Idejno rješenje za unapređenje postojećeg postava montaže protupožarnih aparata

Keretić, Damian

Master's thesis / Diplomski rad

2021

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:200092

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Damian Keretić

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

doc. dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Damian Keretić

Zagreb, 2021.

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

Zahvaljujem se mentoru doc.dr.sc. Tomislavu Stipančiću i njegovom mladom istraživaču Leonu Korenu na pomoći i sugestijama pruženim tijekom izrade ovog rada.

Zahvaljujem se cijeloj svojoj obitelji, posebno roditeljima koji su mi uvijek bili velika podrška.

Svojoj djevojci Eleni koja mi je uvijek pružala potporu i bila tu za mene.

Svojim prijateljima i kolegama s faksa s kojima sam sve ovo zajedno prolazio.

I na kraju se posebno zahvaljujem tvrtki Pastor TVA koja mi je omogućila da za njih radim na ovom projektu, direktoru Martiću, svima u tehničkom uredu i proizvodnji na strpljenju, pomoći i prijenosu znanja.

Damian Keretić

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602-04/21-6/1	
Ur. broj: 15-1703-21	

DIPLOMSKI ZADATAK

Student: **DAMIAN KERETIĆ** Mat. br.: 0035210157

Naslov rada na hrvatskom jeziku: **Idejno rješenje za unapređenje postojećeg postava montaže protupožarnih aparata**

Naslov rada na engleskom jeziku: **Conceptual solution for the improvement of the existing assembly installation for fire extinguishers**

Opis zadatka:

Završna montaža vatrogasnih aparata veoma je složen i fizički naporan posao. Trenutno u sklopu proizvodnog procesa tvrtke Pastor TVA na njemu radi 7 radnika. Humanizacija rada zahtijeva da se ljudska radna snaga zamjeni umjetnom kod svih poslova koji se mogu dovesti u direktnu vezu s narušavanjem zdravlja.

U radu je potrebno predložiti koncept automatizacije završne montaže protupožarnih aparata koji se razlikuju u veličini, težini te tipu korištenog ventila/zatvarača.

Rješenje treba sadržavati:

- analizu postojećeg postava s identificiranim slabim mjestima na kojima je moguće predložiti poboljšanja
- nacrte, modele, simulacije i programe koji će vizualizirati zamišljeni proces
- prijedlog rješenja za računalnu podršku koja omogućuje sinkroni rad te komunikaciju vitalnih komponenti sustava
- prijedlog rješenja korisničkog sučelja za brzu promjenu parametara montažne linije u slučaju promijene tipa aparata
- prijedlog rješenje za sustav praćenja životnog vijeka protupožarnog aparata koje je temeljeno na barcode naljepnicama.

Prema predloženim rješenjima potrebno je prikazati okvirnu financijsku analizu i izračunati isplativost cijelog projekta.
U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan: 30. rujna 2021.

Rok predaje rada: 2. prosinca 2021.

Predviđeni datum obrane: 13. prosinca do 17. prosinca 2021.

Zadatak zadao: doc. dr. sc. Tomislav Stipančić

Predsjednica Povjerenstva: prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Mjerna jedinica	Opis oznake
m	kg	- masa
S	-	- faktor sigurnosti
a	$\frac{\mathrm{m}}{\mathrm{s}^2}$	- akceleracija
g	$\frac{\mathrm{m}}{\mathrm{s}^2}$	- gravitacija
μ	-	- faktor trenja
F_H	N	- sila držanja
F_S	N	- sila usisa
d_c	mm	- promjer kapice
$F_{S,cap}$	N	- usisna sila kapice
V	m^3	- volumen
$d_{crijevo}$	mm	- promjer crijeva
d_{pot}	mm	- potrebni promjer kapice
V_s	m^3	- potrebni volumen po kapici
V_{uk}	m^3	- potrebni ukupni volumen kapica
$d_{mlaznice}$	mm	- promjer mlaznice
V_{max}	m^3	- maksimalni volumen generatora
Air_{cons}	$\frac{\mathrm{l}}{\mathrm{min}}$	- potrošnja zraka
p	bar	- tlak
d_c	mm	- preporučeni unutarnji promjer crijeva
D_c	mm	- preporučeni vanjski promjer crijeva
z	mm	- hod nosača
V_{P6}	m^3	- volumen aparata P6
V_{N2}	m^3	- volumen dušika
φ	$\frac{\mathrm{kg}}{\mathrm{m}^3}$	- gustoća
M	$\frac{\mathrm{g}}{\mathrm{mol}}$	- molarna masa
ϑ	°C	- temperatura u °C
T	K	- temperatura u Kelvinima
R	$\frac{\mathrm{J}}{\mathrm{mol\,K}}$	- univerzalna plinska konstanta
$C_{N2/He,\ P6}$	$\frac{\mathrm{kn}}{\mathrm{aparat}}$	- cijena u kunama po aparatu
C_{uk}	$\frac{\mathrm{kn}}{\mathrm{god}}$	- cijena po godini
n	mol	- množina tvari
y	-	- volumni udio

SAŽETAK

U ovom diplomskom radu odrađen je koncept, ideja i razrada automatizacije završne montaže vatrogasnih aparata u tvtki Pastor TVA. Radi se o najvećem proizvođaču aparata za gašenje požara u ovom dijelu Europe s dugom tradicijom od više od 90 godina postojanja. S obzirom da je tvtka ušla u razdoblje investicija u proizvodni pogon i nove strojeve, odlučeno je da dio toga bude i ovaj projekt u obliku diplomskog rada.

Glavni ciljevi ovog rada su robotizacija, digitalizacija i modernizacija procesa završne montaže koji se u ovom trenutku izvodi u potpunosti manualno/ručno. Konkretno, želi se smanjiti potreban broj ljudi, uvesti digitalizirano praćenje odrađenog posla na dnevnoj bazi, integrirati robote koji bi preuzeli teške fizičke operacije i uvesti upravljanje linijom preko PC/tablet aplikacije. Projekt je zbog financijske i tehničke zahtjevnosti podijeljen u dvije faze. Unutar ovog rada detaljno je razrađena prva faza.

Za modeliranje dijelova i izradu nacrta poslužili su softveri za izradu tehničke dokumentacije i 3D modeliranje, AutoCAD i Catia. Za analizu ciklusa i izradu simulacije korišten je softverski paket RoboDK. Programiranje odabranih Fanucovih robota odrađeno je u programskom paketu Roboguide-u. Aplikacija za praćenje i upravljanje linijom napravljena je u Python-u. Unutar linije predviđeno je i OCR očitavanje serijskih brojeva vatrogasnih aparata. Cilj je očitati serijski broj pomoću kamere, prema njemu kreirati barcode naljepnicu i pomoću nje pratiti životni vijek svakog vatrogasnog aparata od izlaska iz tvornice pa do svakog odrađenog godišnjeg servisa.

Nakon odrađene analize na referentnom aparatu P6, predstavljeni su razni potencijalni benefiti kao što su smanjenje ciklusa sa 40 na 30 sekundi, što donosi izradu oko 150 komada više po smjeni. Smanjenje potrebnog broja ljudi sa 7 na 4. Modernizacija testiranje propusnosti tako da se umjesto uranjanja aparata u vodu, koristi testiranje helijem. Digitalizacija praćenja na način da se nakon svakog radnog dana putem e-maila šalje dnevni izvještaj o napravljenom broju komada. Kao programsko sučelje između linije i operatera, predviđena je aplikacija koja omogućava jednostavno upravljanje linijom bez obzira na radnikovo znanje o robotima i programiranju. Sve zajedno donijelo bi i značajniju financijsku uštedu.

Ključne riječi: vatrogasni aparati, robotizacija, digitalizacija, automatizacija

SUMMARY

In this diploma paper, the concept and idea of automatization of fire extinguishers' final assembly process are represented. Pastor is the largest manufacturer of fire extinguishers in this part of Europe with a long operating tradition of more than 90 years. The company has entered a period of investment in the production plant and new machines, therefore this diploma paper is also a part of this process.

The main objectives are robotization, digitization and modernization of the final assembly process, which is currently performed manually. In particular, it wants to reduce the required number of people, introduce digital monitoring of work done on a daily basis, integrate robots that would take on heavy physical operations and present line control via PC / tablet application. Due to financial and technical complexity, the project is divided into two phases. Within this paper, the first phase is elaborated in detail.

Technical documentation and 3D modeling were done in AutoCAD and Catia. The RoboDK software package was used for cycle analysis and process simulation. The programming of selected Fanuc robots was done in the Roboguide software package. The line tracking and control application was created in Python. OCR reading of serial numbers of fire extinguishers is also planned within the line. The goal is to read the serial number with the help of a camera, create a barcode label according to it and use it to monitor the life of each fire extinguisher from leaving the factory to each completed annual service.

After the analysis on the reference type P6, various potential benefits were presented, such as reducing the cycle from 35 to 30 seconds, which brings about 100-150 more pieces per shift. Reducing the required number of people from 7 to 4. Modernization of the permeability test so that instead of testing in water, helium testing is used. Digitization of monitoring in a way that after each working day, a daily report on the number of pieces made is sent by e-mail. As a programming interface between the line and the operator, an application that enables easy line management regardless of the worker's knowledge of robots and programming is introduced. Altogether would also bring significant financial savings.

Keywords: fire extinguishers, robotization, digitization, automation

UVOD

Za diplomski rad je odrađena ideja automatizacije linije za završnu montažu vatrogasnih aparata. Cijeli projekt se radio za tvrtku Pastor TVA u kojoj sam zaposlen kao student.

Pastor TVA je najveći proizvođač vatrogasnih aparata u ovom području Europe i ima dugu tradiciju od 90 godina postojanja. Pogon tvrtke nalazi se u mjestu Rakitje kraj Zagreba, a na godišnjoj razini proizvede se više od 150 000 vatrogasnih aparata raznih vrsta.

Tvrtka je trenutno ušla u razdoblje investicija u proizvodni pogon i nove strojeve. Dogovoreno je da dio tog projekta bude i ovaj diplomski rad. Radi se o automatizaciji procesa završne montaže koji se odvija u takozvanoj „bijeloj sobi“ . Radu se pristupilo istraživanjem tržišta specijalnih strojeva za vatrogasne aparate i pregledom raznih postojećih automatskih procesa montaže aparate u svijetu. S obzirom da je Pastorova proizvodnja vrlo fleksibilna i ima jako puno raznih tipova aparata, komercijalna rješenja za automatizaciju u ovom slučaju nisu primjenjiva. Potrebno je osmisliti fleksibilni proces koji bi se brzo prilagođavao promjeni tipova aparata i parametara linije. Rad se sastojao od ideje, simulacije, konstrukcije i analize poboljšanog procesa. Korišteni su programski paketi Catia, AutoCAD, RoboDK, Roboguide i Python.

.

1. VATROGASNI APARATI

Vatrogasni aparat je uređaj koji se koristi za hitnu intervenciju pri nastanku požara. Zakonom je propisana njihova obavezna prisutnost u raznim građevinama, vozilima, avionima, plovilima itd. Postoji više vrsta vatrogasnih aparata od kojih su tri glavne skupine [1]:

a. Aparati pod stalnim tlakom
b. Aparati sa bočicom
c. CO_2 aparati

Aparati pod stalnim tlakom (P aparati) punjeni su sredstvom za gašenje (prah) i pogonskim plinom dušikom pod stalnim tlakom od 15 bara. Pritiskom ventila, spremnik se izlaže atmosferskom traku, komprimirani dušik „tjera“ prah i dolazi do prskanja sredstva za gašenje. Prilikom završne montaže potrebno ih je testirati na propusnost.

Slika 1. Aparati pod stalnim tlakom [1]

Aparati sa bočicom (S aparati) imaju kao pogonsko sredstvo bočicu sa CO_2 . Aktiviraju se pritiskom na gumb koji probuši bočicu i uzrokuje tlačenje sredstva za gašenje požara, te njegovo izlaženje iz spremnika. Imaju mlaznicu kojom se po potrebi može prekidati istjecanje sredstva. Prilikom završne montaže nije ih potrebno testirati na propusnost.

Slika 2. Aparati sa bočicom [1]

CO_2 aparati su punjeni ugljičnim dioksidom koji se koristi kao sredstvo za gašenje požara. Posebno se preporučaju za korištenje kod požara osjetljivih materijala i uređaja, jer CO_2 ne oštećuje i ne onečišćuje (elektronika, električni uređaji).

Slika 3. CO_2 aparati [1]

Ova linija bit će namijenjena za P i S aparate do 20 kg težine, s obzirom da se CO_2 aparati pune i sklapaju na drugačiji način u posebnom odjelu proizvodnje. Glavni fokus biti će na aparate pod stalnim tlakom (P aparati) s obzirom da se oni moraju testirati na propusnost i najviše ih se proizvede na godišnjoj razini. Cilj je da linija zadovolji potrebe za najsloženije aparate kod završne montaže pa da se onda lagano može prilagoditi za sve ostale.

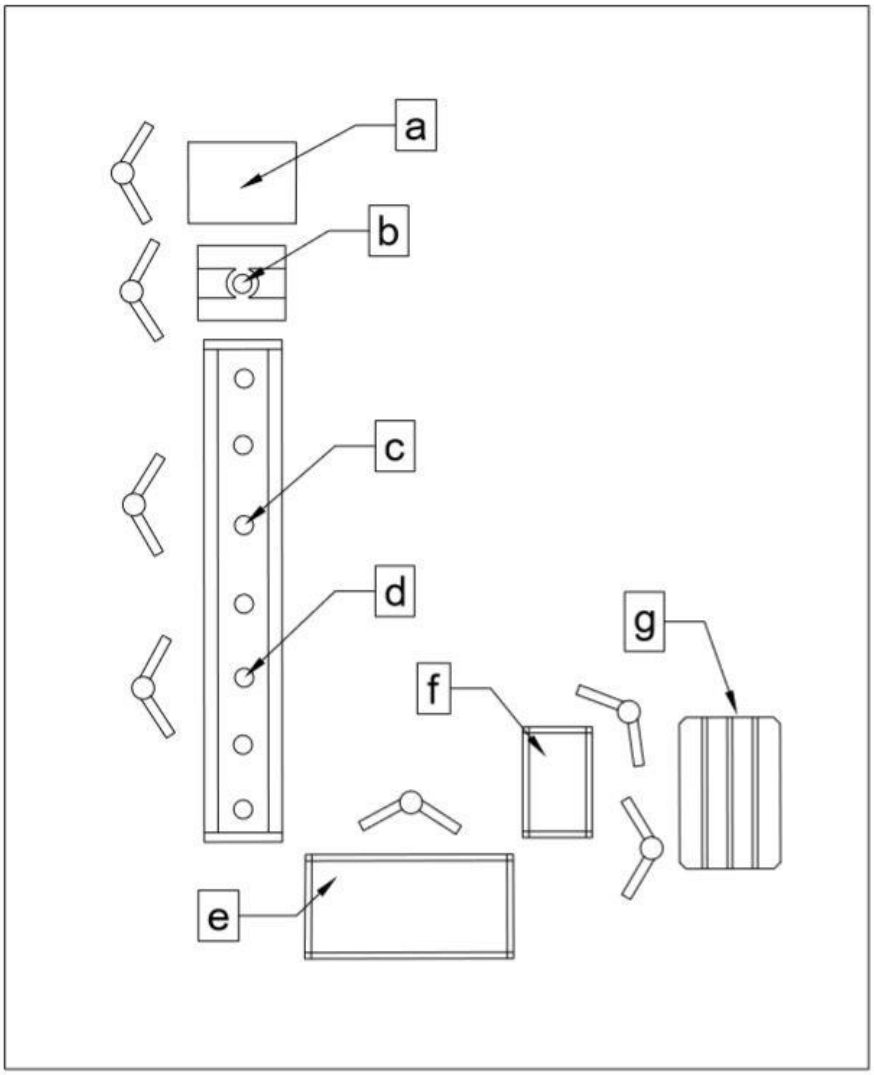
2. POSTOJEĆA LINIJA ZA ZAVRŠNU MONTAŽU

Na trenutnoj liniji za završnu montažu vatrogasnih aparata radi 7 radnika. Za primjer je uzet aparat P6 (najprodavaniji, referentni aparat).

Slika 4. P6 aparat [1]

Proces se sastoji od slijedećih operacija:

a. Punjenje spremnika sredstvom za gašenje (prah)
b. Stezanje ventila na spremnik pneumatskim pištoljem
c. Punjenje aparata pogonskim plinom (dušikom)
d. Stavljanje osigurača i plombe
e. Ispitivanje propusnosti u vodenoj kupki
f. Brisanje aparata i pritezanje mlaznice
g. Stavljanje plastične zaštite i pakiranje aparata u kutije



Slika 5. Tlocrt trenutnog pogona za završnu montažu

Na slici 5. vidi se raspored po radnim mjestima postojećeg pogona za završnu montažu.

