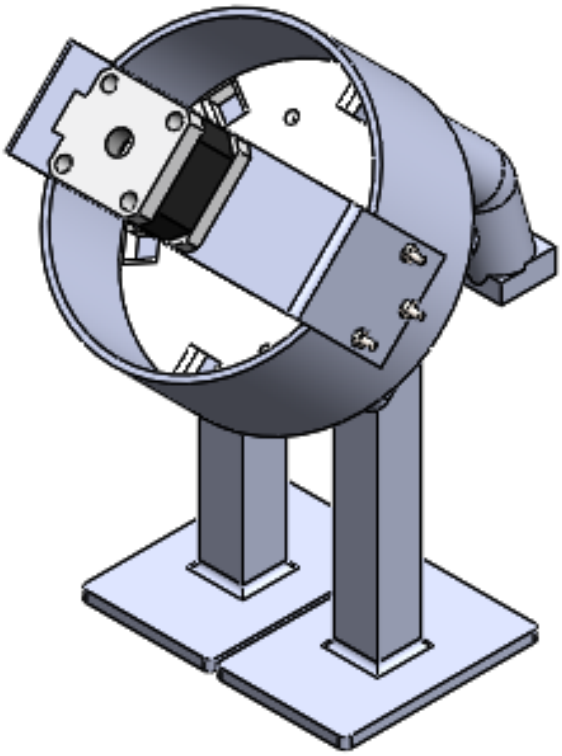
4 KONSTRUIRANJE SORTIRNICE

U ovom poglavlju opisan je postupak konstruiranja različitih pozicija automatske linije za sortiranje zrna graha. Kod razrađivanja svih pozicija obraćala se pozornost na sprječavanje predimenzioniranja, jednostavnost ugradnje te je uvedena multifunkcionalnost gdje god je moguće. Također, pozicije su dimenzionirane na način da se povezuju standardnim spojnim elementima i da koriste standardne ležajeve. Time je nestala potreba za izradu metalnih komponenti na CNC mašinama, tokarskim strojevima, glodalicama i sličnim uređajima koji bi produljili proces izrade.

4.1 Pojedinačno izdvajanje zrna iz hrpe

Budući da sustav ne koristi brze industrijske kamere koje uzimaju nekoliko stotina slika u sekundi ni pneumatsku aktuaciju, već (zbog nabavne cijene) standardne senzore te mali servo motor, zrna graha prije obrade potrebno je izdvojiti iz nesređenog stanja jedno po jedno. Zbog toga osmišljen je koncept koji funkcionira na principu statičnog bubnja i rotirajućeg diska. Na disku postoji 6 utora u koje upadaju zrna. Bubanj i disk rotirani su u odnosu na horizontalnu ravninu za 45° budući da postoji potreba da ostala zrna (koja nisu u utorima), rotiranjem diska i utjecajem gravitacije, padnu nazad u hrpu. U prilogu, crtež broj DR-01-001 prikazuje disk. Na crtežu je vidljivo 6 rupa za zrno kao i središnja rupa za glavu vijka koji je vezan za spojku. S druge strane spojka je vezana s koračnim motorom. Na prošloj slici vidljivo je kako je disk širok 120 mm, kako bi se lakše izvela montaža te izbjeglo trenje u radu, unutarnji promjer bubnja je 122 mm te je razmak između čeonih ploha diska i bubnja 1 mm. U prilogu, crtež broj DR-01-002 prikazuje bubanj. U crtežu (pogled B) vidljiva su 2 utora za magnete kao kvadratni utor 8x15 koji služi za umetanje Hallovog senzora koji služi za sprečavanje zaglavljivanja diska, što je spomenuto u poglavlju naziva upravljanje sustavom. Središnji provrt promjera 20 mm nalazi se tamo

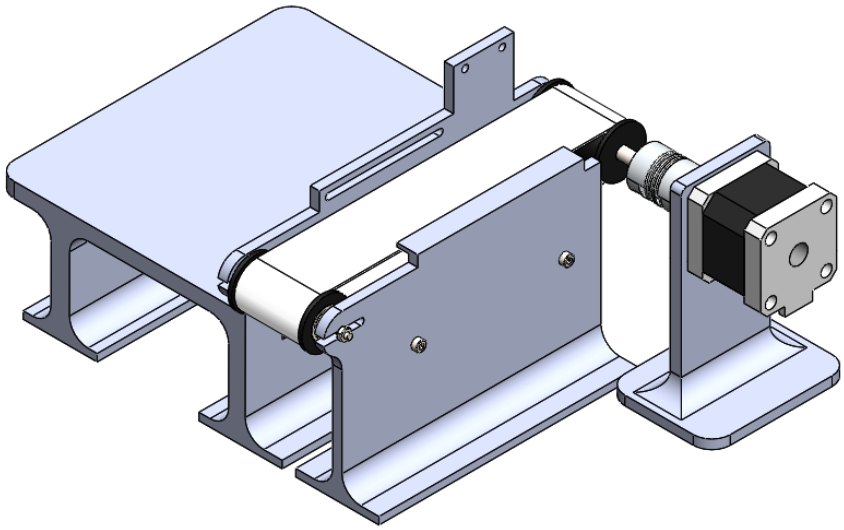
radi eventualnog uklanjanja vijka, dok provrt dimenzije 20 mm služi za ispust zrna budući da zrno koje je upalo u jedan od 6 utora prilikom rotiranja klizi po površini bubnja dok ne dođe do provrta. Nakon toga zrno putuje ispusnom cijevi do konvejera. Prolazne rupe 3,3 mm služe za učvršćivanje vijcima. Na sklopnom crtežu DR-04-001 prikazan je način spajanja ispusnog sklopa. Sve pozicije međusobno su vezane vijcima. Pozicija postolje osigurava koso pozicioniranje bubnja, dok dizajn držača motora određuje orijentaciju diska. Sljedeća slika prikazuje ispusni sklop u programskom okruženju *Solidworks*.



Slika 13: Ispusni sklop

4.2 Konvejer

Prilikom dizajniranja konvejera, njegove bočne stranice, osim za potporu bubnja služe za kretanje i potporu držača kamere te držača servo motora koji se nalaze iznad konvejera. Također, jedna bočna strana proširena je na način da je dodana horizontalna površina na kojoj stoje upravljači motora, postavna ploča te računalo. Napravljeni su utori u bočnim pločama koji omogućuju zatezanje remena. Odabran je remen GG5 35x392 B, plosnati remen s nosivim slojem od poliamida. Bočne ploče pričvršćene su vijcima - na jednoj ploči nalaze se mjedni umetci, a na drugoj prolazna rupa za vijak. Za spoj bubanja konvejera s bočnim pločama koriste se 625 ležajevi zajedno s pozicijama za centriranje (tehnični crtež vidljiv u prilgu) te vijcima i maticama. Pogonski bubanj pomoću spojke vezan je na koračni motor. Na sljedećoj slici prikazan je model konvejera, a njegov sklopni crtež nalazi se u prilogu.