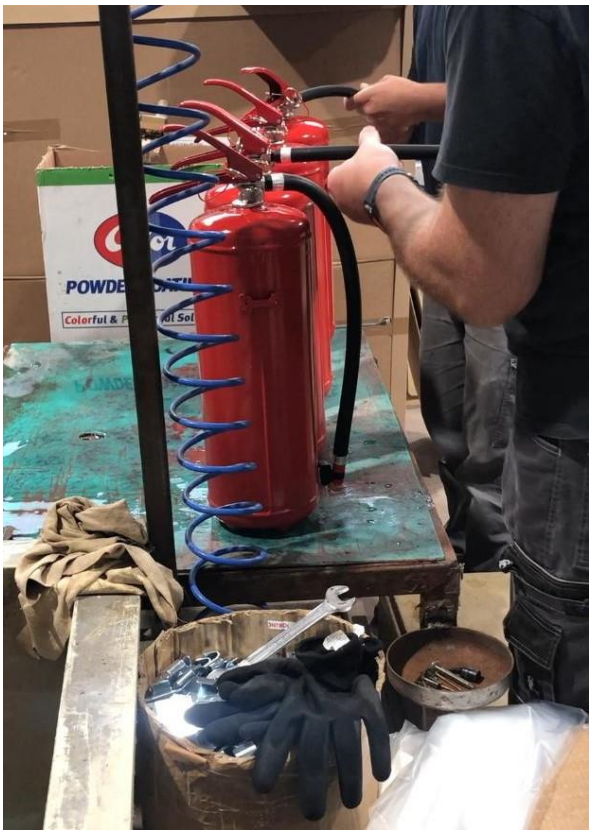
Slika 6. Punjenje aparata prahom

Slika 7. Stezanje ventila pneumatskim pištoljem

Slika 8. Punjenje aparata dušikom

Slika 9. Stavljanje osigurača i plombe

Slika 10. Testiranje aparata u vodi



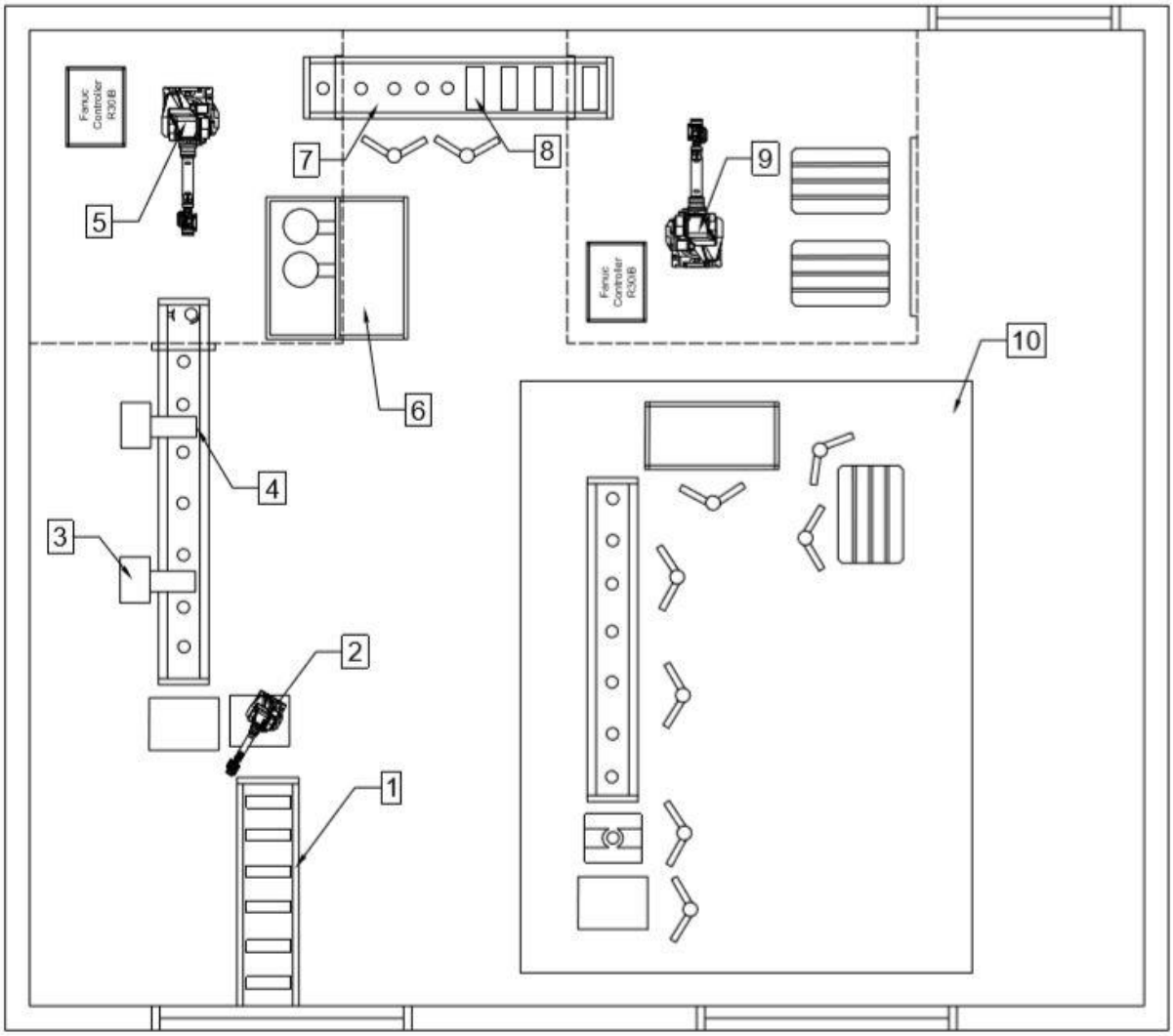
Slika 11. Brisanje aparata i pritezanje mlaznice

Slika 12. Stavljanje plastične zaštite i pakiranje aparata u kutije

Trenutni kapacitet završne montaže za referentni aparat P6 iznosi oko 650-700 komada dnevno. Radi se u jednoj smjeni koja traje 8 sati. Uzimajući u obzir pauze, ispada da je efektivno radno vrijeme po smjeni 7 sati. Težina aparata je oko 9,50 kg.

3. NOVA LINIJA ZA ZAVRŠNU MONTAŽU

Projekt nove linije završne montaže odvijao bi se u dvije faze. Na slici 13. vidljiva je kompletna nova linija nakon što se završe obje faze projekta.



Slika 13. Tlocrt kompletne nove linije i rasporeda procesa završne montaže

Nova linija za završnu montažu zahtijevala bi 2 radnika dok bi sve ostalo bilo automatizirano i robotizirano. Funkcionirala bi na način da bi bila povezana s ostatkom proizvodnje. Nakon što su spremnici kompletno završeni i odrađen je posljednji proces prije odlaska na završnu montažu (tiskanje sita ili lijepljenje naljepnice s certifikatima i nazivom aparata), spremnici bi se stavljali na pokretnu traku koja bi vodila do prostorije gdje se odrađuje završna montaža. Tablica 1. objašnjava pojedine dijelova linije.

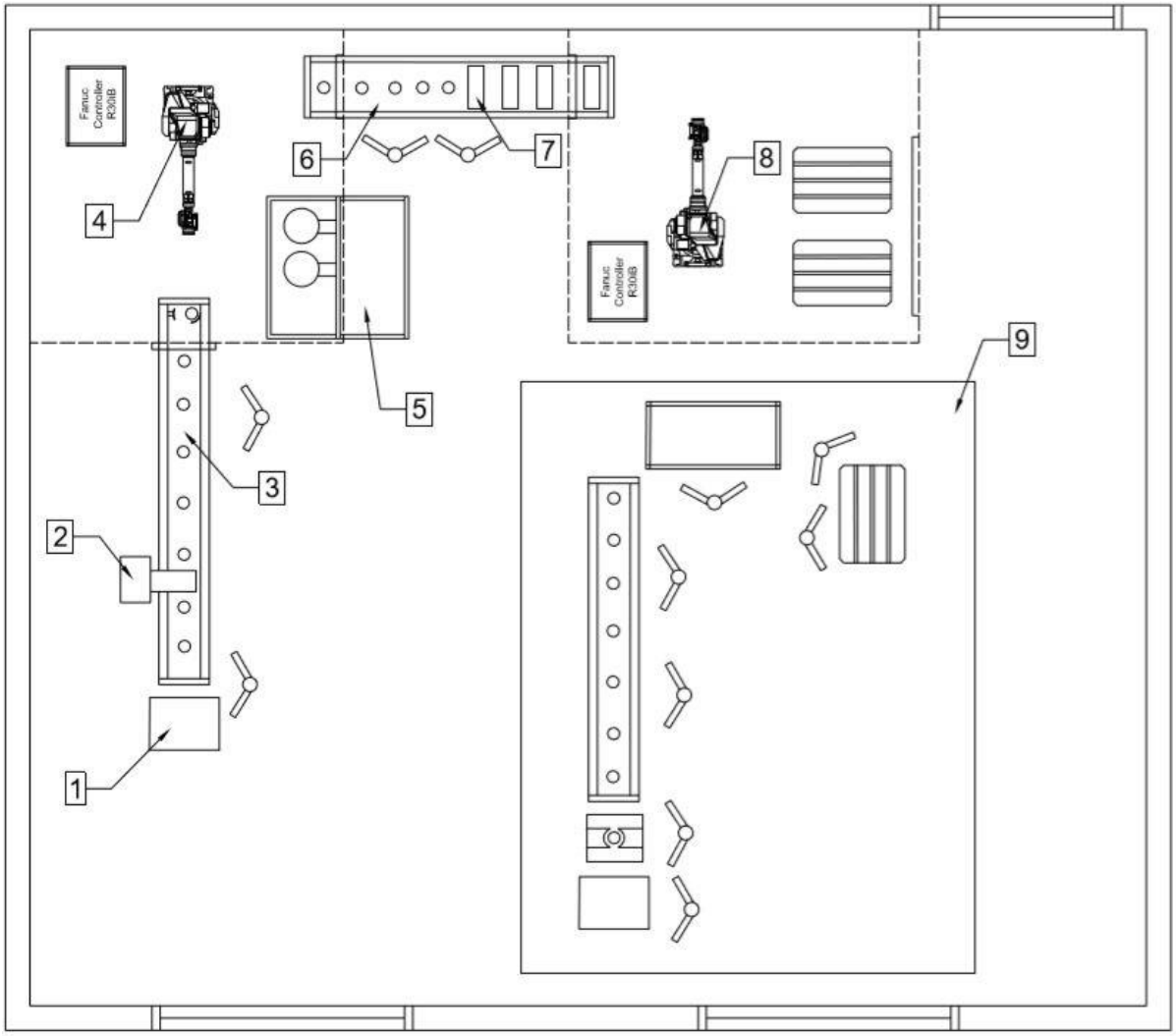
Tablica 1. Elementi automatske linije za završnu montažu

Automatska linija za završnu montažu	
1	Pokretna traka koja bi povezivala završnu montažu s ostatkom proizvodnje. Odmah po otiskivanju podataka ili lijepljenja naljepnice na spremnik oni bi se putem trake 1 slali na završnu montažu.
2	Punilica praha - 1. robotska stanica, kolaborativni robot UR16 (smije raditi u kontaktu s ljudima) opslužuje stroj za punjenje prahom. Nakon završetka punjenja premješta spremnike na drugu pokretnu traku.
3	Stroj za automatsko stezanje ventila, namješta i steže ventil na spremnik.
4	Stroj za automatsko punjenje aparata aktivacijskim plinom (kombinacija dušika i 5% helija).
5	2. robotska stanica - Fanuc M710iC-50 uzima napunjene aparate sa trake i opslužuje stroj za ispitivanje propusnosti pomoću helija, nakon završetka testiranja uzima aparate i stavlja na traku prema 2. robotskoj stanici.
6	Stroj za ispitivanje propusnosti aparata. Funkcionira pomoću helija kojeg ima 5% u aktivacijskoj smjesi, detektira ako postoji bilo kakvo istjecanje helija.
7	Pritezanje mlaznice na aparat, plombiranje i stavljanje plastične zaštite - 1. radno mjesto
8	Sklapanje kutija i stavljanje aparata u kutiju - 2. radno mjesto.
9	3. robotska stanica - Fanuc M710iC-50 paletizacija kutija.
10	Trenutna linija za završnu montažu, pomaknuta u drugi dio sobe da se može koristiti kod posebnih aparata, malih narudžba i dok se nova linija ne uhoda.

Ipak, s obzirom na financijsku i tehničku zahtjevnost kompletnog projekta, fokus ovog diplomskog rada će biti na 1. fazi.

Kod 1. faze nema trake broj 1 koja povezuje završnu montažu sa ostatkom proizvodnje, nema kolaborativnog robota koji opslužuje stroj za punjenje prahom i nema automatskog stroja za punjenje dušikom.

Novi pogon nakon 1. faze vidljiv je na slici 14.



Slika 14. Tlocrt nove linije i rasporeda procesa završne montaže (1.faza)

Tlocrt je rađen u mjerilu prema izmjerenim dimenzijama prostorije u kojoj se vrši završna montaža. Nova linija vidljiva je s lijeve strane i ona bi bila na mjestu gdje je trenutno stara linija. Pod brojem 9 vidljiva je stara linija koja bi se i dalje upotrebljavala dok se nova linija potpuno ne uhoda, te bi se koristila za posebne aparate velikih dimenzija (50-100 kg) koji se rijetko proizvode.

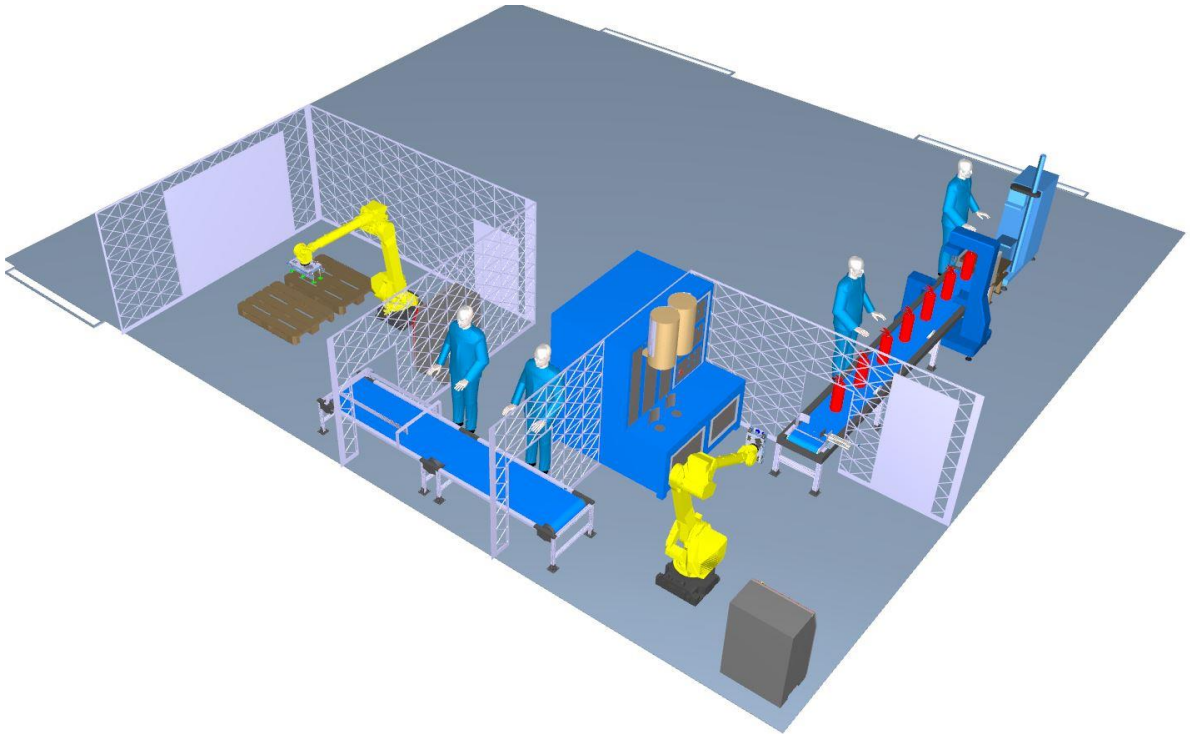
Pojedine stanice nove linije navedene su u tablici 2.

Tablica 2. Elementi automatske linije za završnu montažu (1. faza)

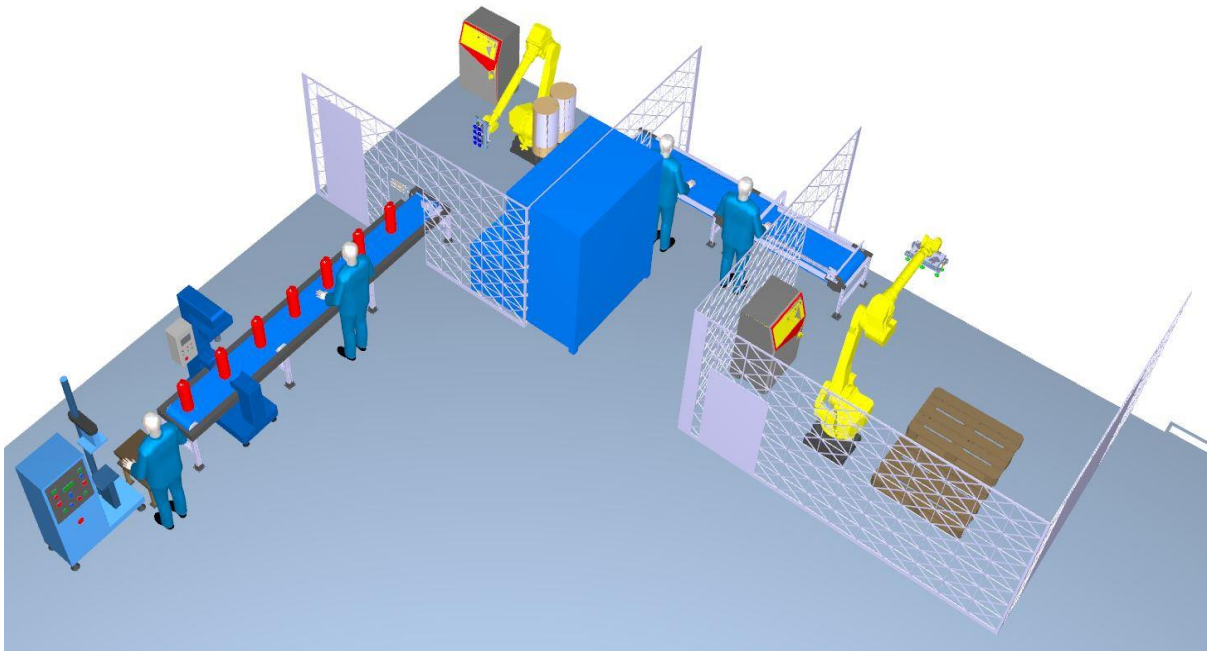
Automatska linija za završnu montažu	
1	Punilica praha - 1. radno mjesto, radnik puni spremnike prahom za gašenje požara, stavlja premaz i prvi navoj ventila.
2	Stroj za automatsko stezanje ventila. Propisanim momentom priteže ventil na spremnik .
3	Punjenje aparata aktivacijskim plinom (kombinacija helija i dušika) i plombiranje - 2. radno mjesto.
4	1. robotska stanica - Fanuc M710iC-50 uzima napunjene aparate sa trake i opslužuje stroj za ispitivanje propusnosti pomoću helija, nakon završetka testiranja uzima aparate i stavlja na traku prema 2. robotskoj stanici.
5	Stroj za ispitivanje propusnosti aparata. Funkcionira pomoću helija kojeg ima 5% u aktivacijskoj smjesi, detektira ako postoji bilo kakvo istjecanje helija.
6	Pritezanje mlaznice na aparat i stavljanje plastične zaštite - 3. radno mjesto.
7	Sklapanje kutija i stavljanje aparata u kutiju - 4. radno mjesto.
8	2. robotska stanica - Fanuc M710iC-50 paletizacija kutija.
9	Trenutna linija za završnu montažu, pomaknuta u drugi dio sobe da se može koristiti kod posebnih aparata, malih narudžba i dok se nova linija ne uhoda.

Na slici 15. i 16. može se vidjeti 3D prikaz u programskom paketu RoboDK u kojem je rađena simulacija i analiza ciklusa nove linije o čemu će biti više detalja u nastavku.

Također, pojedini novi elementi svake stanice biti će detaljno razrađeni u nastavku.



Slika 15. 3D prikaz nove linije u RoboDK



Slika 16. 3D prikaz nove linije u RoboDK (drugi pogled)

4. NOVI ELEMENTI PO STANICAMA

U poglavlju 4. detaljno su razrađeni i objašnjeni svi elementi koji su uvedeni u novu liniju.

4.1 Stroj za automatsko stezanje ventila

Prva promjena u novoj liniji je stroj za automatsko stezanje ventila. Umjesto dosadašnjeg načina, gdje je radnik ručno pomoću pneumatskog pištolja stezao ventile, sada bi to radio stroj. Radnik bi nakon punjenja aparata morao samo staviti prvi navoj ventila, te ga približno pravilno usmjeriti.

Kada aparat dođe na stanicu za stezanje ventila, nakon signala senzora da je aparat detektiran, pneumatski cilindar namješta i spušta glavu za stezanje ventila. Bočni držači stežu aparat.

Zatezanje se odrađuje na unaprijed propisani i postavljeni moment koji ovisi o aparatu (za P6 moment iznosi 50 Nm), nakon ostvarivanja momenta, dodatno se mora zadovoljiti i pozicija unutar tolerancije momenta (položaj ručke ventila od 90 stupnjeva u odnosu na uzdužni zavar).

Slika 17. Stroj za automatsko stezanje ventila

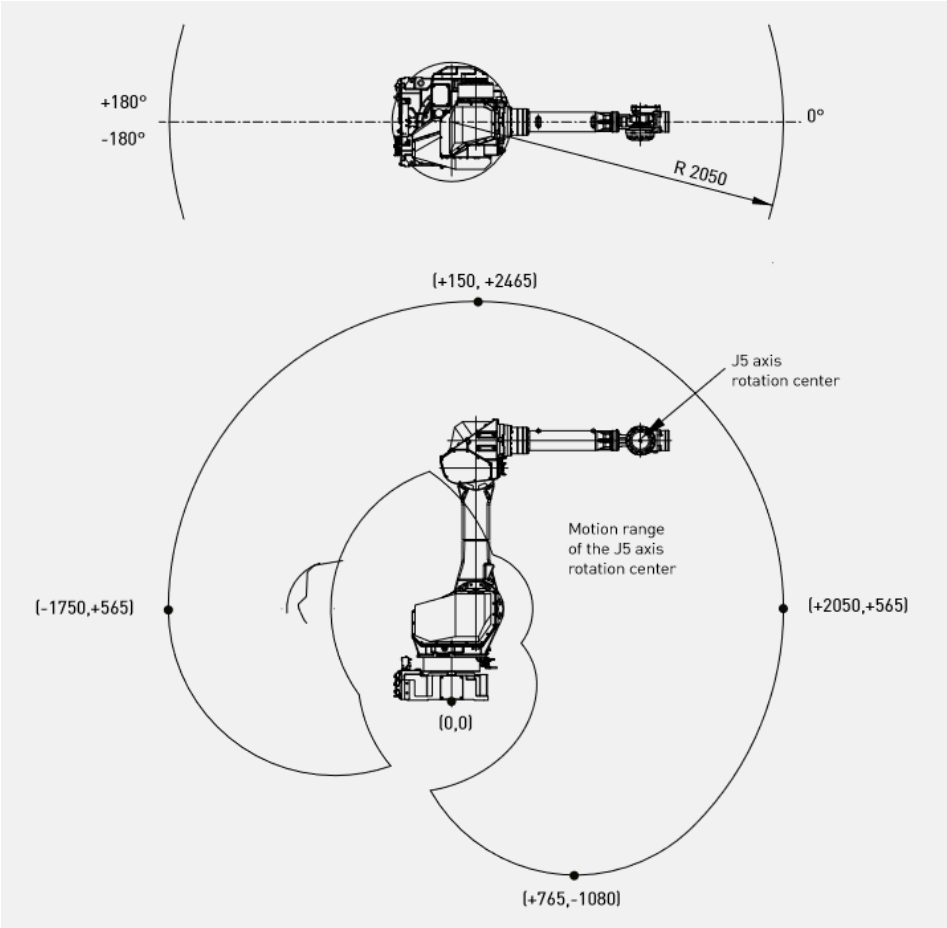
4.2 Prva robotska stanica

Na prvoj robotskog stanici kompletno je zamijenjen dosadašnji način testiranja propusnosti aparata. Umjesto stavljanja aparata u vodenu u kupku i promatranja da li dolazi do istjecanja (pojave mjehurića) aparati će se testirati na istjecanje helija. Takav način testiranja povlači za sobom nekoliko promjena u odnosu na trenutnu liniju.

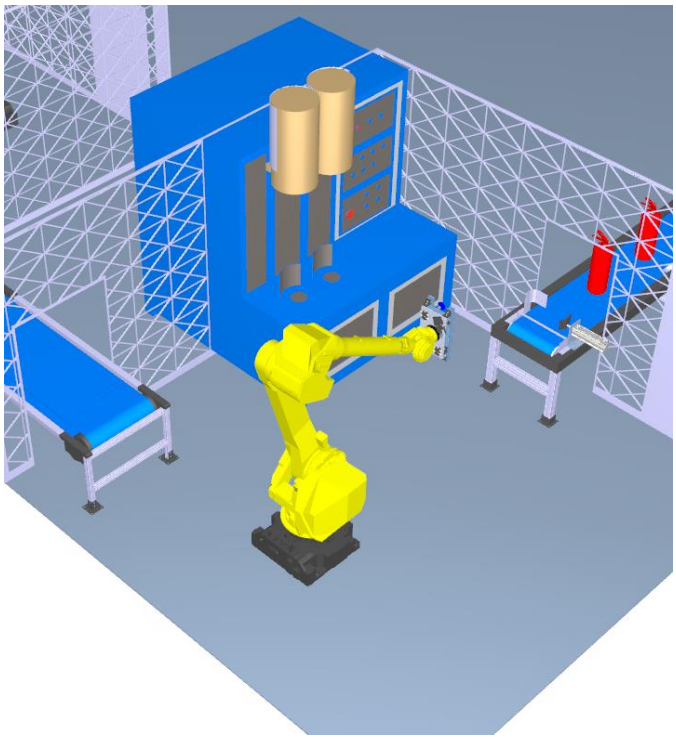
a. Nabava stroja za ispitivanje istjecanja helija
b. Promjena aktivacijskog plina (mješavina dušika i helija - 5% volumnog udjela He)

Zadatak robota je opsluživanje komora u kojima će se odvijati testiranje, te nakon završetka, izuzimanje aparata i stavljanje na liniju prema drugoj robotskoj stanici.
Odabran je Fanucov 6-osni robot M710iC – 70 slijedećih karakteristika:

- Nosivost – 70 kg
- Domet – 2050 mm
- Težina – 560 kg



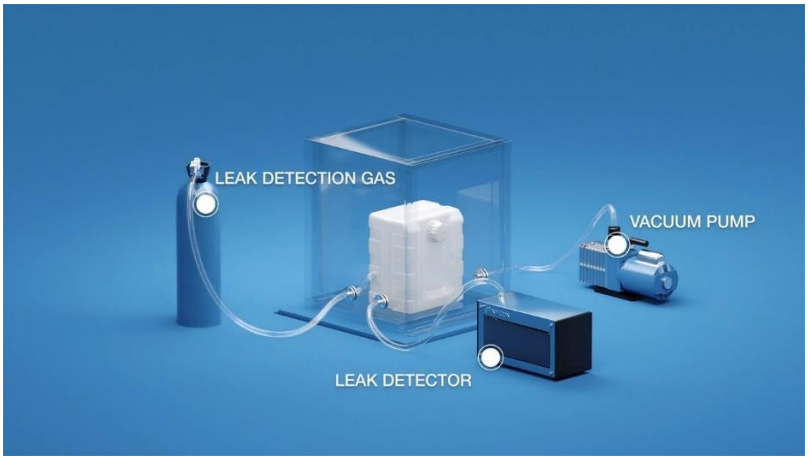
Slika 18. Radni prostor robota Fanuc M710iC-70 [3]



Slika 19. Prva robotska stanica

4.3 Ispitivanje propusnosti aparata helijem

Helij je inertni plin koji ima jednu od najmanjih molekula od svih plinova, siguran je za upotrebu i ne reagira sa materijalima koji se podlažu testiranju [4]. U većini slučajeva za njegovu detekciju koristi se maseni spektrometar. S obzirom na njegova svojstva, ali i cijenu, često se koristi u kombinaciji sa drugim plinovima. U ovom slučaju za testiranje će se koristiti smjesa dušika i 5% helija. U okolini mjesta gdje dolazi do punjenja spremnika, zrak se vrlo brzo zasiti helijem tako da je prilikom testiranja potrebno osigurati čistu okolinu.



Slika 20. Postupak testiranja istjecanja helija [5]

Na slici 20. vidljivi su svi dijelovi i način testiranja koji će se koristiti [5]. Aparat će se nakon punjenja smjesom dušika i helija stavljati u komoru u kojoj će nastupiti testiranje.
Stroj za ispitivanje će se sastojati od vakuumske pumpe, detektora i komore.
Nakon što se aparat zatvori u komoru, pumpa će stvoriti vakuum i detektor će očitavati da li dolazi do istjecanja helija iz spremnika u komoru.

Odabran je stroj od tvrtke Fritz – EMDE koji se sastoji od dvije komore za ispitivanje, dvije vakuumske pumpe koje se koriste za evakuaciju zraka i helij detektora [6].
Potreban je udio od 5% helija u smjesi za ispitivanje, vrijeme ciklusa ispitivanja aparata iznosi oko 25 s. Stroj je vidljiv na slici 21.

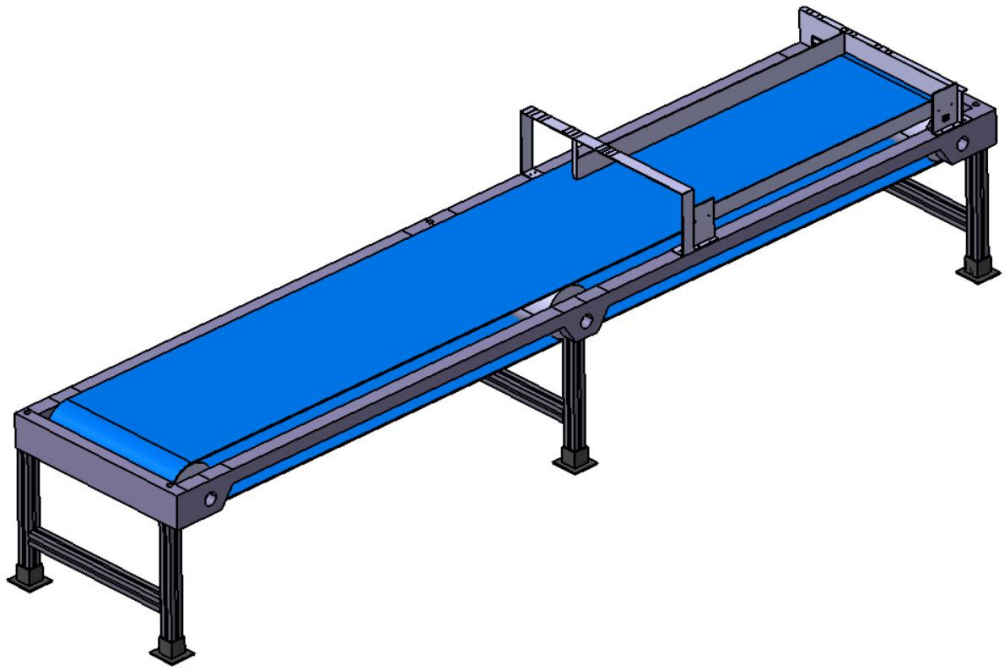
Slika 21. Stroj za ispitivanje istjecanja helija [6]

Za rad stroja, koristi se pneumatski pogon za koji je potreban pročišćeni zrak tlaka 5 bara.
Stroj se koristi za ispitivanje aparata od 1-12 kg punjenja (P1 – P12).

4.4 Traka podesiva prema paletizaciji kutija

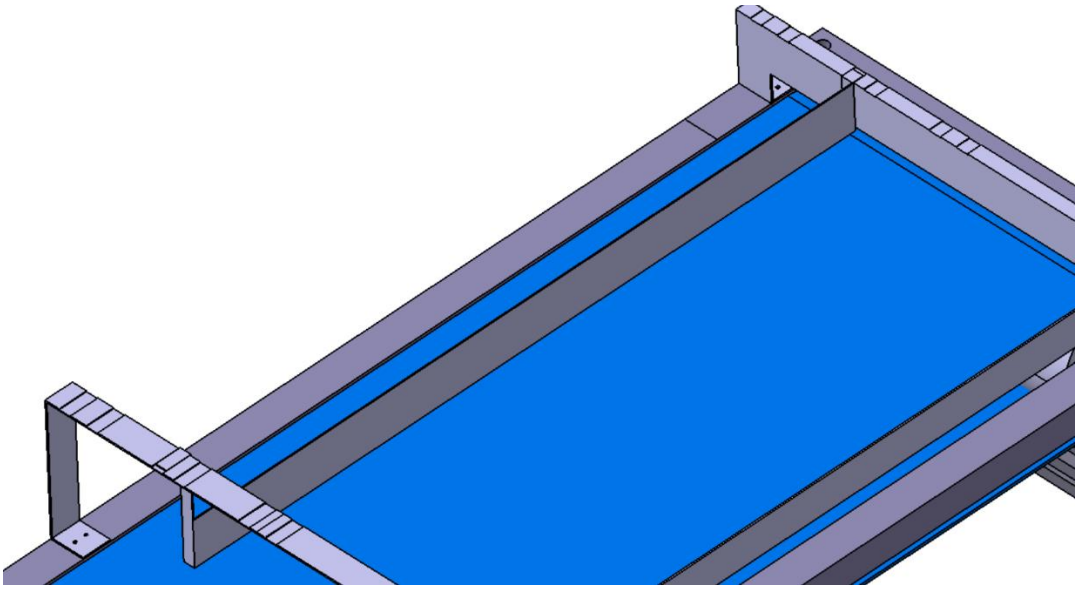
Nakon završetka ispitivanja propusnosti, robot uzima aparat i stavlja ga na traku prema paletizaciji kutija.

Dva radnika zadužena su za pritezanje mlaznice (ako je potrebno, ovisi o aparatu), stavljanja plastične navlake na aparat, kompletiranja kutije i polaganja aparata u kutiju.
Traka je zamišljena da pomoću ručno podesivih limova. Laganim pomakom gornjeg lima koji djeluje kao graničnik omogućeno je jednostavno i brzo podešavanje trake ovisno o tipu aparata to jest veličini kutije o kojoj se radi. Na zaustavnom mjestu, po detekciji senzora, aktivira se mali pneumatski cilindar koji dodatno stisne kutiju i fiksira je na točnu poziciju izuzimanja.
Za detekciju kutija koristi se induktivni senzor.



Slika 22. Model transportne trake prema paletizaciji kutija

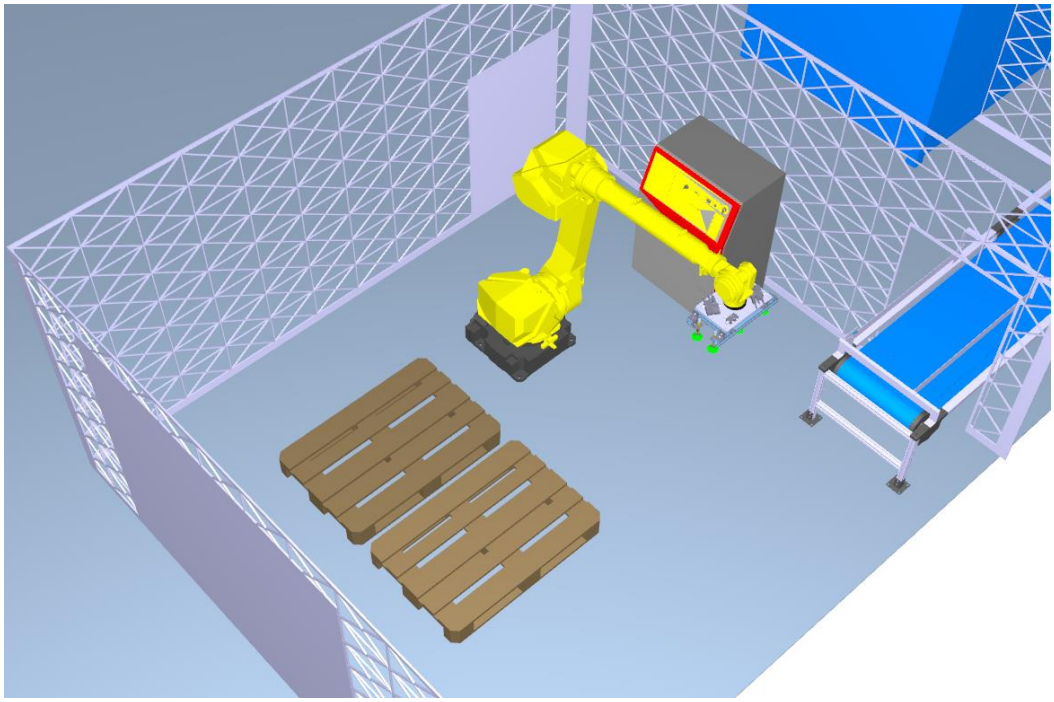
Na slici 23. vidi se način na koji se limeni graničnik pomiče u utore na nosačima ovisno o veličini kutije.



Slika 23. Utori za pomak graničnika ovisno o veličini kutije

4.5 Druga robotska stanica

Na drugoj robotskoj stanici vrši se paletizacija kutija na dvije palete. Odabran je isti robot kao i na prvoj robotskoj stanici – Fanuc M710iC – 70. Nakon što se napune obje palete (za slučaj aparata P6 – 56 do 60 komada po paleti) otvara se zaštitna ograda i palete se izuzimaju te prenose dalje u skladište. Druga robotska stanica vidljiva je na slici 24.

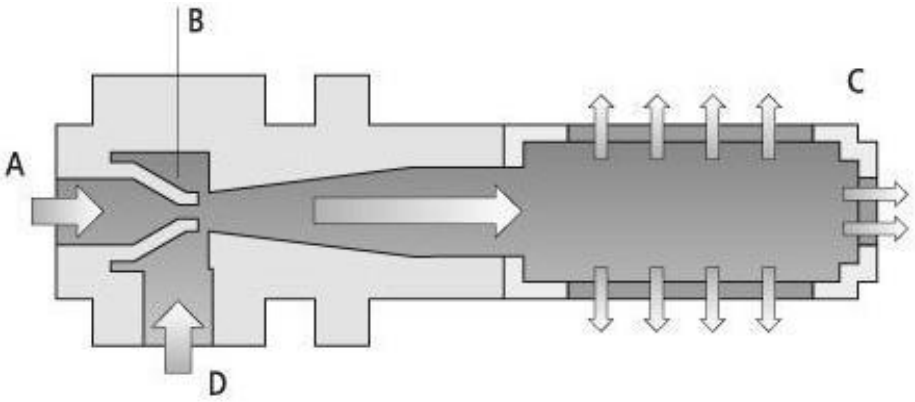


Slika 24. Druga robotska stanica

5. ROBOTSKE HVATALJKE

Za oba robota izrađene su konceptualne pneumatske hvataljke s vakuumskim kapicama. Ideja je da obje hvataljke budu prilagodljive ovisno o dimenzijama predmeta tako da prilikom promjene vrste aparata ne bude potrebe za nikakvom hardverskom promjenom na hvataljkama nego samo softverskom.