Slika 14: Konvejer

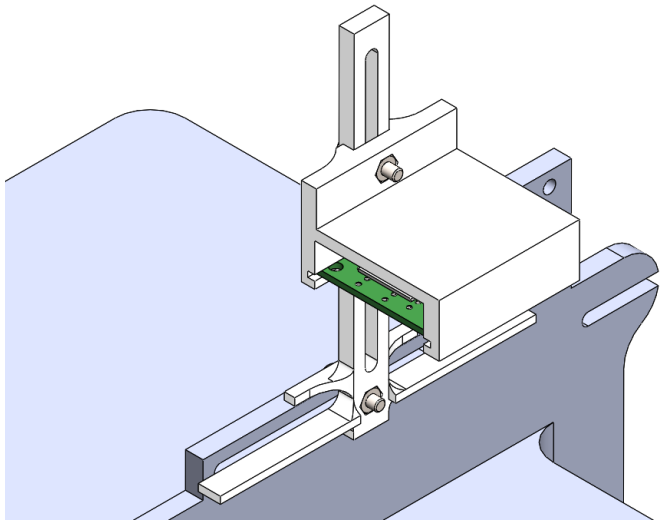
4.3 Nosač kamere

Veliku ulogu u određivanju kvalitete zrna pomoću softvera igra kvaliteta slike. Budući da je odabrano računalo *Raspberry Pi 4* zbog kompaktnosti, pristupačne cijene, mogućnosti spajanja te pružanja napona od 5 V za električne komponente, odabrana je *Raspberry Pi Camera Module 3* kamera.

Slika 15: Kamera [11]

Spomenuta kamera koristi 12 megapixel-ni *Sony IMX708* senzor te su joj glavne prednosti, uz pristupačnu cijenu, mogućnost autofokusa, visok dinamički raspon te visoka osjetljivost prilikom niske razine osvjetljenja. Budući da tijekom konstruiranja držača kamere nije bila poznata optimalna pozicija kamere, pristupilo se izradi držača kamere s dva stupnja slobode gibanja. Kamera se može pomicati duž konvejera te po vertikalnoj osi. Budući da se kamera pomiče po potrebi, a ne konstantno te da bi se dobio kompaktiniji dizajn, nisu korišteni ležajevi. Na bočnoj stranici konvejera napravljen je utor kroz kojeg prolazi tijelo vijka, a u vertikalnom stupu nalazi se matica čije je okretanje spriječeno oblikom stoga je

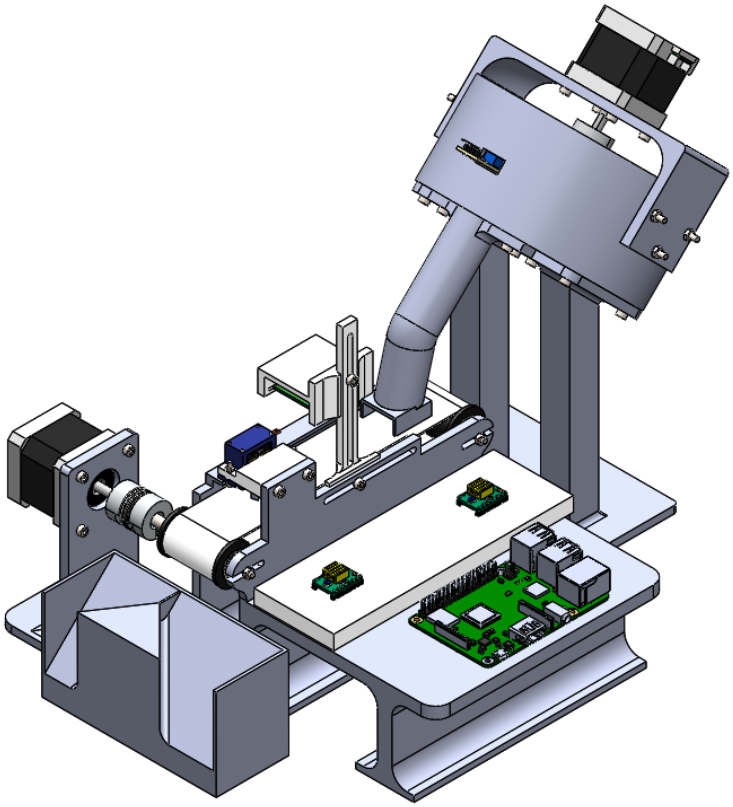
osiguranje horizontalne pozicije vertikalnog stupa (a time i kamere) postignuto pritiskom kojeg tvore vijak i matica. Za stvaranje jasnijeg dojma, radionički crteži stupa te držača kamere vidljivi su u prilogu. Na vertikalnom stupu nosača također je napravljen utor kroz kojeg prolazi vijak te je matica umetnuta u držač kamere što omogućava pomicanje kamere po vertikalnoj osi u otpuštenom stanju, odnosno osiguranje pozicije u zategnutom stanju. Kamera se u držač umetne klizanjem po utoru. Na sljedećoj slici vidljiv je izometrijski pogled nosača kamere zajedno s dijelom bočne stranice konvejera.



Slika 16: Nosač kamere

4.4 Ostatak sustava

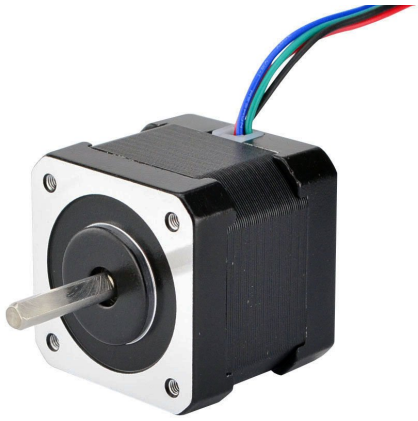
Do sada je opisan način rada ispusnog mehanizma, konvejera te mehanizma kamere. Uz to, konstruiran je i držač servo motora koji se pričvršćuje za bočnu stranicu konvejera vijcima. Na poslijetku konstruirana je pozicija koja dijeli zrno na dvije različite strane. Na sljedećoj slici vidljiv je potpuni sustav uključujući električne komponente.



Slika 17: Cjelokupni sustav

5 UPRAVLJANJE SUSTAVOM

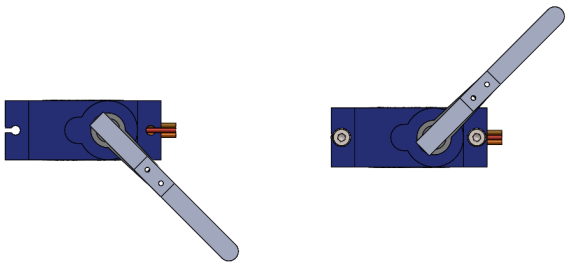
U ovom poglavlju opisano je upravljanje sustavom iz programskog te hardverskog aspekta. Već je ranije spomenuto kako su konvejer te rotirajući disk za izdvajanje zrna pokretani koračnim motorima. Razlog je velik moment motora i mogućnost programskog upravljanja brzinom pulsko širinsko modulacijom. Odabrani su motori NEMA 17. Oba su upravljana pomoću A4988 upravljača čija je prednost jednostavna povezivost.