Kod obje hvataljke korišteni su vakuum generatori koji funkcioniraju na Venturijevom principu.



Slika 25. Venturijev vakuum generator [9]

Na ulaz A ulazi komprimirani zrak. U točki B dolazi do suženja cijevi i povećanja brzine zraka. Nakon što zrak izađe iz suženog dijela nastaje vakuum (tlak zraka niži od atmosferskog) i dolazi do „usisavanja“ zraka u ulaz D.

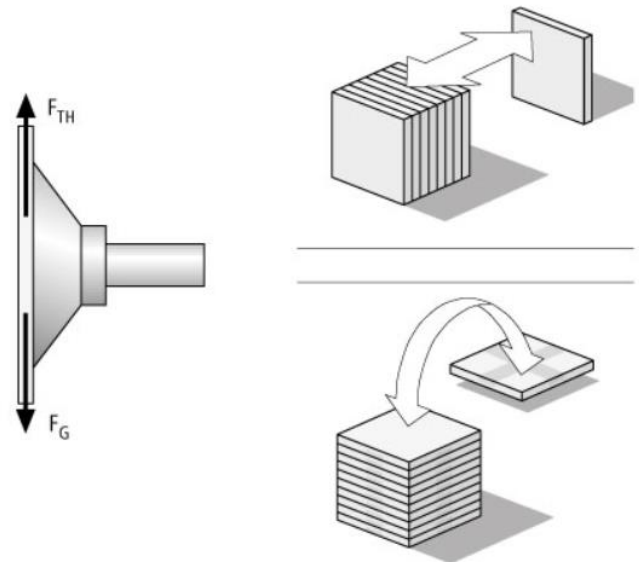
Komprimirani i usisani zrak zajedno izlaze preko „silencera“ C.

5.1 Hvataljka za spremnike

Proračun hvataljke za spremnike rađen je prema [7] i [8] , a komponente su odabrane od poznate kompanije za pneumatsku opremu – Schmalz [9].

5.1.1 Sila držanja

Za proračun je uzet slučaj u kojem imamo: vertikalno podizanje predmeta tako da usisna sila djeluje vertikalno, horizontalni prijenos do stroja za testiranje propusnosti helijem, izuzimanje predmeta sa stroja i spuštanja na paletu. U obzir moramo uzeti horizontalne i vertikalne sile. Vakuumski tlak je -0,6 bar.



Slika 26. Sile u slučaju vertikalnog podizanja i prijenosa predmeta [7]

Proračun se radi na najekstremnijem slučaju kada je težina aparata 18,8 kg (S12 aparat) , faktor sigurnosti 2 i koeficijent trenja 0,5 za metal prema [7] .

Parametri :

$$m = 18{,}8\ kg$$

$$S = 2$$

$$a = 5\ \frac{m}{s^2}$$

$$g = 9{,}81\ \frac{m}{s^2}$$

$$\mu = 0{,}5 \tag{1.1}$$

Sila držanja („holding force“):

$$F_H = \frac{m}{\mu} \cdot (g + a) \cdot S \tag{1.2}$$

$$F_H = 1113{,}71\ N \tag{1.3}$$

Sila usisa („suction force“):

$$F_S = \frac{F_H}{n} \tag{1.4}$$

Gdje n predstavlja broj kapica (u ovom slučaju 8):

$$F_S = 139{,}21\ N \tag{1.5}$$

5.1.2 Vakuumske kapice

Sa stranice kompanije Schmalz [9] su odabrane SAX 60 ED-85 G1/4 IG vakumske kapice promjera 60 mm sa sljedećim karakteristikama:

$$\begin{gathered} d_c = 60\ mm \\ F_{s\_cap} = 154\ N > F_s \\ V = 25{,}2\ cm^3 \\ d_{crijevo} = 6\ mm \\ m = 0{,}0428\ kg \end{gathered} \tag{1.6}$$

Kontrolni proračun prema [8]:

$$d_{pot} = 2 \cdot \sqrt{\frac{(g+a)}{\mu \cdot \pi} \cdot \frac{m \cdot S}{\Delta p \cdot n}} \tag{1.7}$$

$$d_{pot} = 54\ mm < d_c \tag{1.8}$$

Prema (1.6) i (1.8) vidljivo je da odabrane vakuumske kapice zadovoljavaju.

5.1.3 Crijevo za zrak

Iz dokumentacije vakuumskih kapica [9] vidljivo je da unutarnji promjer crijeva za zrak mora biti 6 mm pa je odabrano crijevo VSL 8-6 PE unutarnjeg promjera d = 6 mm i vanjskoga D = 8 mm.

5.1.4 Spojni elementi

Odabrani su podesivi nosači HTR UNI 2N G2 80 koji imaju oslonac koji je prilagodljiv obliku predmeta što je u ovom slučaju potrebno s obzirom da su aparati cilindričnog oblika. Karakteristike su prema [9]:

$$m = 0{,}170\ kg$$

Konektor za kapice = G1/4 M

Konektor za vakuum = G1/4 F

(1.9)

5.1.5 Distributeri za vakuum

Odabrana su dva VTR G1/4 IG 5x G1/8 distributera sa 5 izlaza. Oba distributera će biti spojena na po 4 kapice. Veće kutije hvatat će se sa svih 8 kapica, dok će se manje hvatati sa 4.

5.1.6 Vakuum generator

Prema tablici iz [9]:

Suction capacity as a function of suction cap diameter

Suction-cap Ø	Suction capacity V_S	
Up to 60 mm	0.5 m^3/h	8.3 l/min
Up to 120 mm	1.0 m^3/h	16.6 l/min
Up to 215 mm	2.0 m^3/h	33.3 l/min
Up to 450 mm	4.0 m^3/h	66.6 l/min

Slika 27. Tablica kapaciteta kapice u odnosu na promjer [9]

S obzirom da odabrane kapice imaju promjer 60 mm odabire se malo veća vrijednost od one koja je dana u tablici na slici 27.

$$V_s = 10\ \frac{l}{min} \tag{1.10}$$

Odlučeno je da će se koristiti dva kompaktna vakuum generatora sa integriranim funkcijama. Svaki će biti spojen na po 4 kapice. U slučaju većih aparata koristit će se svih 8 kapica pa će oba vakuum generatora biti aktuirana, u slučaju manjih aparata aktuirat će se samo jedan generator.

Ukupni protok zraka koji svaki generator mora zadovoljiti za 4 kapice:

$$V_{uk} = n \cdot V_s = 40 \ \frac{l}{min} \tag{1.11}$$

Odabran je vakuum generator SCP 15 NO AS sa sljedećim karakteristikama [2]:

$$m = 0{,}275 \ kg$$

$$d_{mlaznice} = 1{,}5 \ mm$$

$$V_{max} = 65 \ \frac{l}{min} > V_{uk}$$

$$Air_{consump.} = 117 \ \frac{l}{min}$$

$$p = 5 - 6 \ bar$$

$$d_{crijeva,tlak} = 4 \ mm$$

$$d_{crijeva,vakuum} = 6 \ mm \tag{1.12}$$

Iz karakteristika vidimo da odabrani vakuumski injektor (generator) sa integriranim funkcijama (razvodnik, nepovratni ventil, ispušni ventil) zadovoljava tražene karakteristike.

Odabrano je i crijevo prema podacima (1.12):

VSL 6-4 PE:

$$d_{c1} = 4 \ mm$$

$$D_{c2} = 6 \ mm \tag{1.13}$$

5.1.7 Konektori

Na kraju su odabrani potrebni konektori, pločice za spajanje i aluminijski ekstrudirani profili vidljivi na slici 28. i 29.

5.1.8 Popis svih dijelova

Tablica 3. Dijelovi hvataljke za spremnike [9]

	Ime dijela	Kom/m	Cijena (kom, m)	Masa (kom,m)
1.	SAX 60 ED – 85 G1/4 IG	8	€26	0,0428 kg
2.	Crijevo VSL 8-6 PE	10 m	€1.04	0,203 kg
3.	HTR UNI 2N G2 80	8	€91,91	0,170 kg
4.	STV-GE G1/8-AG 8	10	€4.03	0,0163 kg
5.	VTR G1/4-IG 5xG1/8	2	€21,06	0,090 kg
6.	VRS-SB G1/8-AG ISKT O-Ring	2	€1.69	0,006 kg
7.	STV-W G1/4-AG 8	10	€9.88	0,049 kg
8.	Crijevo VSL 6-4 PE	3 m	€0.78	0,145 kg
9.	SCP 15 NO AS	2	€296.40	0,275 kg
10.	STV-W G1/8-AG 6	2	€7.93	0,037 kg
11.	STV-W G1/8-AG 8	1	€9.23	0,043 kg
12.	BEF-WIN 40x40x40 5 MO-PROF	8	€10.92	
13.	NUT-STEI 10x20 M6-IG	40	€3.38	0,005 kg
14.	MO-PROF 40 3TN AL-1 440 mm	2	€53.30	1,8 kg
15.	MO-PROF 40 3TN AL-1 215 mm	2	€53.30	1,8 kg
16.	Aluminijska ploča	1		1,394 kg

Ukupna masa hvataljke bez vijaka, cijevi i pločica iznosi oko 7,16 kg.

Ukupna cijena bez vijaka i troškova izrade aluminijske ploče iznosi €2,629.69 prema [9] .

Nacrt aluminijske ploče nalazi se u prilogu.

5.1.9 Provjera hvatanja najtežeg aparata P3 sa 4 vakuumske kapice

Masa aparata P3:

$$m_{P3} = 5{,}4 \ \ kg \tag{1.17}$$

Sila držanja („holding force“):

$$F_H = \frac{m_{P3}}{\mu} \cdot (g + a) \cdot S$$

$$F_H = 319{,}90 \ N \tag{1.18}$$

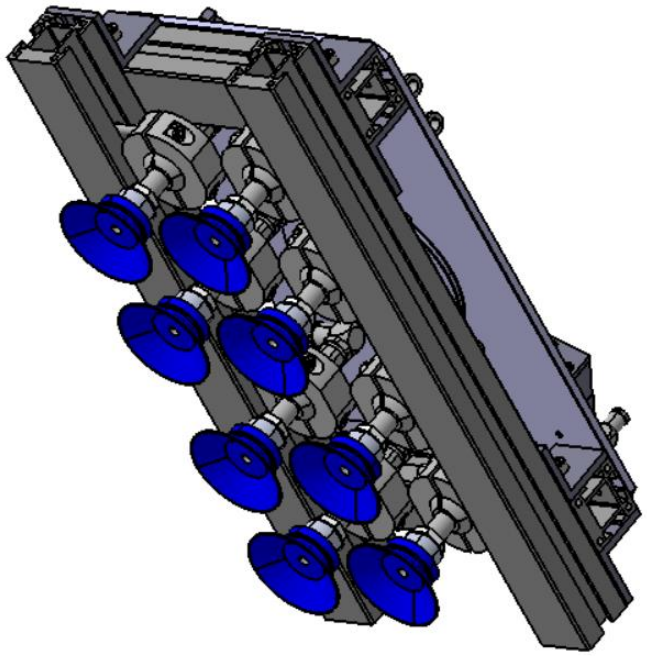
Sila usisa („suction force“):

$$F_S = \frac{F_H}{n} \tag{1.19}$$

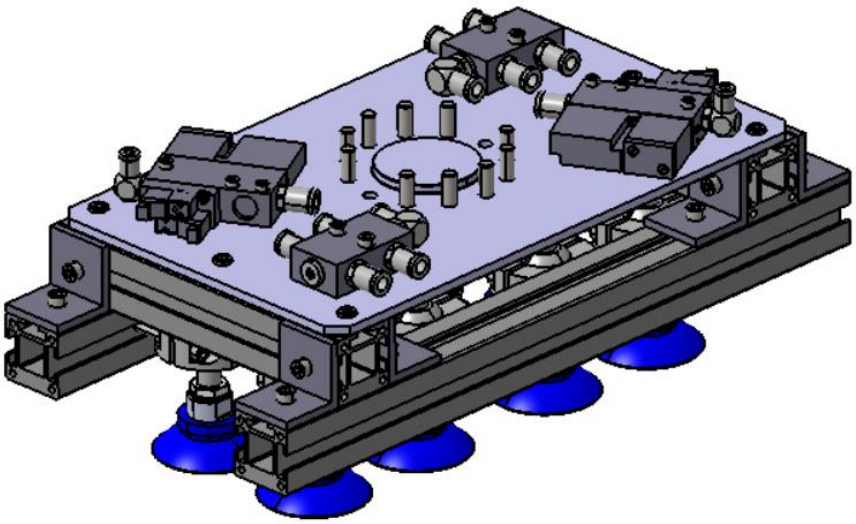
Gdje n predstavlja broj kapica (u ovom slučaju 4):

$$F_S = 79{,}974 \ N < \ F_{S,cap} \tag{1.20}$$

Dakle odabrane kapice zadovoljavaju i u ovom slučaju.



Slika 28. Konceptualna vakuumska hvataljka za spremnike

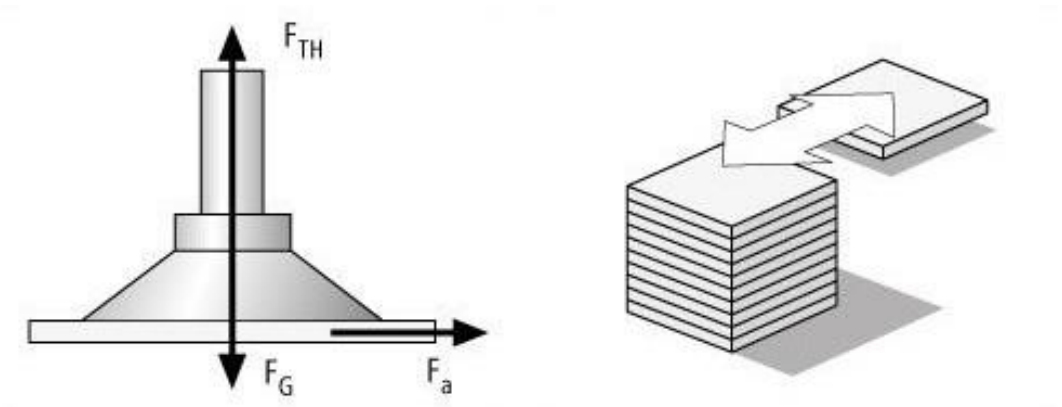


Slika 29. Konceptualna vakuumska hvataljka za spremnike (drugi pogled)

5.2 Hvataljka za kutije

5.2.1 Sila držanja

Za proračun je uzet slučaj u kojem imamo: vertikalno podizanje predmeta, horizontalni prijenos do palete i spuštanje predmeta. U obzir moramo uzeti horizontalne i vertikalne sile. Vakuumski tlak je -0,6 bar.



Slika 30. Sile u slučaju horizontalnog i vertikalnog prijenosa predmeta [7]

Proračun se radi na najekstremnijem slučaju kada je težina kutije 19 kg, faktor sigurnosti 2 i koeficijent trenja 0,6 za karton prema [7] .

Parametri :

$$m = 19\ kg$$

$$S = 2$$

$$a = 5\ \frac{m}{s^2}$$

$$g = 9{,}81\ \frac{m}{s^2}$$

$$\mu = 0{,}6$$

(1.21)

Sila držanja („holding force“):

$$F_H = m \cdot \left(g + \frac{a}{\mu}\right) \cdot S \tag{1.22}$$

$$F_H = 689{,}45\ N \tag{1.23}$$

Sila usisa („suction force“):

$$F_S = \frac{F_H}{n} \tag{1.24}$$

Gdje n predstavlja broj kapica (u ovom slučaju 6):

$$F_S = 114{,}91\ N \tag{1.25}$$

5.2.2 Vakuumske kapice

Sa stranice kompanije Schmalz [9] su odabrane PFYN 60 VU1-72 G1/4IG vakumske kapice promjera 60 mm sa sljedećim karakteristikama:

$$d_c = 60\ mm$$

$$F_{s\_cap} = 130\ N > F_s$$

$$V = 15\ cm^3$$

$$d_{crijevo} = 6\ mm$$

$$m = 0{,}024\ kg$$

(1.26)

Kontrolni proračun prema [8]:

$$d_{pot} = 2 \cdot \sqrt{\frac{\left(g+\frac{a}{\mu}\right)}{\pi} \cdot \frac{m \cdot S}{\Delta p \cdot n}} \tag{1.27}$$

$$d_{pot} = 48\ mm < \ d_c \tag{1.28}$$

Prema (1.26) i (1.28) vidljivo je da odabrane vakuumske kapice zadovoljavaju.

5.2.3 Crijevo za zrak

Iz dokumentacije vakuumskih kapica [9] vidljivo je da unutarnji promjer crijeva za zrak mora biti 6 mm pa je odabrano crijevo VSL 8-6 PE unutarnjeg promjera d = 6 mm i vanjskoga D = 8 mm.

5.2.4 Spojni elementi

Odabrani su nosači sa oprugama FSTE G1/4 – AG 25 VG sljedećih karakteristika prema [9]:

$$m = 0{,}144\ kg$$

Konektor za kapice = G1/4 M

Konektor za vakuum = G1/8 F

$$z = 25\ mm\ (hod\ nosača)$$

(1.29)

5.2.5 Distributeri za vakuum

Odabrana su dva VTR G1/4 IG 5x G1/8 distributera sa 5 izlaza. Jedan od njih će biti spojen na četiri kapice, a drugi na dvije. Veće kutije hvatat će se sa svih šest kapica, dok će se manje hvatati samo sa dvije.

5.2.6 Vakuum generator

Prema tablici iz [9]:

Suction capacity as a function of suction cap diameter

Suction-cap Ø	Suction capacity V_S	
Up to 60 mm	0.5 m^3/h	8.3 l/min
Up to 120 mm	1.0 m^3/h	16.6 l/min
Up to 215 mm	2.0 m^3/h	33.3 l/min
Up to 450 mm	4.0 m^3/h	66.6 l/min

Slika 31. Tablica kapaciteta kapice u odnosu na promjer (kutije)

S obzirom da odabrane kapice imaju promjer 60 mm odabrat ćemo malo veću vrijednost od one koja je dana u tablici na slici 31.

$$V_s = 10\ \frac{l}{min} \tag{1.30}$$

Odlučeno je da će se koristiti dva kompaktna vakuum generatora sa integriranim funkcijama, jedan će kontrolirati 4 kapice, a drugi 2. U slučaju velikih kutija koristit će se svih 6 kapica pa će oba vakuum generatora biti aktuirana, u slučaju manjih kutija aktuirat će se samo manji generator.

Ukupni protok zraka koji veći generator mora zadovoljiti za 4 kapice:

$$V_{uk1} = n_1 \cdot V_s = 40\ \frac{l}{min} \tag{1.31}$$

Odabran je vakuum generator SCP 15 NO AS sa sljedećim karakteristikama [9]:

$$m = 0{,}275\ \ kg$$

$$d_{mlaznice} = 1{,}5\ \ mm$$

$$V_{max} = 65\ \frac{l}{min} > V_{uk1}$$

$$Air_{consump.} = 117\ \frac{l}{min}$$

$$p = 5 - 6\ \ bar$$

$$d_{crijeva,tlak} = 4\ \ mm$$

$$d_{crijeva,vakuum} = 6\ \ mm \tag{1.32}$$

Iz karakteristika vidimo da odabrani vakuumski injektor (generator) sa integriranim funkcijama (razvodnik, nepovratni ventil, ispušni ventil) zadovoljava tražene karakteristike.

Ukupni protok zraka koji manji vakuum generator mora zadovoljiti:

$$V_{uk2} = n_2 \cdot V_s = 20\ \frac{l}{min} \tag{1.33}$$

Odabran je vakuum generator SCP 10 NO AS sa sljedećim karakteristikama [9]:

$$m = 0{,}275\ kg$$

$$d_{mlaznice} = 1{,}0\ mm$$

$$V_{max} = 37\ \frac{l}{min} > V_{uk2}$$

$$Air_{consump.} = 53\ \frac{l}{min}$$

$$p = 5 - 6\ bar$$

$$d_{crijeva,tlak} = 2\ mm$$

$$d_{crijeva,vakuum} = 4\ mm \tag{1.34}$$

Iz (1.34) vidljivo je da i manji vakuumski generator zadovoljava postavljene zahtjeve.

Odabrana su i crijeva prema podacima (1.32) i (1.34):

VSL 6-4 PE:

$$d_{c1} = 4\ mm$$
$$D_{c2} = 6\ mm \tag{1.35}$$

VSL 4-2 PE:

$$d_{c1} = 2\ mm$$
$$D_{c2} = 4\ mm \tag{1.36}$$

5.2.7 Konektori

Na kraju su odabrani potrebni konektori, pločice za spajanje i aluminijski ekstrudirani profili vidljivi na slici 32. i 33.

5.2.8 Popis dijelova

Tablica 4. Dijelovi hvataljke za kutije [9]

	Ime dijela	Kom/m	Cijena (kom/m)	Masa (kom/m)
1.	PFYN 60 VU1-72 G1/4 IG	6	€28,80	0,0237 kg
2.	Crijevo VSL 8-6 PE	10 m	€1.04	0,203 kg
3.	FSTE G1/4-AG 25	6	€49.66	0,145 kg
4.	STV-GE G1/8-AG 8	11	€4.03	0,0163 kg
5.	STW-GE G1/8-AG 8	2	€9,23	0,043 kg
6.	VTR G1/4-IG 5xG1/8	2	€21,06	0,090 kg
7.	VRS-SB G1/8-AG ISKT O-Ring	4	€1.69	0,006 kg
8.	STV-W G1/4-AG 6	1	€8.71	0,037 kg
9.	STV-W G1/4-AG 8	1	€9.88	0,049 kg
10.	Crijevo VSL 6-4 PE	3 m	€0.78	0,145 kg
11.	Crijevo VSL 4-2 PE	3 m	€0.70	0,087 kg
12.	SCP 15 NO AS	1	€296.40	0,275 kg
13.	SCP 10 NO AS	1	€254.80	0,275 kg
14.	STV-W G1/8-AG 6	1	€7.93	0,030 kg
15.	STV-W G1/8-AG 8	1	€9.23	0,043 kg
16.	HTR-STA 55 1N D20 WI	6	€17.81	0,070 kg
17.	BEF-WIN 40x40x40 5 MO-PROF	8	€10.92	
18.	NUT-STEI 10x20 M6-IG	34	€3.38	0,005 kg
19.	MO-PROF 40 3TN AL-1 470 mm	2	€53.30	1,8 kg
20.	MO-PROF 40 3TN AL-1 230 mm	2	€53.30	1,8 kg
21	Aluminijska ploča	1		1,482 kg

Ukupna masa hvataljke bez vijaka, cijevi, matica i pločica iznosi oko 6,82 kg.

Ukupna cijena bez vijaka i troškova izrade aluminijske ploče iznosi €2,073.91 prema [9] .

Nacrt aluminijske ploče nalazi se u prilogu.

5.2.9 Provjera hvatanja najteže kutije sa dvije vakuumske kapice:

Masa kutije P3 aparata (najteža kutija koja će se podizati sa 2 kapice:

$$m_{P3} = 5{,}4\ \ kg \tag{1.37}$$

Sila držanja („holding force“):

$$F_H = m_{P3} \cdot \left(g + \frac{a}{\mu}\right) \cdot S \tag{1.38}$$

$$F_H = 195{,}95\ \ N \tag{1.39}$$

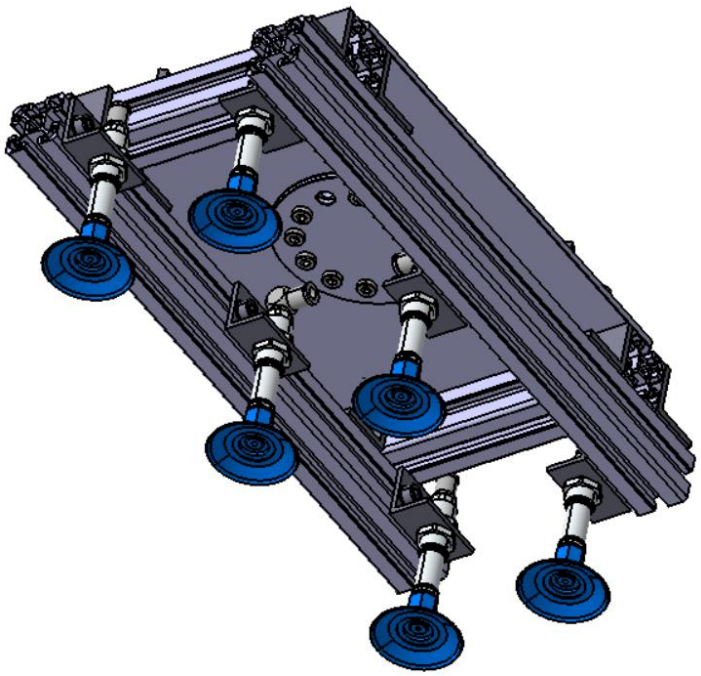
Sila usisa („suction force“):

$$F_S = \frac{F_H}{n} \tag{1.40}$$

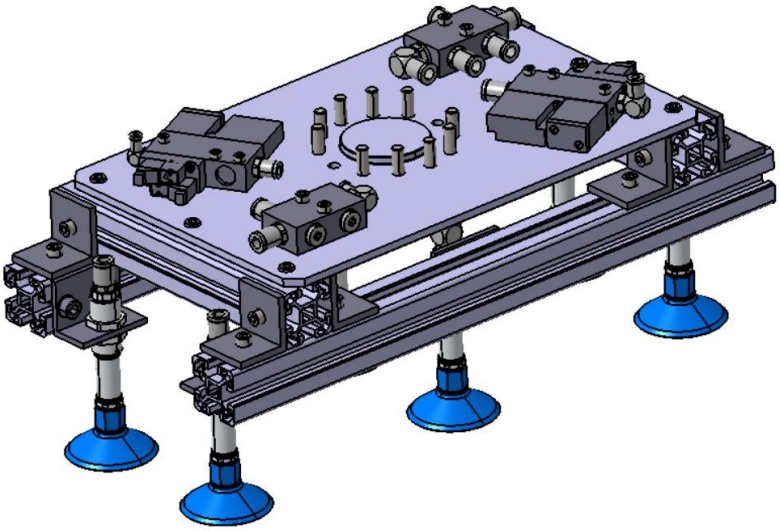
Gdje n predstavlja broj kapica (u ovom slučaju 2):

$$F_S = 97{,}98\ \ N < \ \ F_{S,cap} \tag{1.41}$$

Dakle hvataljka zadovoljava i u ovom slučaju.



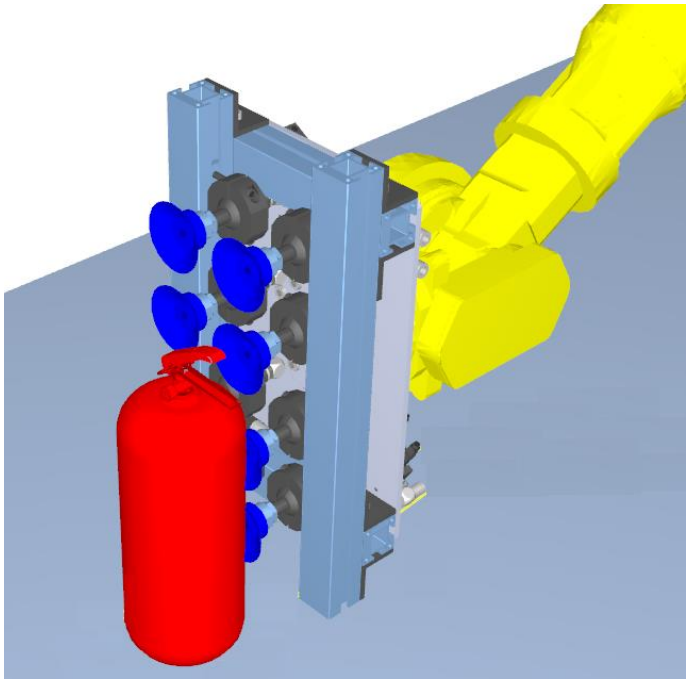
Slika 32. Vakuumska hvataljka za kutije



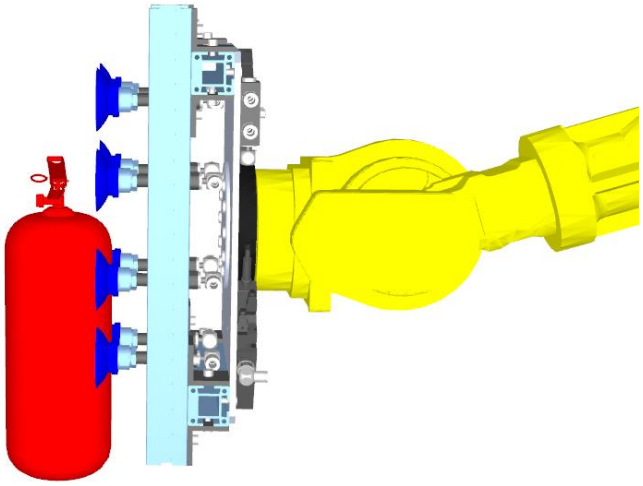
Slika 33. Vakuumska hvataljka za kutije (drugi pogled)

5.3 Načini hvatanja

Hvataljke su izvedene na način da se dio kapica aktivira s jednim generatorom, a dio sa drugim. Ideja je da s istom konstrukcijom (hardverom) postoji mogućnost hvatanja svih veličina spremnika i kutija. Jedino što se mijenja je digitalni signal, to jest naredba u programu na način da se kod većih aparata (P6,P9,P12) šalju dva digitalna signala istovremeno tako da se aktiviraju sve kapice, a kod manjih (P1,P2,P3) samo jedan. Na slikama 34. i 35. vidi se način hvatanja kad je aktivan samo jedan generator kod hvatanja aparata P3.



Slika 34. Hvatanje aparata P3 sa četiri vakuumske kapice



Slika 35. Hvatanje aparata P3 sa četiri vakuumske kapice (bočni pogled)

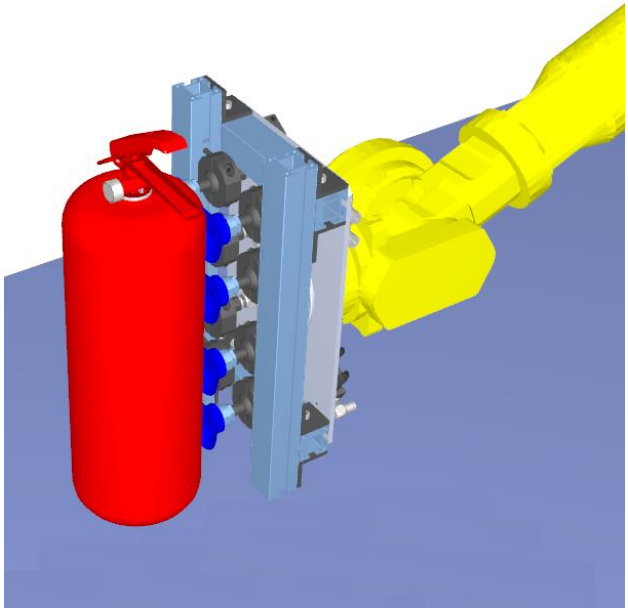
Držači kapica postavljeni su na kuglične oslonce tako da se kapice prilagode obliku predmeta.

U simulaciji nije simulirano prilagođavanje kapica ali može se vidjeti njihov okvirni položaj i način hvatanja.

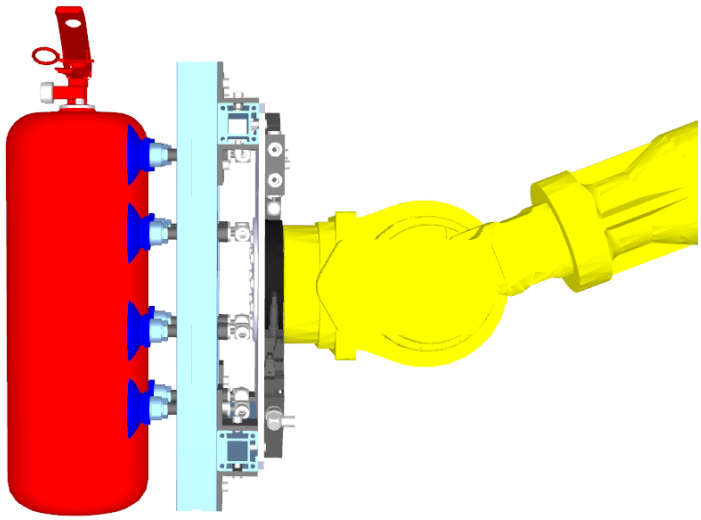
Na isti način hvataju se i aparati P1 i P2.

Na slikama 36. i 37. vidi se drugi način hvatanja koji se koristi kod većih aparata. U ovom slučaju kao primjer je uzet aparat P6. Aktivna su oba generatora (dva digitalna signala) i koristi se svih osam kapica.

Na isti način hvataju se aparati P9 i P12.



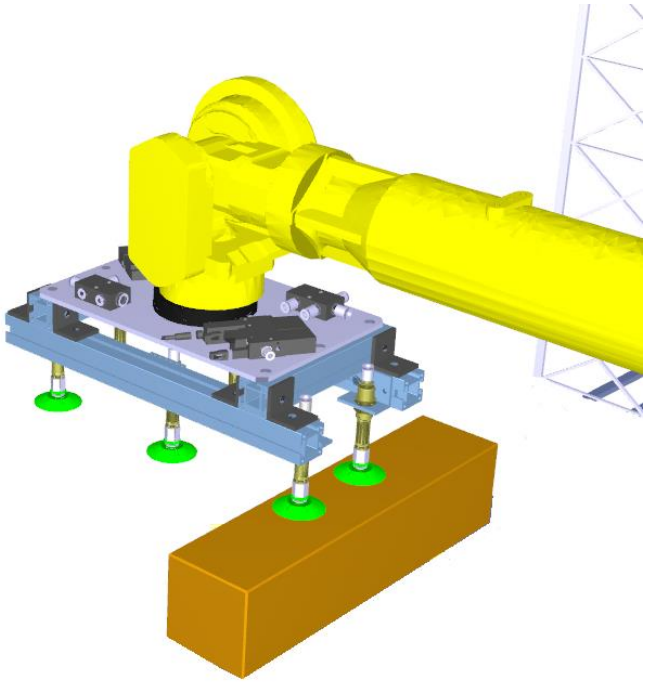
Slika 36. Hvatanje aparata P6 sa osam vakuumskih kapica



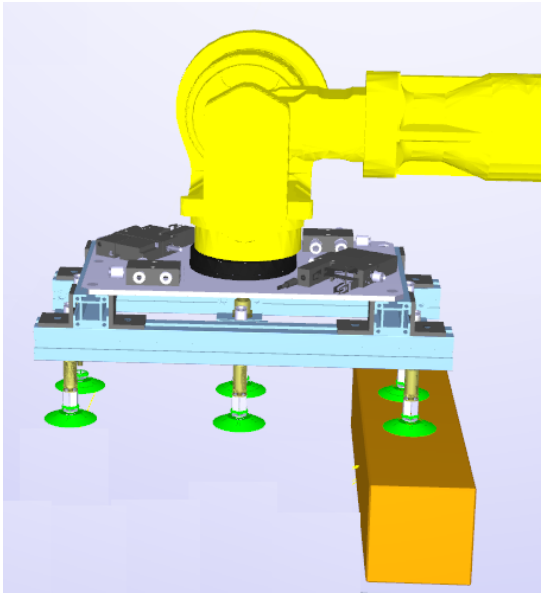
Slika 37. Hvatanje aparata P6 sa osam vakuumskih kapica (bočni pogled)

Sličan princip korišten je i kod hvatanja kutija. Kutije u koje se pakiraju aparati P1,P2 i P3 se hvataju sa aktivacijom jednog generatora (2 kapice) , a kutije P6,P9 i P12 sa aktivacijom oba generatora (4 kapice).

Razlika je u tome što hvataljka kod hvatanja kutija ima šest kapica i ima obične držače. S obzirom da je kutija ravna, te nema potrebe za prilagođavanjem oblika kapica obliku predmeta. Na slikama 38. i 39. može se vidjeti način hvatanja manjih kutija.