Slika 18: NEMA 17 koračni motor [13]

Prilikom opisivanja tehničkog crteža bubnja spomenut je sustav protiv zaglavljivanja. On radi pomoću Hallovog senzora koji očitava vrijednost magnetskog polja u blizini. Prednosti Hallovog senzora uključuju brzo očitavanje, visoku preciznost, nisku potrošnju energije i dug vijek trajanja. Oni su također otporni na vibracije i imaju širok raspon radnih temperatura. U rotirajući disk umetnuta su 2 magneta, stoga tijekom jednog punog okreta vrijednost napona očitanja na senzoru prijeđe iz niske u visoku. U upravljačkom programu vidljivom u prilogu (Skripta 3: Upravljanje) kontinuirano se provjerava vrijeme proteklo

od zadnjeg prolaska magneta ispod senzora. Ukoliko je ta vrijednost veća od nekoliko sekundi znači da je došlo do zaglavljenja diska uslijed zaglavljenja zrna graha negdje unutar bubnja (npr. kod pozicije za blokiranje zrna). Da bi se otklonilo zaglavljenje disk se na trenutak okrene u suprotnu stranu kako bi pritisak popustio te se smjer rotacije vraća na standardni. Upravljački sustav konstantno snima stanje konvejera te daje informaciju o objektima na njemu. Kako bi to bilo moguće, model dubokog učenja pretvoren je u *Tensorflow lite* strukturu koja je razvijena od strane *Google*-a za izvršavanje aplikacija dubokog učenja na manjim računalima bez prevelikih zauzimanja procesorskih resursa. Model u izlaznom sloju sadrži neuron koji daje informaciju nalazi li se na slici dobar grah (vrijednost izlaza manja od 0,33), loš grah (vrijednost izlaza veća od 0,66) ili je na slici samo traka konvejera (vrijednost izlaza veća od 0,33 i manja od 0,66). Ukoliko sustav pruži informaciju da se ispod kamere trenutno nalazi dobar grah, pozicija servo motora postavlja se na 45 stupnjeva, a ukoliko je na slici loš grah pozicija se postavlja na 135 stupnjeva. Na taj način lopatice servo motora prebacuju zrno na odgovarajuće polovice konvejerske trake te zrna na poslijetku padaju na odgovarajuće strane na ispustnu poziciju. Odabran je servo motor SG90. Na sljedećoj slici vidljive su spomenute dvije pozicije servo motora sa spojenim lopaticama.



Slika 19: Pozicije servo motora

Valja napomenuti kako motor konvejera radi neprestano. Budući da se motori moraju

okretati istovremeno, u upravljačkoj skripti stvorena je dretva koja omogućava da se funkcija *run_conveyor* odvija paralelno s glavnom petljom. U spomenutoj funkciji na STEP kanal upravljača motora konvejera šalje se signal pravokutnog oblika. Nadalje, signal istog oblika, no različite frekvencije šalje se na isti kanal upravljača motora diska unutar funkcije *run_disk* koja se također odvija paralelno s izvođenjem glavne petlje, jedina razlika je što se tamo provjerava vrijednost varijable *dir1* koja određuje smjer vrtnje diska. Uz već spomenutu upravljačku skriptu, u prilogu je također prikazana i shema spajanja električnih komponenti.

5.1 Usklađivanje brzina motora

Zahtjev za sortirnicu je što veći kapacitet sortiranja, odnosno maksimizacija broja zrna koja se sortira u vremenu. Kako bi se to ostvarilo važno je uskladiti brzine okretanja motora diska i motora konvejera. Budući da je udaljenost kamere varijabilna, u ovoj fazi planiranja izabrana je udaljenost između ispusne cijevi i središta kamere po osi konvejera od 26 mm što rezultira udaljenošću središta kamere do vrha lopatice servo motra od 21 mm što je važno budući da će servo motor zahvatiti sljedeće zrno graha do vremena kada se prethodno nađe na ispod središta kamere. Također, zahtjev (zbog pouzdanijeg rada mreže) je da sljedeće zrno izađe iz ispusne cijevi kada se prethodno nalazi točno vertikalno ispod središta kamere. Dakle, zahtjev da konvejer prijeđe put od 26 mm u periodu između dva pada zrna iz konvejera. Budući da bubanj ima 6 rupa, period padanja zrna T_g određen je sljedećim izrazom:

$$T_g = \frac{1}{6 \cdot n_d} \tag{9}$$

gdje je n_d broj okretaja motora konvejera u sekundi. Nadalje, brzina kretanja v_r remena ovisi o okretajima motora konvejera u sekundi n_r na sljedeći način (23 mm predstavlja

srednji promjer remena):

$$v_r = 23\text{mm} \cdot \pi \cdot n_r \tag{10}$$

Prema tome, vrijeme da remen prijeđe 26 mm iznosi:

$$t = \frac{26}{23 \cdot \pi \cdot n_r} \tag{11}$$

Na kraju, kako bi se dobili odnosi brzina potrebno je izjednačiti izraze (9) i (11) budući da je zahtjev da ta vremena budu jednaka, time se dobiva razmjer:

$$n_r = \frac{6 \cdot 26 \cdot n_d}{23 \cdot \pi} = 2,16\ n_d \tag{12}$$

Dakle, da bi motori radili po zahtjevima, brzina vrtnje motora konvejera mora biti 2,16 puta veća od brzine motora diska.

6 ZAKLJUČAK

U ovom radu prikazani su svi koraci potrebni za izradu automatirizirane sortirnice za grah te su potkrepljeni kodovima, električnim shemama te tehničkim crtežima. Valja napomenuti kako bi nakon fizičke izrade sortirnice vjerojatno bilo potrebno pristupiti iterativnom usavršavanju rada stroja, kako u dizajnu izrađenih komponenti, tako u programskom kodu modela dubokog učenja te usklađivanju rada električnih komponenti. Ovakav uređaj potpuno bi mogao zamijeniti rad čovjeka, a cijena za izradu je približno 300 €, a obzirom na podatke iz uvoda, prosječno gospodarstvo postiglo bi povrat investicije već u prvoj godini čime je ostvaren cilj za isplativim uređajem. Trenutni nedostatak sortirnice je mala dimenzija ulaznog bubnja stoga bi krajnji korisnik morao ostvariti prilagođeni usipnik kojim bi nova zrna konstantno tekla u bubanj. U odnosu na postojeća rješenja na tržištu, nedostatak ovog uređaja je kapacitet sortiranja, no veći kapacitet ne može se ostvariti bez drastičnog povećanja cijene uređaja koji bi u tom slučaju koristio serije brzih industrijskih kamera visoke rezolucije uz pneumatske aktuatore te sustav osvjetljenja. Prednost ovog uređaja u odnosu na postojeća rješenja na tržištu su male gabaritne dimenzije te znatno manja masa. Također, svi dijelovi lako su dostupni za održavanje te montažu i demontažu.

7 LITERATURA

1. Metak sortirnica - https://www.metakcolorsorter.com/color-sorter-machine-rcz.html

2. Godec D.; Šercer M.; Aditivna proizvodnja; FSB; Zagreb, 2015.

3. Gibson I. et al.; Additive Manufacturing Technologies; Springer; New York; 2010.

4. Tensorflow dokumentacija - https://www.tensorflow.org/api_docs - 16.02.2023.

5. Keras dokumentacija - https://keras.io/api/ - 16.02.2023.

6. XML standard https://www.w3.org/standards/xml/core - 16.02.2023.

7. CSV format - https://dev.socrata.com/docs/formats/csv.html - 16.02.2023.

8. VGG16 model - https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918 - 16.02.2023.

9. Simonyan K., Zisserman A., Very deep convolutional network for large-scale image recognition, ICLR, 2015.

10. Autodesk Eagle podrška https://www.autodesk.com/support/technical/product/eagle?sort=score - 15.06.2023.

11. Raspberry Pi dokumentacija https://www.raspberrypi.com/documentation/ - 14.04.2023.

12. Gospodarski list - isplativa poljoprivredna proizvodnja https://gospodarski.hr/rubrike/agroekonomika/isplativa-poljoprivredna-proizvodnja/