Slika 38. Hvatanje manjih kutija sa dvije vakuumske kapice

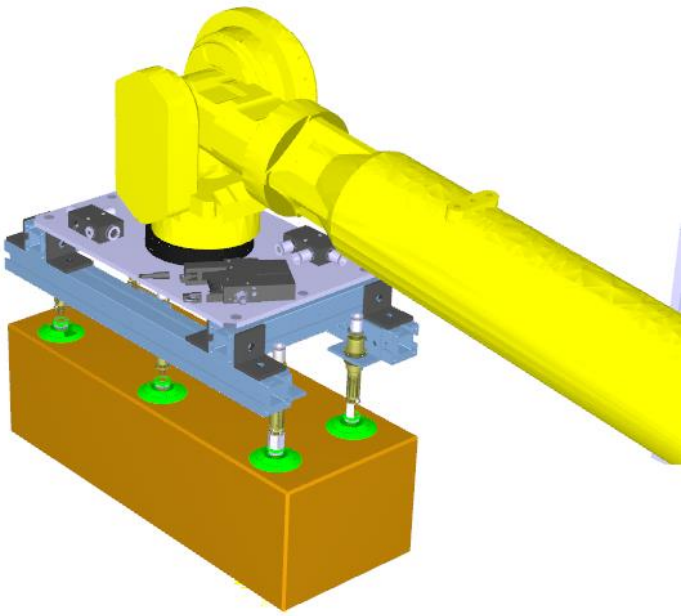


Slika 39. Hvatanje manjih kutija sa dvije vakuumske kapice (drugi pogled)

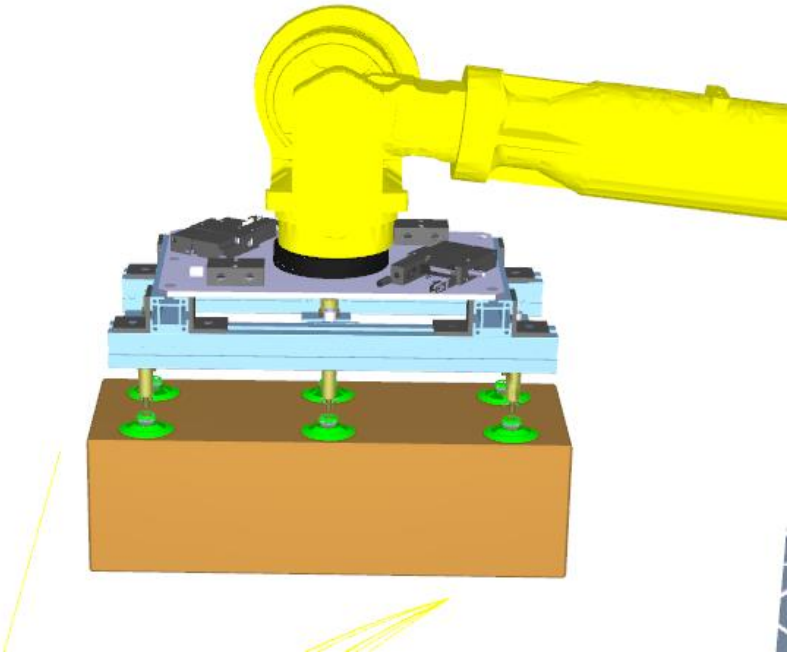
Na slikama 38. i 39. korištena je kutija aparata P3.

Na slikama 40. i 41. može se vidjeti način hvatanja većih kutija.

Za ovaj primjer je korištena kutija aparata P6 koji se smatra referentnim.



Slika 40. Hvatanje većih kutija sa šest vakuumskih kapica



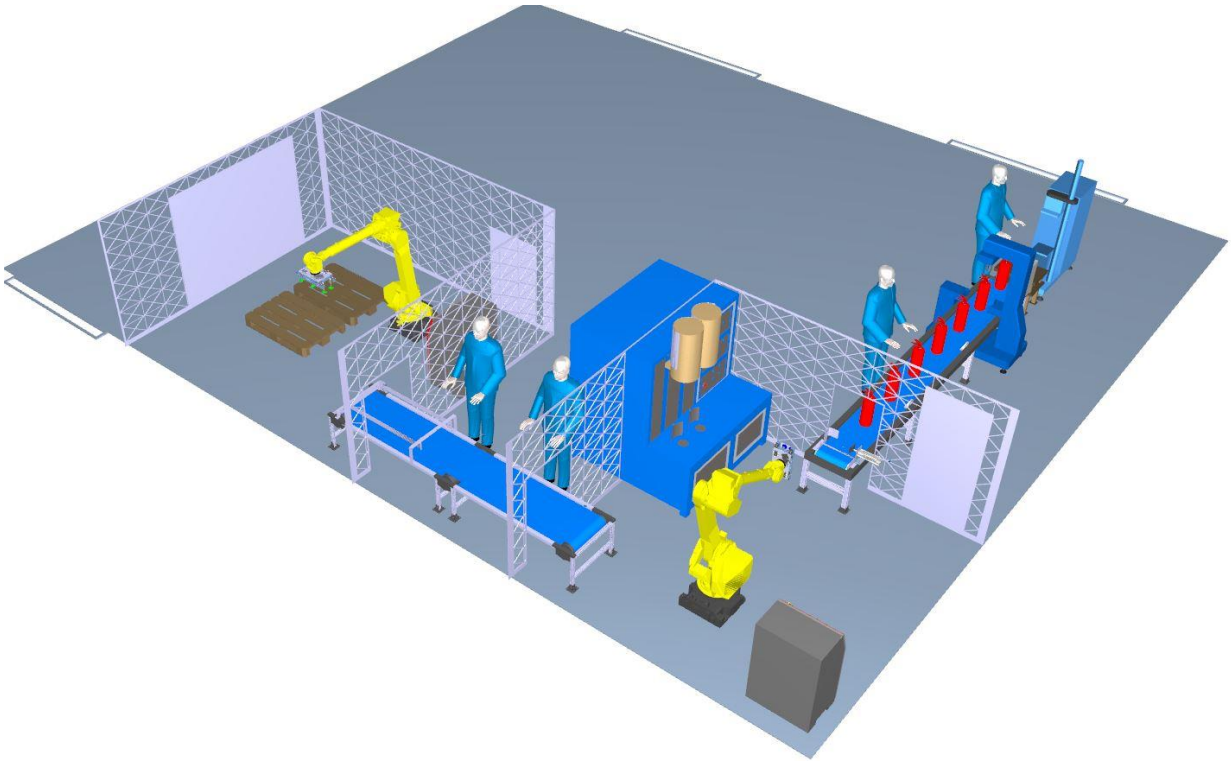
Slika 41. Hvatanje većih kutija sa šest vakuumskih kapica (drugi pogled)

6. SIMULACIJA U ROBODK

Napravljena je simulacija u softveru RoboDK koji se koristi za offline programiranje i simuliranje industrijskih robota.
U ovom slučaju je iskorišten za izradu simulacije i okvirnu analizu vremena ciklusa da bi se vidjela isplativost i efikasnost linije. Pravi robotski programi su napravljeni u Roboguide-u koji je specijalizirani softver za programiranje Fanucovih robota (vidjeti poglavlje 7.).

6.1 Kompletna linija

Napravljena je simulacije i analiza cijele linije koja se vidi na slici 42.



Slika 42. Automatska linija u RoboDK

Svi objekti su modelirani u programskom paketu Catia. Napravljeni su programi za senzore na način da se detektira kolizija između različitih objekata u RoboDK.

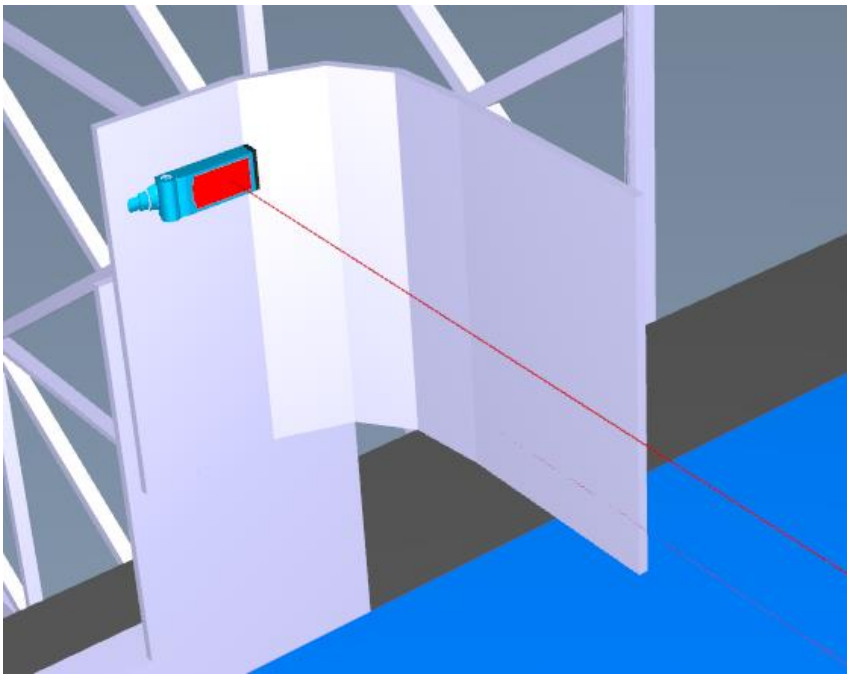
Na slici 43. vidi se logika funkcioniranja senzora u RoboDK programirana pomoću Pythona.

Nakon što se definira objekt i predmet koji se želi detektirati napravi se lista u kojoj se nalaze svi objekti iz stanice.

Ako dođe do kolizije bilo kojeg predmeta i zrake senzora, program provjerava da li taj predmet ima ime koje je zadano na početku programa kao željeni objekt detekcije. Ako ima, stanje senzora mijenja se u jedinicu.

```python
# Definiranje senzora
SENSOR_NAME = 'Sensor SICK WL4S_2'
SENSOR_VARIABLE = 'SENSOR2'

# Ime objekta koji se detektira
PART_KEYWORD = 'Ventil '

# Update status every 1 ms
RECHECK_PERIOD = 0.001

# Definiranje sensora iz simulacijskog prostora
SENSOR = RDK.Item(SENSOR_NAME, ITEM_TYPE_OBJECT)

# Lista svih objekata sa zadanim nazivom
all_objects = RDK.ItemList(ITEM_TYPE_OBJECT, False)
part_objects = []
for obj in all_objects:
    if obj.Name().count(PART_KEYWORD) > 0:
        part_objects.append(obj)

# Beskonačna petlja
detected_status = -1
while True:
    detected = 0
    for obj in part_objects:
        # Provjerava prisutnost objekta na način da modelirana zraka ulazi u koliziju sa predmetom
        if SENSOR.Collision(obj) > 0:
            detected = 1
            break

    # Osvježavanje statusa senzora
    if detected_status != detected:
        detected_status = detected
        RDK.setParam(SENSOR_VARIABLE, detected_status)
        print('Object detected status --> %i' % detected_status)

    # Ispituje da li je predmet prisutan svakih "RECHECK_PERIOD" sekundi
    pause(RECHECK_PERIOD)
```

Slika 43. Python program za simuliranje senzora u RoboDK



Slika 44. Model laserskog senzora sa vidljivom zrakom u RoboDK

Nakon toga, napravljeni su programi za kretanje traka.

```
distance = 608.33
SENSOR_VARIABLE_3 = 'SENSOR3' # station variable

# Definiranje objekata iz simulacijskog okruženja
traka_2             = RDK.Item('Traka_2', ITEM_TYPE_ROBOT)
frame_conv_2        = RDK.Item('Traka_2', ITEM_TYPE_FRAME)
frame_conv_moving_2 = RDK.Item('Pomicni_koord_dva', ITEM_TYPE_FRAME)
spremnik            = RDK.Item('Spremnik', ITEM_TYPE_OBJECT)

# Definiranje targeta
target_home_conv = RDK.Item('Home_conv_2', ITEM_TYPE_TARGET)
target_move_conv = RDK.Item('Move_conv_2', ITEM_TYPE_TARGET)

# Postavljanje brzine kretanja trake
traka_2.setSpeed(250)

# Parametri i definicija beskonačne petlje
n = -1
i = 0

while i > n:
# Svaki put kada senzor detektira da je stavljen spremnik na prvu transportnu traku, traka se pomakne za
# definiranu udaljenost
            if RDK.RunMode() == RUNMODE_SIMULATE:
                if RDK.getParam(SENSOR_VARIABLE_3) == 0:
                    distance_new = i*distance
                    traka_2.setPoseFrame(frame_conv_2)
                    target_move_conv_new = target_move_conv.Pose()*transl(distance_new,0,0)
                    traka_2.MoveJ(target_move_conv_new)
                    pause(10)
                    i=i+1
                else:
                    pause(0.001)
```

Slika 45. Python program za upravljanje trakama u RoboDK

Nakon definiranja traka i parametara, program ulazi u beskonačnu u petlju u kojoj pomakne traku za zadanu udaljenost svaki puta kada senzor detektira spremnik.

Potom je napravljen program za simuliranje pneumatskog cilindra koji stisne predmet nakon što ga senzor detektira na zaustavnom mjestu kod izuzimanja.

To je napravljeno tako da je koordinatni sustav cilindra izveden kao „gripper“ koji uhvati predmet i pomakne ga prema mjestu izuzimanja.

Programski kod je vidljiv na slici 46.

```python
SENSOR_VARIABLE = 'SENSOR'

# Definiranje objekata iz simulacije
Trigger_mech        = RDK.Item('Trigger_mech', ITEM_TYPE_ROBOT)
frame_conv_trigg    = RDK.Item('Trigger_frame_main', ITEM_TYPE_FRAME)
fanuc_base          = RDK.Item('Fanuc M-710iC/45M Base',ITEM_TYPE_FRAME)
tool                = RDK.Item('Tool 2', ITEM_TYPE_TOOL)
Move_trigg          = RDK.Item('Move_trigg', ITEM_TYPE_TARGET)
Home_trigg          = RDK.Item('Home_trigg', ITEM_TYPE_TARGET)

def TCP_On(toolitem):
    # Funkcija koja hvata najbliži predmet za koordinatni sustav definiranog mehanizma ili robota
    toolitem.AttachClosest()
    toolitem.RDK().RunProgram('TCP_On()');

def TCP_Off(toolitem, itemleave=0):
    # Funkcija koja otpušta predmet sa koordinatnog sustava definiranog mehanizma ili robota
    toolitem.DetachAll(itemleave)
    toolitem.RDK().RunProgram('TCP_Off()');

Trigger_mech.setSpeed(350)
Trigger_mech.setPoseFrame(frame_conv_trigg)

# Parametri i postavljanje beskonačne petlje
i = 0
n = -1
while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        # Ako se predmet detektira, koordinatni sustav (pneumatski cilindar) gura aparat na mjesto izuzimanja
        while RDK.getParam(SENSOR_VARIABLE) == 0:
            pause(0.001)
        if RDK.getParam(SENSOR_VARIABLE) == 1:
            pause(0.5)
            TCP_On(tool)
            Trigger_mech.MoveJ(Move_trigg)
            TCP_Off(tool,fanuc_base)
            Trigger_mech.MoveJ(Home_trigg)
            pause(5)
```

Slika 46. Python program za simuliranje pneumatskog cilindra u RoboDK

Potom je napravljen program za Fanucovog robota na prvoj robotskoj stanici. Zadatak robota je da izuzima aparate sa trake, opslužuje stroj za ispitivanje propusnosti aparata helijem i nakon završetka testiranja izuzima aparate i prosljeđuje ih dalje na traku prema paletizaciji.

Program je vidljiv na slikama 47. , 48. i 49.

Prvo se definiraju objekti, senzori, „targeti“ i koordinatni sustavi.

Potom slijedi definiranje Python funkcija za aktivaciju i deaktivaciju hvataljke, izuzimanje aparata sa trake, odlaganje aparata na stroj, izuzimanje sa stroja nakon testiranja i odlaganje aparata na traku prema paletizaciji.

Funkcije se vide na slikama 47. i 48.

```
87  def Take_chamber(robot,target):
88      # Funkcija za izuzimanje predmeta nakon ispitivanja
89      target_app = target.Pose()*transl(0,50,-1.5*APPROACH)
90      target_up = target.Pose()*transl(0,50,0)
91      target_back = target.Pose()*transl(0,0,-1.5*APPROACH)
92
93      robot.MoveL(target_back)
94      robot.MoveL(target)
95      TCP_On(tool)
96      robot.MoveL(target_up)
97      robot.MoveL(target_app);
98
99  def Place_conv(robot):
100     # Funkcija za odlaganje predmeta na transportnu traku prema paletizaciji
101     robot.MoveJ(target_pass)
102     robot.MoveL(target_app_3)
103     robot.MoveL(target_up_3)
104     robot.MoveL(target_conv_3_place)
105     TCP_Off(tool,frame_conv_3)
106     robot.MoveL(target_back_3);
107
```

Slika 47. Python program – prva robotska stanica u RoboDK (1.dio)

```
46  # Definiranje aktivnog koordinatnog sustava i brzina robota
47  robot.setPoseFrame(frame_fanuc)
48  robot.setSpeed(-1,speedJ)
49  robot.setSpeed(speedL)
50
51  def TCP_On(toolitem):
52      # Funkcija za hvatanje predmeta pomoću grippera
53      toolitem.AttachClosest()
54      pause(0.1)
55      toolitem.RDK().RunMessage('Set air valve on')
56      toolitem.RDK().RunProgram('TCP_On()');
57
58  def TCP_Off(toolitem, itemleave=0):
59      # Funkcija za otpuštanje predmeta iz grippera
60      toolitem.DetachAll(itemleave)
61      pause(0.1)
62      toolitem.RDK().RunMessage('Set air valve off')
63      toolitem.RDK().RunProgram('TCP_Off()');
64
65  def Pick_conv(robot):
66      # Funkcija za prilazak i podizanje predmeta sa mjesta izuzimanja
67      pause(0.5)
68      robot.MoveJ(target_pick_app)
69      robot.MoveJ(target_pick)
70      TCP_On(tool)
71      robot.MoveL(target_pick_up)
72      robot.MoveL(target_pick_app_up)
73      robot.MoveJ(target_home);
74
75  def Place_chamber(robot,target):
76      # Funkcija za odlaganje predmeta u komoru za ispitivanje
77      target_app = target.Pose()*transl(0,50,-1.5*APPROACH)
78      target_up = target.Pose()*transl(0,50,0)
79      target_back = target.Pose()*transl(0,0,-1.5*APPROACH)
80
81      robot.MoveL(target_app)
82      robot.MoveL(target_up)
83      robot.MoveL(target)
84      TCP_Off(tool)
85      robot.MoveL(target_back);
86
```

Slika 48. Python program – prva robotska stanica u RoboDK (2.dio)

```
# Beskonačna petlja u kojoj robot ovisno o reakcijama senzora na mjestu izuzimanja, senzora koji mjere da li
# je pojedina komora otvorena ili zatvorena odlaže ili izuzima predmeta sa stroja za ispitivanje propusnosti
i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        # Simulate the sensor by waiting for the SENSOR status to turn to 1 (object present)
        while RDK.getParam(SENSOR_VARIABLE_3) == 0:
            pause(0.001)

        if RDK.getParam(SENSOR_VARIABLE_3) == 1:
            if i < 2:
                i = i + 1
                pause(3.0)

            robot.MoveJ(target_home)
            Pick_conv(robot)


            if RDK.getParam(SENSOR_VARIABLE_4) == 0:
                print("Load chamber 1")
                Place_chamber(robot,target_place_1)

                while  RDK.getParam(SENSOR_VARIABLE_5) == 1 : #& (if test_2_finished==1)
                    if RDK.getParam(SENSOR_VARIABLE_10) == 1:
                        print("Unload chamber 2 and put it on conv_3")
                        Take_chamber(robot,target_place_2)
                        Place_conv(robot)



            elif RDK.getParam(SENSOR_VARIABLE_5) == 0:
                print("Load chamber 2")
                Place_chamber(robot,target_place_2)

                while  RDK.getParam(SENSOR_VARIABLE_4) == 1 : #& (if test_1_finished==1)"""
                    if RDK.getParam(SENSOR_VARIABLE_9) == 1:
                        print("Unload chamber 1 and put it on conv_3")
                        Take_chamber(robot,target_place_1)
                        Place_conv(robot)
```

Slika 49. Python program – prva robotska stanica u RoboDK (3.dio)

Na kraju se program baca u beskonačnu petlju u kojoj se prate reakcije 4 senzora.