13. NEMA 17 dokumentacija - https://components101.com/motors/nema17-stepper-motor - 15.06.2023

8 PRILOG

Skripta 1: Pretvaranje XML elemenata u CSV datoteku

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET



def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            if member[0].text == 'dobar':
                klasa = 0
            if member[0].text == 'nista':
                klasa = 0.5
            if member[0].text == 'los':
                klasa = 1
            value = (root.find('filename').text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
```

```
                    int(member[4][3].text),
                    klasa
                    )
            xml_list.append(value)
    column_name = ['filename',  'xmin', 'ymin', 'xmax', 'ymax','class']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    image_path = os.path.join('Tensorflow','workspace','images','train')
    xml_df = xml_to_csv(image_path)
    xml_df.to_csv('grahTest.csv', index=None)
    print('Successfully converted xml to csv.')

main()
```

Skripta 2: Postav modela, treniranje i evaluacija

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import img_to_array
```

```
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.models import load_model
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import mimetypes
import argparse
import imutils
import cv2
import os

def plt_imshow(title, image):

    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.imshow(image)
    plt.title(title)
    plt.grid(False)
    plt.show()

class Config:
    BASE_PATH = os.path.join('BoundingBoxes','dataset')
    IMAGES_PATH = os.path.sep.join([BASE_PATH, "images"])
    ANNOTS_PATH = os.path.sep.join([BASE_PATH, "grah.csv"])

    BASE_OUTPUT = os.path.join('BoundingBoxes','output')
```

```
    MODEL_PATH = os.path.sep.join([BASE_OUTPUT, "detector.h5"])
    PLOT_PATH = os.path.sep.join([BASE_OUTPUT, "plot.png"])
    TEST_FILENAMES = os.path.sep.join([BASE_OUTPUT, "test_images.txt"])


    INIT_LR = 1e-4
    NUM_EPOCHS = 50
    BATCH_SIZE = 32

config = Config()

rows = open(config.ANNOTS_PATH).read().strip().split("\n")

data = []
targets = []
filenames = []

for row in rows:

    row = row.split(",")
    (filename, startX, startY, endX, endY, klasa) = row
    imagePath = os.path.sep.join([config.IMAGES_PATH, filename])
    image = cv2.imread(imagePath)
    (h, w) = image.shape[:2]
```

```
    startX = float(startX) / w
    startY = float(startY) / h
    endX = float(endX) / w
    endY = float(endY) / h

    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)

    data.append(image)
    targets.append((startX, startY, endX, endY, klasa))
    filenames.append(filename)

data = np.array(data, dtype="float32") / 255.0
targets = np.array(targets, dtype="float32")

split = train_test_split(data, targets, filenames, test_size=0.10,
    random_state=42)

(trainImages, testImages) = split[:2]
(trainTargets, testTargets) = split[2:4]
(trainFilenames, testFilenames) = split[4:]

f = open(config.TEST_FILENAMES, "w")
f.write("\n".join(testFilenames))
f.close()
```

```
vgg = VGG16(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

vgg.trainable = False

flatten = vgg.output
flatten = Flatten()(flatten)

bboxHead = Dense(128, activation="relu")(flatten)
bboxHead = Dense(64, activation="relu")(bboxHead)
bboxHead = Dense(32, activation="relu")(bboxHead)
bboxHead = Dense(5, activation="sigmoid")(bboxHead)

model = Model(inputs=vgg.input, outputs=bboxHead)


opt = Adam(lr=config.INIT_LR)
model.compile(loss="mse", optimizer=opt)
print(model.summary())

H = model.fit(
    trainImages, trainTargets,
    validation_data=(testImages, testTargets),
    batch_size=config.BATCH_SIZE,
    epochs=config.NUM_EPOCHS,
    verbose=1)
```

```
model.save(config.MODEL_PATH, save_format="h5")

N = config.NUM_EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.title("Bounding Box Regression Loss on Training Set")
plt.xlabel("Epoch #")
plt.ylabel("Loss")
plt.legend(loc="lower left")
plt.show()

args = {
    "input": "BoundingBoxes/output/test_images.txt"
    #"input":os.path.join('Tensorflow','workspace','images','test','Alter.jpg')
}

import mimetypes

filetype = mimetypes.guess_type(args["input"])[0]
imagePaths = [args["input"]]

groundTruthBoxes = []
realClasses = []
```

```
if "text/plain" == filetype:
    filenames = open(args["input"]).read().strip().split("\n")
    imagePaths = []

    for f in filenames:
        p = os.path.sep.join([config.IMAGES_PATH, f])
        imagePaths.append(p)

        image_name = f
        matches = [elem for elem in rows if image_name in elem][0].split(",")
        (name, xmin,ymin, xmax,ymax,klasa) = matches
        groundTrouthBox =  [int(xmin),int(ymin), int(xmax),int(ymax)]
        groundTruthBoxes.append(groundTrouthBox)
        realClasses.append(int(klasa))

predictedBoxes = []
predictedClasses= []

model = load_model(config.MODEL_PATH)


i = 0
for imagePath in imagePaths:

    image = load_img(imagePath, target_size=(224, 224))
```

```
image = img_to_array(image) / 255.0
image = np.expand_dims(image, axis=0)

preds = model.predict(image)[0]
print(preds)
(startX, startY, endX, endY, klasa) = preds

if round(klasa) == 0:
    strKlasa = "dobar"
else:
    strKlasa = "los"
klasa = format(klasa, '.2f')

image = cv2.imread(imagePath)
#image = imutils.resize(image, width=600)
(h, w) = image.shape[:2]

startX = int(startX * w)
startY = int(startY * h)
endX = int(endX * w)
endY = int(endY * h)

predictedBox = [startX,startY,endX,endY]
predictedBoxes.append(predictedBox)

cv2.rectangle(image, (startX, startY), (endX, endY),
```

```
        (0, 255, 0), 2)
    cv2.rectangle(image, (int(groundTruthBoxes[i][0]),int(groundTruthBoxes[i][1])), (
        (0, 255, 0), 2)
    coordinates = (startX,startY-5)
    cv2.putText(image,strKlasa + "  "+str(klasa),coordinates,cv2.FONT_HERSHEY_SIMPLEX

    cv2.imwrite("output"+str(i)+".jpg",image)

    predictedClasses.append(round(float(klasa)))
    i+=1

def bbox_iou(boxA, boxB):

    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])

    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)

    iou = interArea / float(boxAArea + boxBArea - interArea)

    return iou
```

```
import statistics

AP_dobar_5_95 = []
AP_los_5_95 = []

for countr in range (50,100,5):
    IoUTreshold = countr/100
    TP_dobar = 0
    TP_los = 0

    FP_dobar = 0
    FP_los = 0

    FN_dobar = 0
    FN_los = 0

    TN_dobar = 0
    TN_los = 0

    for k in range (len(realClasses)):
        if realClasses[k] - predictedClasses[k] != 0:
            if realClasses[k] == 0:
                FN_dobar += 1
                FP_los += 1
            else:
```

```
                FN_los += 1
                FP_dobar += 1
        else:
            IoU = bbox_iou(groundTruthBoxes[k],predictedBoxes[k])
            if realClasses[k] == 0:
                if IoU > IoUTreshold:
                    TP_dobar += 1
                else:
                    FP_dobar += 1
                TN_los += 1
            else:
                if IoU > IoUTreshold:
                    TP_los += 1
                else:
                    FP_los += 1
                TN_dobar += 1
    AP_dobar = TP_dobar/(TP_dobar+FP_dobar)
    AP_los = TP_los/(TP_los+FP_los)

    AP_dobar_5_95.append(AP_dobar)
    AP_los_5_95.append(AP_los)
    print(AP_dobar,IoUTreshold)

mAP_dobar = statistics.mean(AP_dobar_5_95)
mAP_los = statistics.mean(AP_los_5_95)
print((mAP_dobar+mAP_los)/2)
```

Skripta 3: Upravljanje

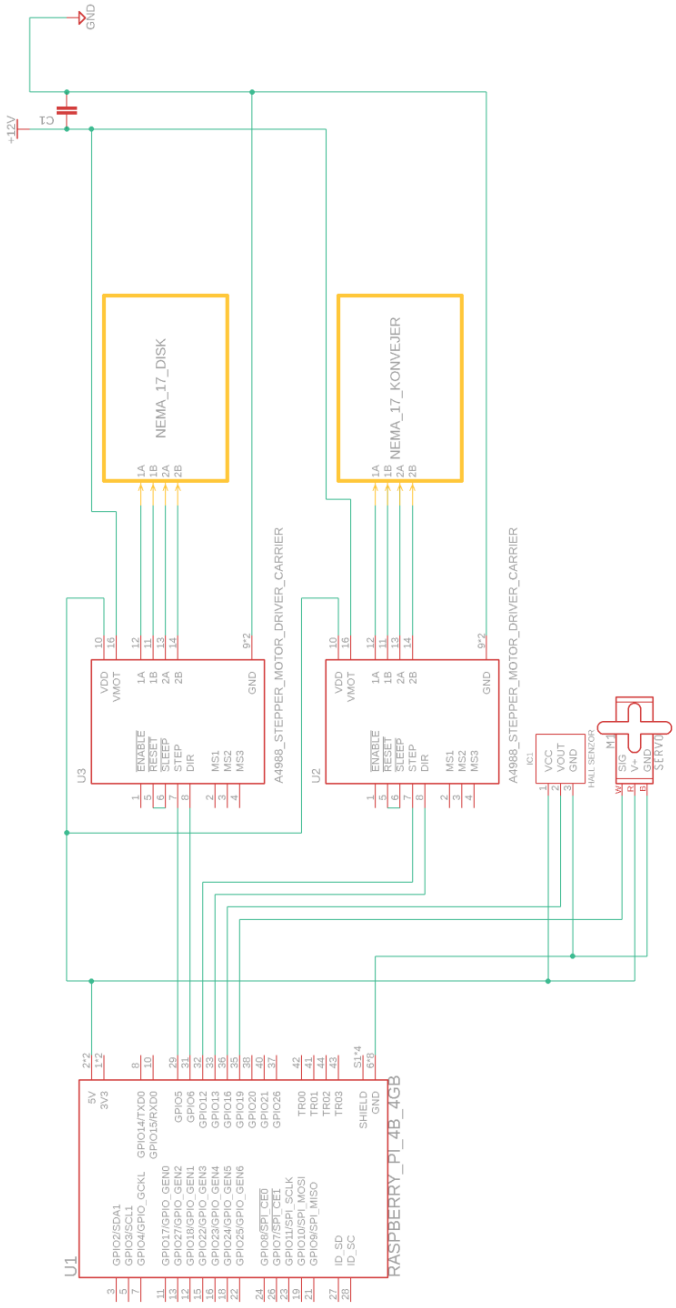
```
import RPi.GPIO as GPIO
from gpiozero import Servo
import time
import cv2
import numpy as np
import tflite_runtime.interpreter as tflite
import threading

# Mreza
model_path = "model.tflite"
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

camera = cv2.VideoCapture(0)

# Step motors
# 6 - DIR1, 5 - STEP1, 13 - DIR2, 12 - STEP2
GPIO.setup(5, GPIO.OUT)
GPIO.setup(6, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
```

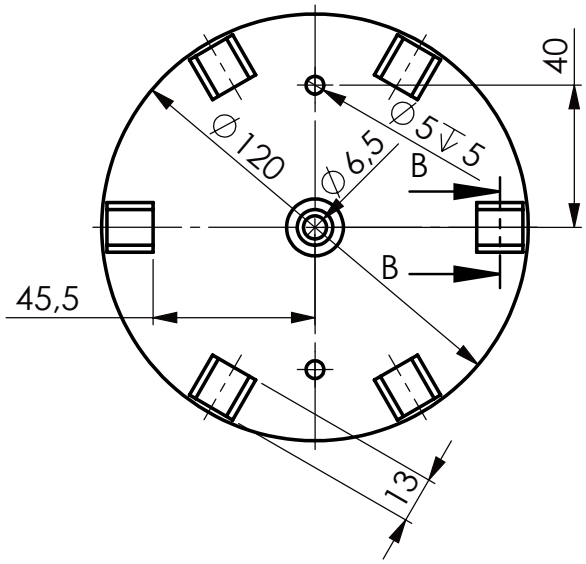
```
# Hall
GPIO.setup(16, GPIO.IN)
hall_output = GPIO.input(16)

# Servo
servo = Servo(19)

def run_conveyor():
    GPIO.output(13, GPIO.HIGH)
    while True:
        GPIO.output(12, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(12, GPIO.LOW)
        time.sleep(0.1)

def run_disk():
    if dir1 == 'CW':
        GPIO.output(6, GPIO.HIGH)
    elif dir1 == 'CCW':
        GPIO.output(6, GPIO.LOW)
    while True:
        GPIO.output(5, GPIO.HIGH)
        time.sleep(0.216)
        GPIO.output(5, GPIO.LOW)
```

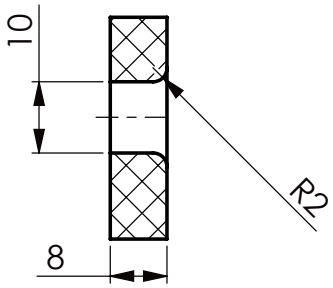
```
        time.sleep(0.216)

global dir1
dir1 = 'CW'

threading.Thread(target=run_conveyor).start()
threading.Thread(target=run_disk).start()

while True:
    if hall_output == 1:
        hall_time = time.time()

    elapsed = time.time() - hall_time
    if elapsed > 6:
        dir1 = 'CCW'
        hall_time = time.time()
    else:
        dir1 = 'CW'

    ret, frame = camera.read()
    input_shape = input_details[0]['shape']
    input_frame = cv2.resize(frame, (input_shape[1], input_shape[2]))
    input_frame = np.expand_dims(input_frame, axis=0)
    input_frame = input_frame.astype(np.float32)

    interpreter.set_tensor(input_details[0]['index'], input_frame)
```

```
    interpreter.invoke()

    output = interpreter.get_tensor(output_details[0]['index'])

    if output < 0.33:
        servo.value = -0.5

    elif output > 0.66:
        servo.value = 0.5

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

camera.release()
GPIO.cleanup()
```