Senzor koji detektira dolazak predmeta na zaustavno mjesto, senzor detekcije otvorene komore, senzor detekcije zatvorene komore i senzor prisutnosti predmeta na mjestu komore.

Ovisno o reakcijama senzora robot uzima predmet i opslužuje ili izuzima predmete sa stroja za ispitivanje propusnosti.

Dodan je i dio programa koji usporava robota pri početnom opsluživanju stroja. Potom je napravljen program za otvaranje i zatvaranje komora nakon postavljanja aparata i signala sa senzora. Trajanje zatvorenosti komore stavljeno je na točno 25s [naredba pause] što je jednako vremenu trajanja testiranja po aparatu prema specifikaciji proizvođača [6].

```python
# Definiranje senzora prisutnosti predmeta i dva senzora koji javljaju da je komora otvorena ili zatvorena
SENSOR_VARIABLE_4 = 'SENSOR4'
SENSOR_VARIABLE_6 = 'SENSOR6'
SENSOR_VARIABLE_9 = 'SENSOR9' # station variable

# Definiranje objekata
Komora_1            = RDK.Item('Komora_1', ITEM_TYPE_ROBOT)
Komora_1_1            = RDK.Item('Komora_1', ITEM_TYPE_FRAME)

# Definiranje targeta
target_home_1 = RDK.Item('Home_komora_1', ITEM_TYPE_TARGET)
target_move_1 = RDK.Item('Move_komora_1', ITEM_TYPE_TARGET)


# Brzina otvaranja i zatvaranja komore
Komora_1.setSpeed(250)
Komora_1.setPoseFrame(Komora_1_1)

i = 0
n = -1

# Beskonačna petlja, zatvara komoru kad senzor detektira aparat, drži je zatvorenom dok traje test točno 25 s
# prema specifikaciji proizvođača, nakon toga otvara komoru
while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        if RDK.getParam(SENSOR_VARIABLE_4) == 1:
            Komora_1.MoveJ(target_move_1)
            pause(25)
            Komora_1.MoveJ(target_home_1)
            while RDK.getParam(SENSOR_VARIABLE_9) == 1:
                pause(0.001)
                if RDK.getParam(SENSOR_VARIABLE_6) == 1:
                    break
        else:
            pause(0.001)
```

Slika 50. Python program za simuliranje stroja za testiranje propusnosti

```python
i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        if RDK.getParam(SENSOR_VARIABLE_7) == 1:
            i = i+1

            if i==1:
                ventil_1.setVisible(False,False)
                kutija_1.setVisible(True,False)
                pause(5.0)
            elif i==2:
                ventil_2.setVisible(False,False)
                kutija_2.setVisible(True,False)
                pause(5.0)
```

Slika 51. Python program za simulaciju stavljanja aparata u kutije

Zatim je napravljen program koji simulira stavljanje aparata u kutije na način da se sakrije objekt u RoboDK koji predstavlja aparat, a pojavi se objekt koji predstavlja kutiju. Vidljiv je na slici 51.

Na kraju je napravljen program za Fanucovog robota na drugoj robotskoj stanici gdje se odvija paletizacija kutija.

```python
# Funkcija za aktivaciju grippera i hvatanje predmeta
def TCP_On(toolitem):
    pause(0.25)
    toolitem.AttachClosest()
    toolitem.RDK().RunProgram('TCP_On()');

# Funkcija za ispuštanje predmeta
def TCP_Off(toolitem, itemleave=0):
    pause(0.25)
    toolitem.DetachAll(itemleave)
    toolitem.RDK().RunProgram('TCP_Off()');

# Funkcija za izuzimanje predmeta
def Pick_box(robot):
    robot.setPoseFrame(frame_robot)
    robot.MoveJ(target_upp_pick)
    robot.MoveL(target_pick)
    pause(0.2)
    TCP_On(tool)
    robot.MoveL(target_upp_pick)
    robot.MoveJ(target_home);

# Funkcija za odlaganje predmeta na paletu
def Place_box(robot,target):
    robot.setPoseFrame(frame_paleta)
    robot.MoveJ(target_app_place)
    robot.MoveL(target)
    pause(0.2)
    TCP_Off(tool,frame_paleta)
    robot.MoveL(target_app_place)
    robot.MoveJ(target_home);

# Postavljanje brzine robota
robot.setSpeed(-1,speedJ)
robot.setSpeed(speedL)

# Brojači koji se koriste za izračunavanje pozicija odlaganja predmeta
i = 0
n = -1
m = 0
z = 0
j = 0
x = -1
y = 0

```

Slika 52. Python program – druga robotska stanica u RoboDK

Najprije su definirane funkcije za hvataljku, te hvatanje i otpuštanje predmeta na paletu. Postavljene su brzine robota i inicijalizirane su varijable koje će se koristiti kao brojači.

```
# Program za paletizaciju, izračunavanje koordinata nakon svake detekcije senzora. Prvo se miče po X-koordinati
# potom po Y i na kraju po Z-u. Ubačeno je i bočno odlaganje predmeta (rotacija za 90 stupnjeva) da se ispuni paleta
while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        # Simulate the sensor by waiting for the SENSOR status to turn to 1 (object present)
        while RDK.getParam(SENSOR_VARIABLE_8) == 0:
            pause(0.001)

        if RDK.getParam(SENSOR_VARIABLE_8) == 1:
            x = x + 1
            if x == 4:
                x = 0
                j = j + 1
            if j == 2:
                j = 0
                target_place_hor = target_place_2.Pose()*transl(z*x_box,0,-m*z_box)
                target_app_place = target_place_2.Pose()*transl(z*x_box,0,-m*z_box-height)
                Pick_box(robot)
                Place_box(robot,target_place_hor)
                z = z + 1
                y = y + 1
            if z == 2:
                z = 0
                m = m + 1
            if m == 8:
                m = 0
                robot.MoveJ(target_home)


            if y != 1:
                target_place_1 = target_place.Pose()*transl(-j*x2_box,-x*y_box-z*x_box,-m*z_box)
                target_app_place = target_place.Pose()*transl(-j*x2_box,-x*y_box-z*x_box,-m*z_box-height)
                Pick_box(robot)
                Place_box(robot,target_place_1)
            if y == 1:
                y = 0
                x = x - 1
```

Slika 53. Python program – druga robotska stanica u RoboDK (2.dio)

Zatim program ulazi u beskonačnu petlju, na signal senzora izuzima predmet sa trake, kalkulira mjesto odlaganja (na svaki signal se pomakne prvo po X-u pa po Y-u i na kraju po Z-u) te stavlja predmet na paletu.

Nakon analiza dobiveno je okvirno vrijeme ciklusa koje iznosi 30s. Što znači da svakih 30s dolazi kutija na zadnje zaustavno mjesto sa kojeg se izuzima i stavlja na paletu.

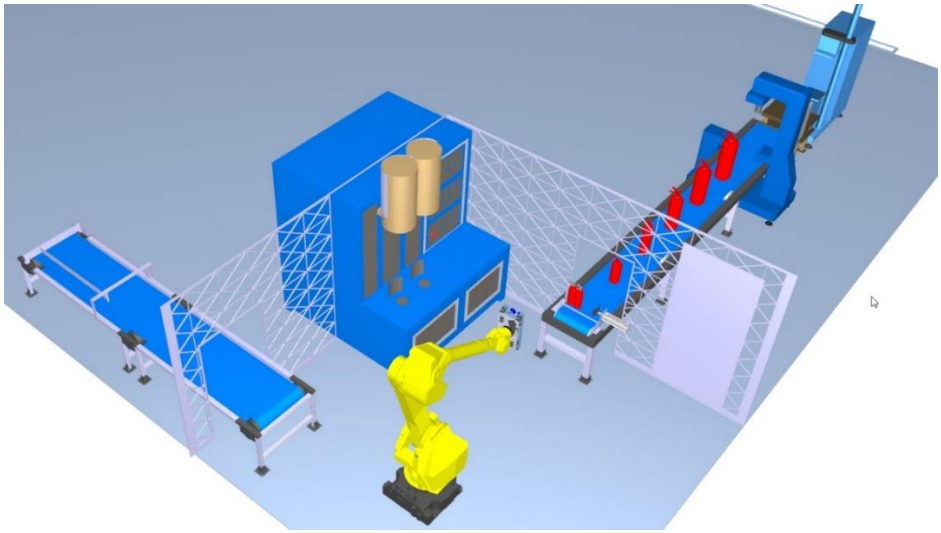
Ako uzmemo u obzir efektivno radno vrijeme trake od 7 sati (puno radno vrijeme je 8 sati) uz vrijeme ciklusa 30s dobijemo 840 komada po smjeni (trenutno oko 700).

Proces je, po potrebi, moguće dodatno ubrzati tako da se svaki 3. aparat ne testira nego prosljeđuje direktno prema traci za paletizaciju (testiranje po uzorku, 2 od 3 bi se testirala).

Također kod manjih aparata (npr. P1) postoji mogućnost hvatanja dva aparata i dvije kutije odjednom što bi dodatno ubrzalo rad linije.

6.2 Prva robotska stanica

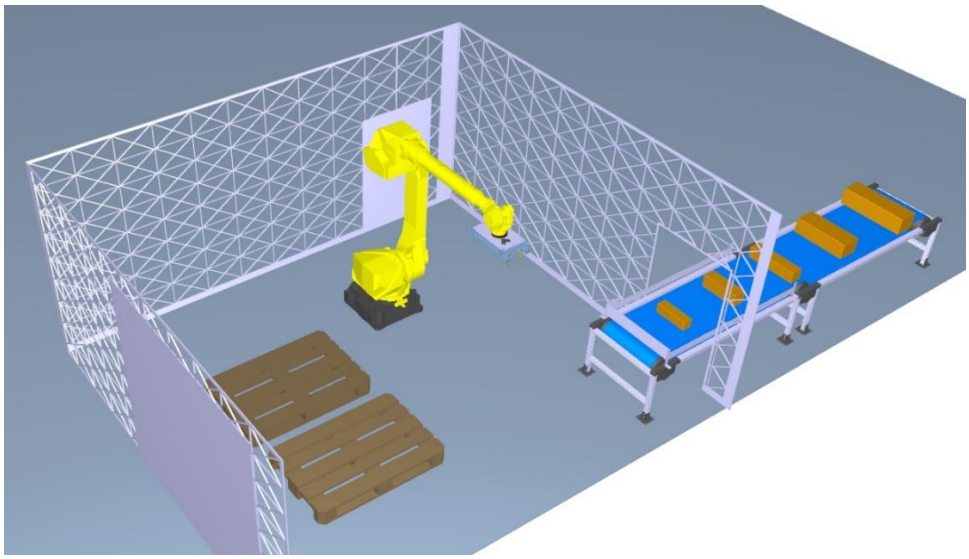
Nakon kompletne linije, napravljene su i simulacije za svaku robotsku stanicu posebno tako da se vidi način hvatanja aparata i kutija svih veličina.



Slika 54. Prva robotska stanica sa različitim vrstama aparata

Programska logika je slična kao i kod kompletne linije, a programski kodovi se mogu vidjeti u prilogu.

6.3 Druga robotska stanica



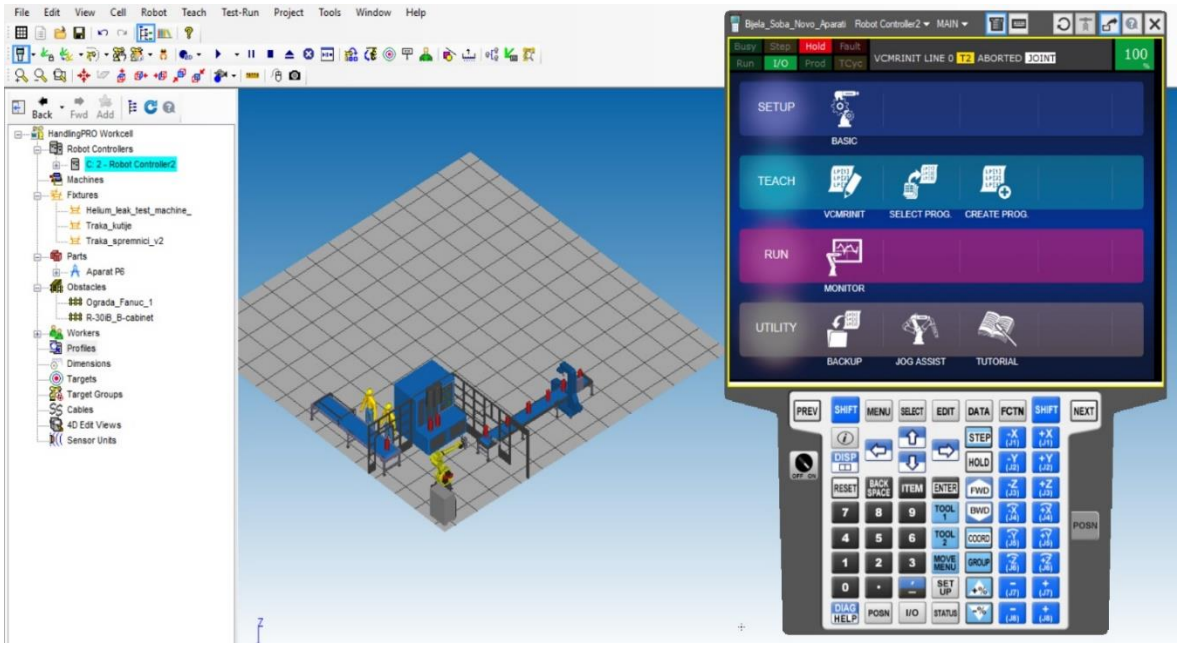
Slika 55. Druga robotska stanica sa različitim vrstama kutija

Druga robotska stanica je napravljena tako da se napravi simulacija hvatanja kutije za sve vrste to jest veličine aparata.

Programska logika je također vidljiva u prilogu.

7. ROBOGUIDE

Roboguide je specijalizirani programski paket za programiranje i simulaciju Fanucovih robota. U ovom diplomskom radu iskorišten je za izradu konkretnih robotskih programa, vizualizaciju i simulaciju robotskog ponašanja, te uspostavu i testiranje komunikacije za upravljanje linijom. Software nudi programiranje robota preko „Teach pada“ (privjeska za učenje) i Karela (programski jezik koji se koristi za dohvaćanje i postavljanje registara, digitalnih ulaza i izlaza, uspostavu komunikacije itd.). Praksa je da se svi pokreti koji se izvode na robotima programiraju preko TP, a komunikacija, naprednije funkcije i upravljanje robotskim kontrolerom pomoću Karela.

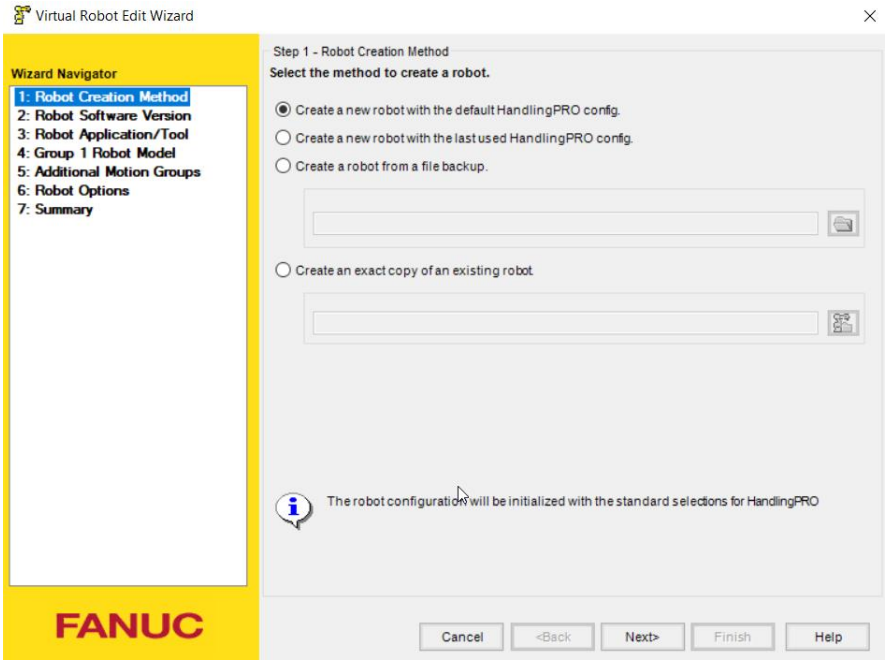


Slika 56. Roboguide korisničko sučelje sa TP-om

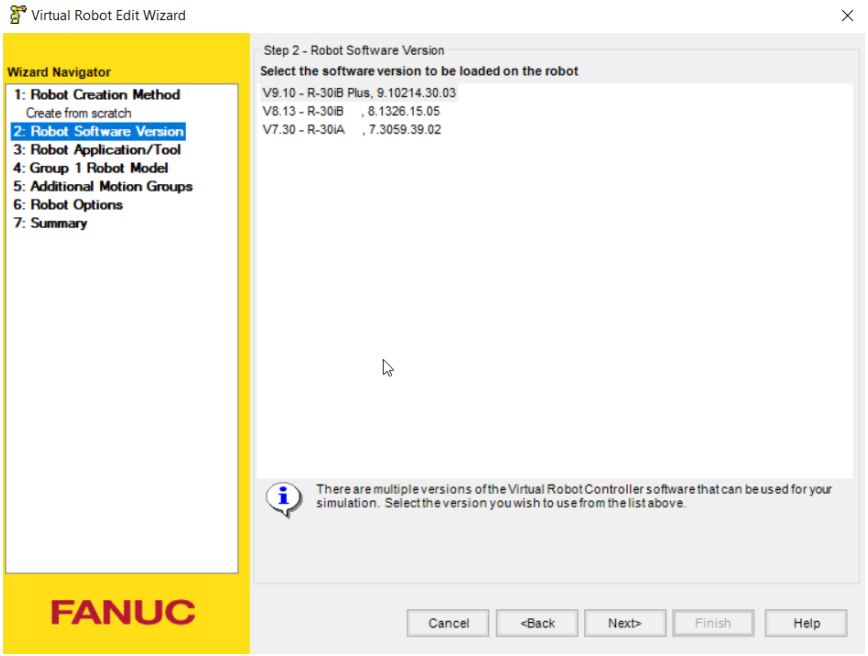
Kod definiranja nove robotske stanice otvara se prozor sa slike 57.