Shema spajanja električnih komponenti

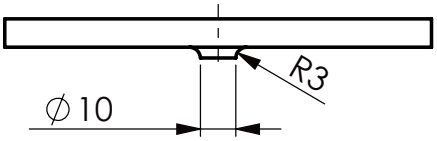


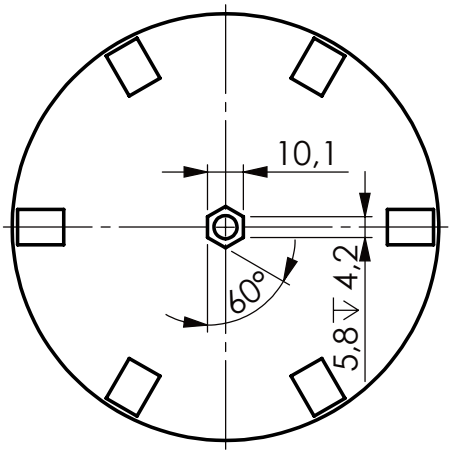




B-B (1 : 1)

F



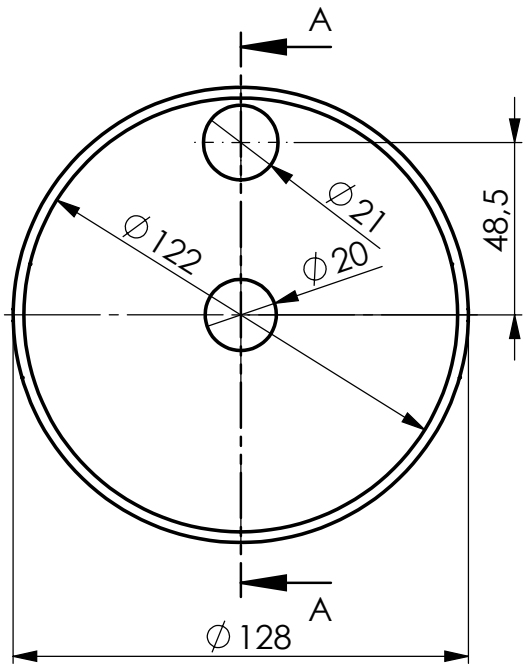


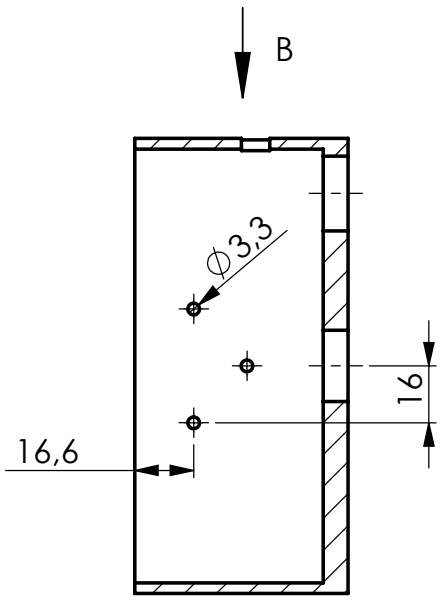
pogled F

	Datum	Ime i prezime	Potpis	FSB Zagreb
Projektirao	06.2023.	Ivan Maletić		
Razradio	06.2023.	Ivan Maletić		
Crtao	06.2023.	Ivan Maletić		
Pregledao				

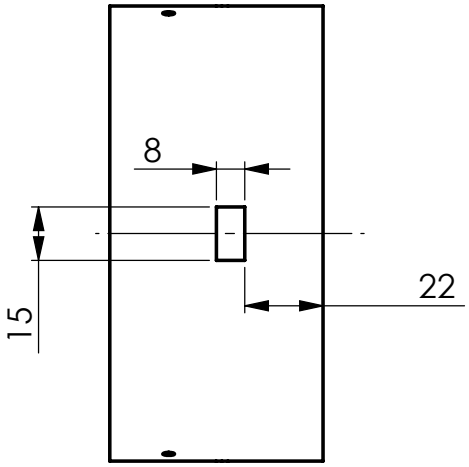
Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala 1:2	Naziv: Disk / Pozicija: / Format: A4 / Listova: 1 / List: 1
	Crtež broj: DR-01-001

Design by CADLab





A-A



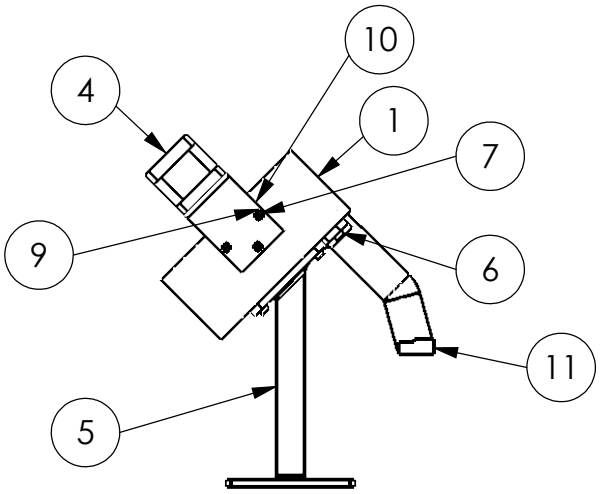
pogled B

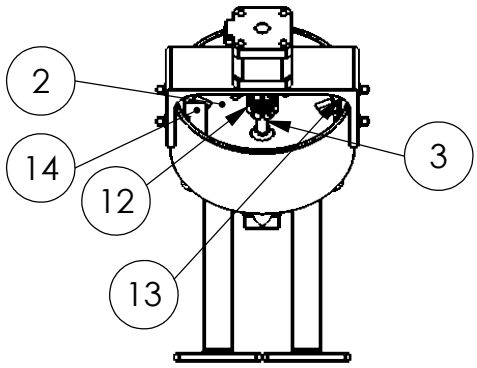
	Datum	Ime i prezime	Potpis
Projektirao	06.2023.	Ivan Maletić	
Razradio	06.2023.	Ivan Maletić	
Crtao	06.2023.	Ivan Maletić	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala 1:2	Naziv: Bubanj / Pozicija: / Format: A4 / Listova: 1 / List: 1
	Crtež broj: DR-01-002

Design by CADLab





Pozicija	Naziv	Komada
1	Bubanj	1
2	Disk	1
3	DIN EN 24015 - M6 x 35 x 18-N	1
4	Koračni motor	1
5	Postolje	2
6	ISO 4762 M3 x 8 - 8N	10
7	EN ISO 4762 M3 x 20 - 18N	6
8	EN ISO 4762 M3 x 12 - 12N	4
9	ISO 4032 - M3 - W - N	6
10	Držač motora	1
11	ispust	1
12	Spojka	1
13	poravnanje	2
14	poravnanje i bloker	1

	Datum	Ime i prezime	Potpis
Projektirao	06.2023.	Ivan Maletić	
Razradio	06.2023.	Ivan Maletić	
Crtao	06.2023.	Ivan Maletić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena:
Čeone plohe diska i bubnja udaljiti 1 mm

Kopija

Materijal:

Masa:

Naziv: Izuzimanje sklop

Pozicija:

Format: A4

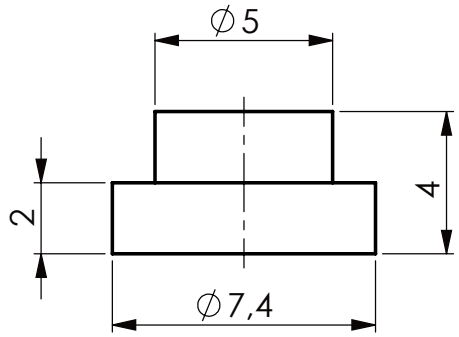
Mjerilo originala
1:5

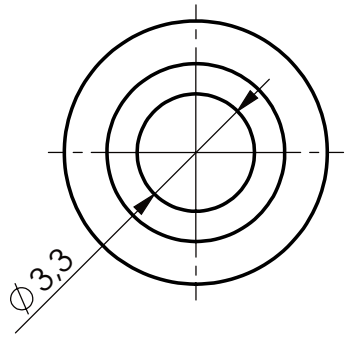
Listova: 1

Crtež broj: DR-04-001

List: 1

Design by CADLab

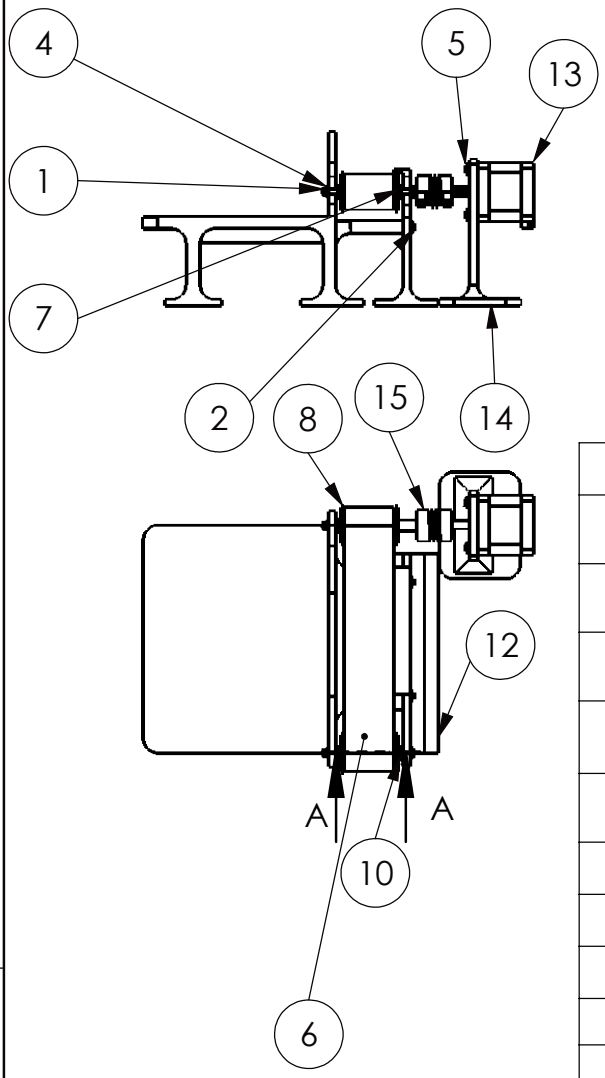


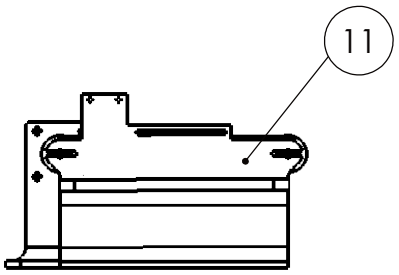


	Datum	Ime i prezime	Potpis	
Projektirao	06.2023.	Ivan Maletić		FSB Zagreb
Razradio	06.2023.	Ivan Maletić		
Crtao	06.2023.	Ivan Maletić		
Pregledao				

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala 5:1	Naziv: Centriranje ležaja / Pozicija: / Format: A4 / Listova: 1 / List: 1
	Crtež broj: DR-01-003

Design by CADLab





Pozicija	Naziv	Komada
1	DIN 912 M3 x 20 --- 20N	3
2	DIN 912 M3 x 10 --- 10N	2
3	DIN EN 24015 - M6 x 25 x 18-N	1
4	ISO 4032 - M3 - W - N	3
5	EN ISO 4762 M3 x 12 - 12N	4
6	Remen	1
7	Bubanj	1
8	Bubanj motora	1
9	Ležaj 625	3
10	Centriranje	6
11	Desna bočna ploča	1
12	Lijeva bočna ploča	1
13	Nema 17 Stepper Motor v5.step	1
14	Držač motora konvejera	1
15	Elastična spojka	1

	Datum	Ime i prezime	Potpis
Projektirao	06.2023.	Ivan Maletić	
Razradio	06.2023.	Ivan Maletić	
Crtao	06.2023.	Ivan Maletić	
Pregledao			

FSB Zagreb

Objekt:

Objekt broj:

R. N. broj:

Napomena:

Kopija

Materijal:

Masa:

Mjerilo originala

1:5

Naziv:

Konvejer

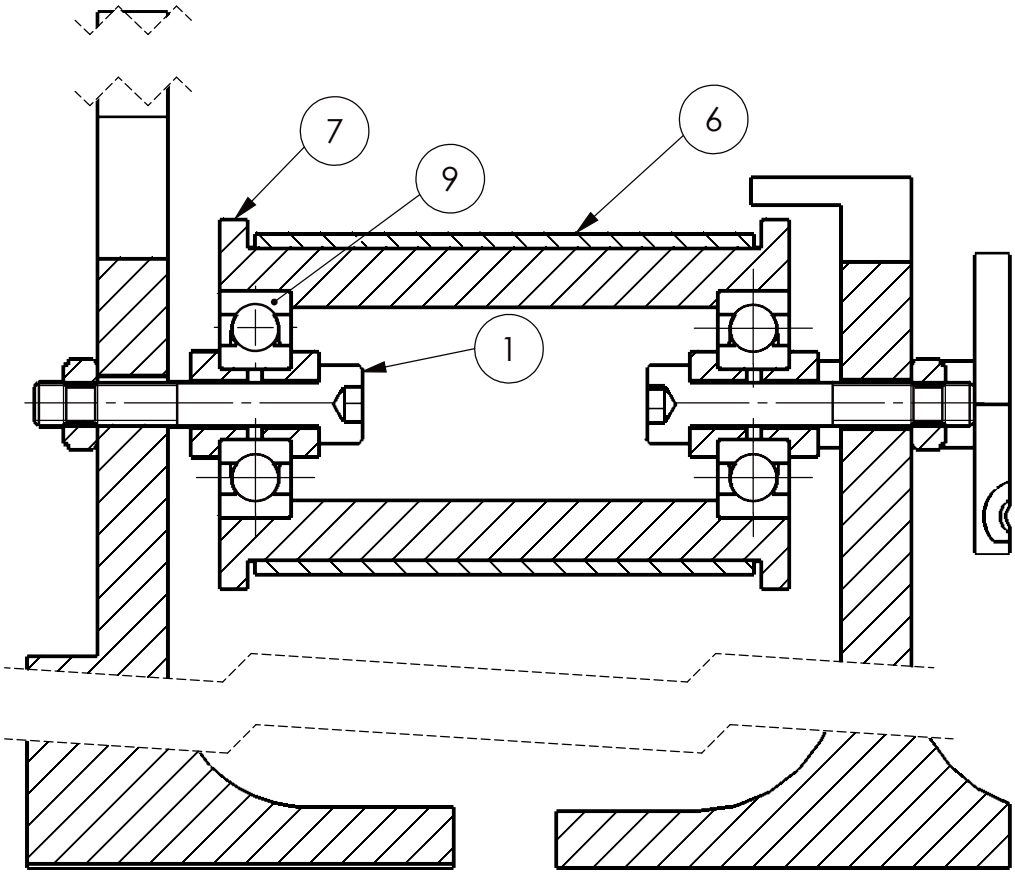
Pozicija:

Format: A4

Listova: 2

Crtež broj: DR-04-002

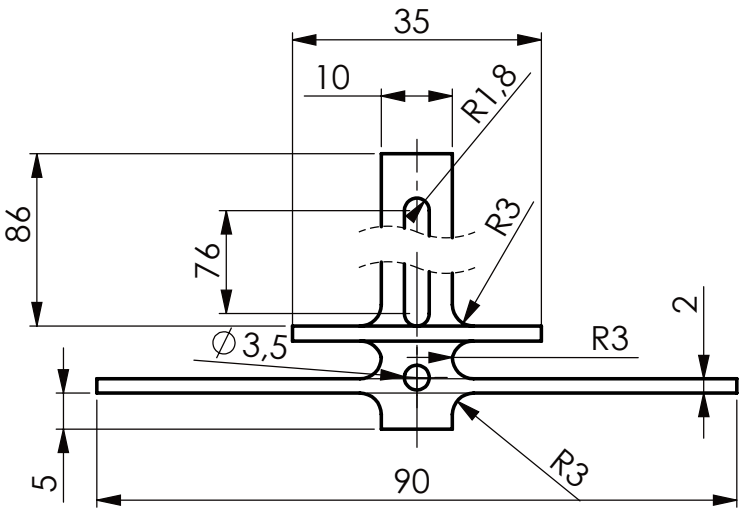
List: 1

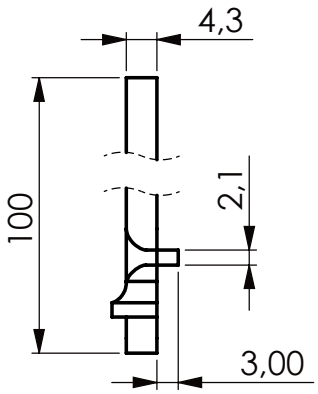


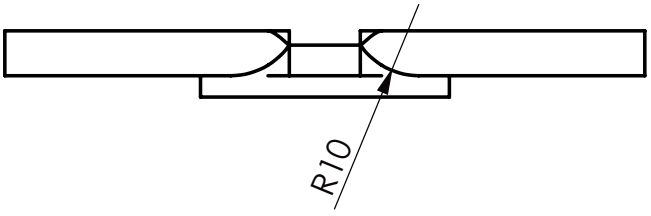
A-A (2 : 1)

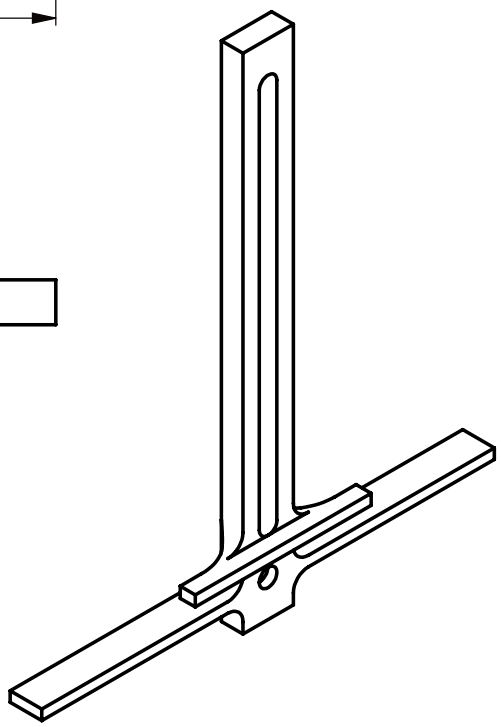
	Datum	Ime i prezime	Potpis	
Projektirao	06.2023.	Ivan Maletić		FSB Zagreb
Razradio	06.2023.	Ivan Maletić		
Crtao	06.2023.	Ivan Maletić		
Pregledao				

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija
Materijal:	Masa:	
Mjerilo originala 1:5	Naziv: Konvejer	Pozicija:
	Crtež broj: DR-02-003	Format: A4
		Listova: 2
		List: 2







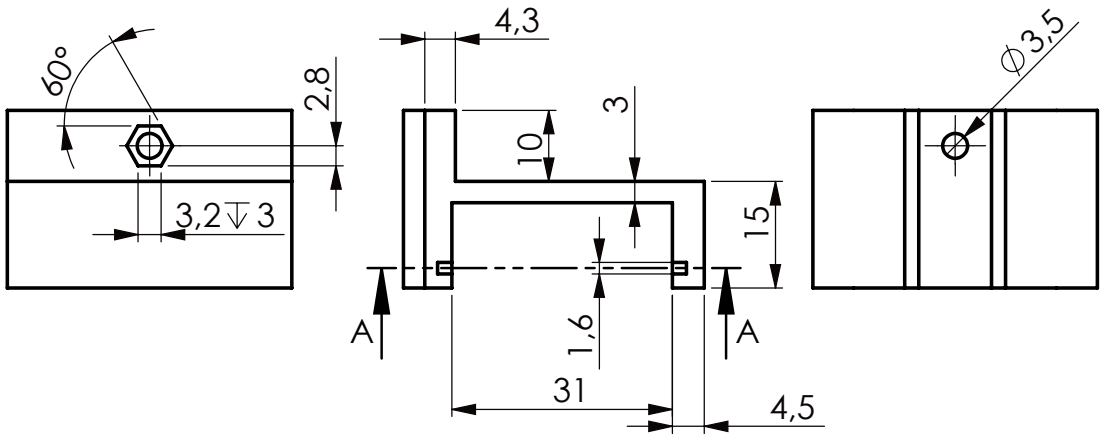


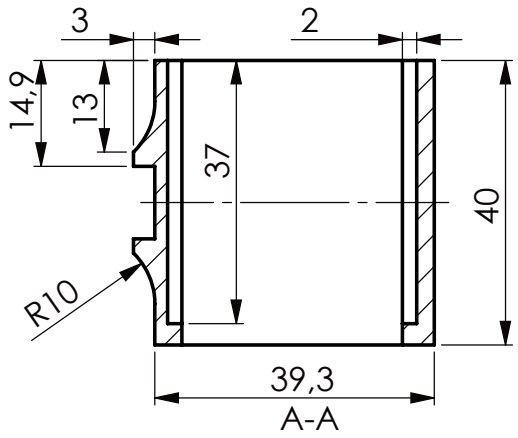
	Datum	Ime i prezime	Potpis
Projektirao	06.2023.	Ivan Maletić	
Razradio	06.2023.	Ivan Maletić	
Crtao	06.2023.	Ivan Maletić	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala 1:1	Naziv: Vertikalni stup kamere / Pozicija:
	Format: A4
	Listova: 1
Crtež broj: DR-01-004	List: 1

Design by CADLab





	Datum	Ime i prezime	Potpis
Projektirao	06.2023.	Ivan Maletić	
Razradio	06.2023.	Ivan Maletić	
Crtao	06.2023.	Ivan Maletić	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena:	Kopija
Materijal: / Masa:	
Mjerilo originala 1:1	Naziv: Držač kamere / Pozicija: / Format: A4 / Listova: 1 / List: 1
	Crtež broj: DR-01-005

Design by CADLab