Ovdje je potrebno obratiti pažnju na grupe broj 2. i 6. U grupi broj 2. odabire se verzija robotskog kontrolera koji se nalazi na robotu. Da bi programi funkcionirali i da bi TP i korisničko sučelje bilo isto kao na pravom robotu potrebno je odabrati točno onu verziju koja

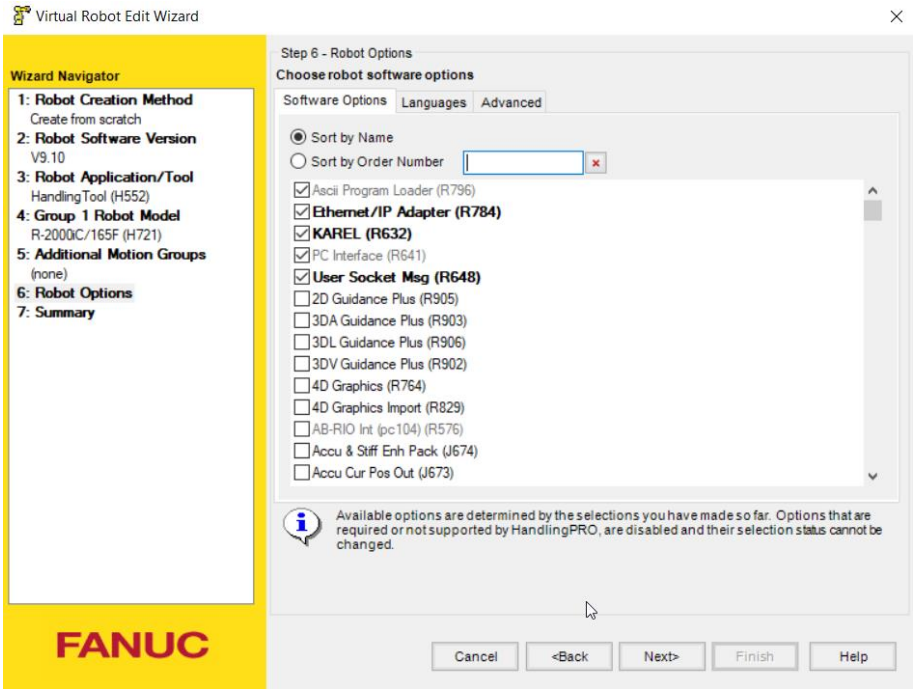
se nalazi na stvarnom robotu. S obzirom da se u okviru ovog projekta planiraju kupiti novi roboti odabran je najnoviji kontroler R30iB Plus kao što je vidljivo na slici 58. U grupi 6 potrebno je odabrati opcije koje se nalaze na robotu. Odabrane su opcije vidljive na slici 59. Osim onih koje su definirane po „defaultu“, odabrane su Ethernet IP Adapter, Karel i User Socket Msg. One nam omogućavaju uspostavu TCP/IP komunikacije sa robotom preko Etherneta koja će nam biti potrebna za razvoj aplikacije koja će upravljati parametrima linije.



Slika 57. Otvaranje nove robotske ćelije u Roboguide-u

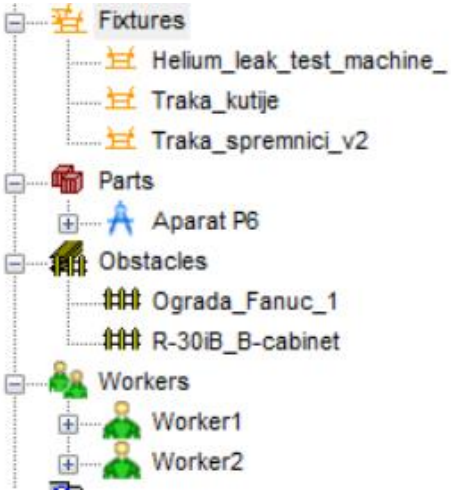


Slika 58. Odabir robotskog kontrolera u Roboguide-u



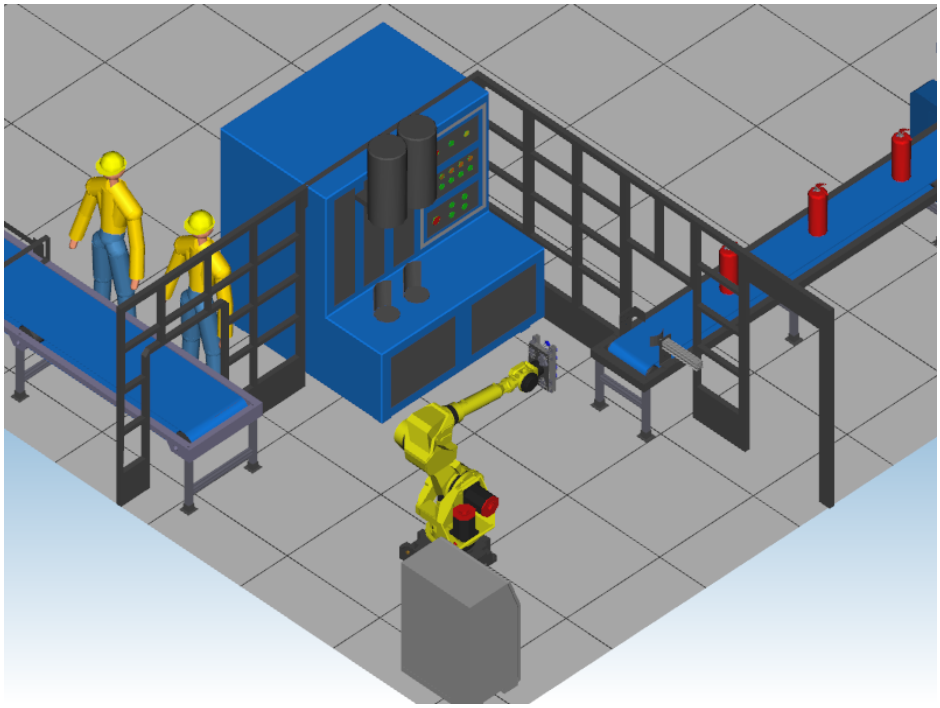
Slika 59. Odabir dodatnih opcija robotskog software-a u Roboguide-u

Nakon što se otvori nova ćelija i definiraju se sve opcije, u stanicu je potrebno ubaciti step/igs modele koji će poslužiti za izradu simulacije. Modele s kojih će se izuzimati i odlagati dijelovi potrebno je ubaciti pod polje „fixtures“. Modeli s kojima će roboti manipulirati potrebno je postaviti u polje „parts“. U polje „obstacles“ i „workers“ moguće je ubaciti neke popratne dijelove i modele radnika koji će se nalaziti unutar ćelije.



Slika 60. Odabir dodatnih opcija robotskog software-a u Roboguide-u

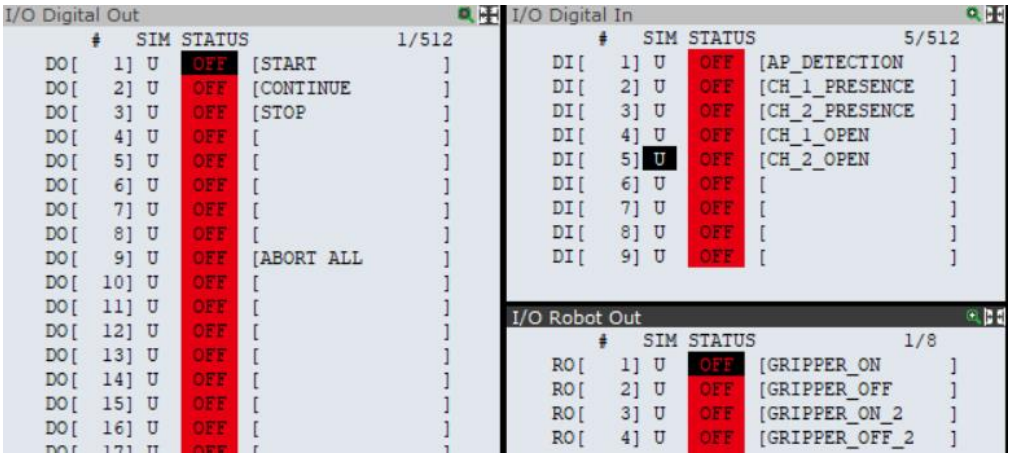
Na slici 61. vidljiva je prva robotska stanica nakon što se ubace svi potrebni modeli i postave sve opcije.



Slika 61. Prva robotska stanica u Roboguide-u

7.1 Programi za prvu robotsku stanicu

Nakon definiranja ćelije potrebno je otvoriti TP (privjesak za učenje) i postaviti digitalne/robotske ulaze/izlaze koji će se koristiti pri programiranju.

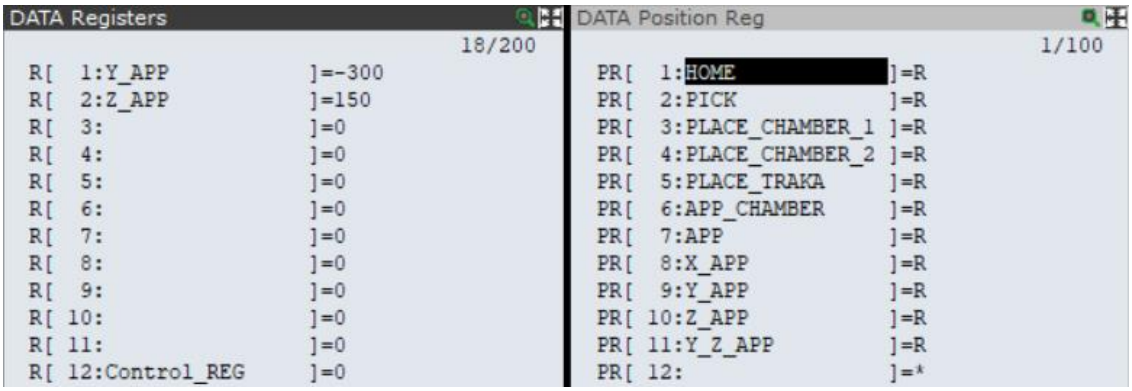


Slika 62. Definiranje digitalnih izlaza i ulaza

Digitalni izlazi će se kontrolirati preko aplikacije koja će biti prikazana i objašnjena u nastavku. Na slici 62. vidljivi su opisi svakog digitalnog izlaza. DO1 koristi se za pokretanje programa.

DO2 iako je naveden, u ovoj stanici nema funkciju, koristi se kod druge stanice kada robot čeka signal da su popunjene palete zamijenjene. DO3 se koristi za zaustavljanje programa, a DO9 za prekidanje svih programa (TP i Karel). RO1 i RO2 su predviđeni za kontrolu hvataljke. RO3 i RO4 također imaju istu funkciju s obzirom da se kod izuzimanja i odlaganja manjih aparata koristi samo dio kapica (pali se jedan RO), a kod većih aparata koriste se sve kapice (pale se dva RO). Način hvatanja objašnjen je u poglavlju 5.3.

Digitalni ulazi su signali koji dolaze sa senzora. DI1 je senzor koji se nalazi na traci i javlja robotu dolazak aparata na mjesto izuzimanja. DI2 i DI3 koriste se za detekciju dolaska aparata na mjesto testiranja. DI4 i DI5 javljaju robotu da li je komora otvorena ili zatvorena (da li je testiranje u tijeku ili je završeno).



Slika 63. Numerički i pozicijski registri

Nakon postavljanja digitalnih signala potrebno je postaviti registre. Registri služe za spremanje brojčanih vrijednosti u robotski kontroler koje će se kasnije koristiti kod programiranja robotskih kretnji. Razlikujemo numeričke i pozicijske registre. Numerički služe za spremanje samo jedne brojčane vrijednosti i obično se koriste za brojanje, definiranje udaljenosti prilaska robota i slično. Pozicijski služe za spremanje šest brojčanih vrijednosti (tri translacije po X,Y,Z osi i tri rotacije oko X,Y,Z osi) i koriste se za spremanje koordinatnih sustava ili točaka u prostoru.

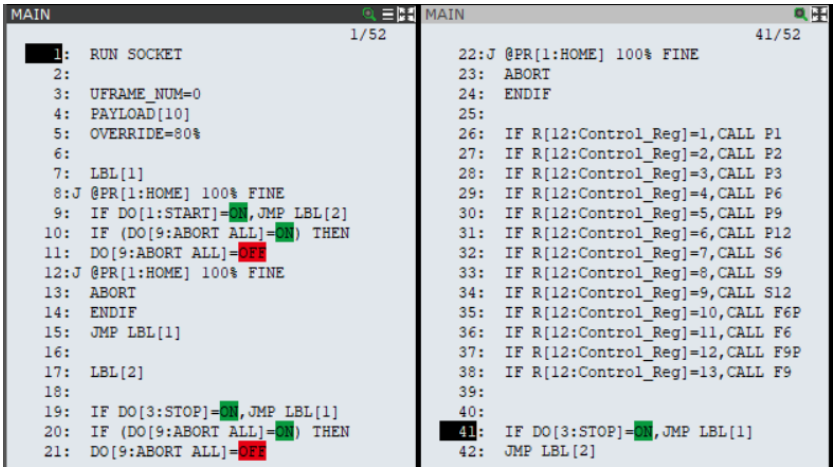
Na slici 63. vidljivi su numerički i pozicijski registri koji su korišteni kod programiranja prve robotske stanice. Numeričkih ima samo tri, od koji prva dva predstavljaju udaljenosti kod prilaska robota po Y osi i po Z osi. Registar R12 koristi se za odabir programa. Ovisno o broju koji se preko aplikacije šalje i sprema u taj registar, program na TP-u poziva program za određeni aparat. Programi za sve P aparate su gotovo jednaki, jedina razlika je u „targetima“ koji se mijenjaju ovisno o veličini aparata. Slična situacija je sa S i F aparatima. Pozicijski

registri su iskorišteni za spremanje točaka kroz koje se robot kreće. PR1 je početna pozicija robota, PR2 - PR5 su točke u kojima robot podiže ili odlaže predmet i PR6 – PR11 su prilazne točke trakama i stroju za ispitivanje propusnosti helijem. Sve navedene točke je potrebno snimiti u stvarnom okruženju na pravom robotu.

Nakon postavljanja svih potrebnih parametara, započinje se s programiranjem robotskih kretnji. Na TP se pritisne tipka „SELECT“ i potom odabere „CREATE“. Izgled prozora na TP vidljiv je na slici 64.

Slika 64. Kreiranje novog programa na TP

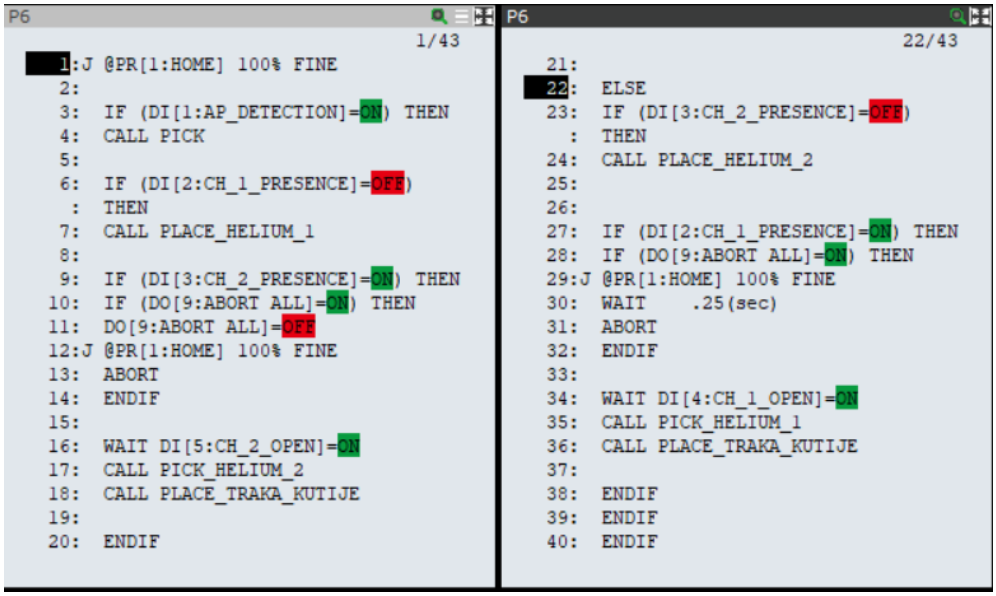
Prvi korak je kreiranje programa „MAIN“ koji će raditi kao glavni program iz kojega će se onda pozivati svi ostali potprogrami. Program je vidljiv na slici 65.



Slika 65. Program „MAIN“ na TP

Program prvo poziva Karel potprogram SOCKET pomoću naredbe RUN. On služi za komunikaciju između aplikacije i robotskog kontrolera i izvodi se paralelno sa programom na TP – u. Detaljnije je objašnjen u poglavlju 8.3. Nakon toga, definira se koordinatni sustav koji je u ovom slučaju globalni koordinatni sustav robota (označen sa 0). Postavlja se nosivost robota i brzina pomoću naredba „PAYLOAD“ i „OVERRIDE“.

Program potom ulazi u prvu beskonačnu petlju. Robot dolazi u početnu poziciju i čeka da se DO1 prebaci u „ON“ stanje, to jest da sa aplikacije dođe naredba da se pokrene program. U slučaju pokretanja, program izlazi iz prve beskonačne petlje i ulazi u drugu u kojoj prema broju koji je spremljen u registar R12 (to jest ovisno o naredbi koja je došla sa aplikacije o tome koji aparat se trenutno radi) pokreće program za definirani aparat. Program se izvodi sve do završetka osim ako sa aplikacije ne dođe signal da se program zaustavi (paljenje DO3 i vraćanje programa u prvu petlju u kojoj ponovo čeka signal za pokretanje programa). Ukoliko sa aplikacije dođe signal da se upali DO9 (gumb „TURN OFF“), prekida se izvođenje svih programa (TP i Karel).

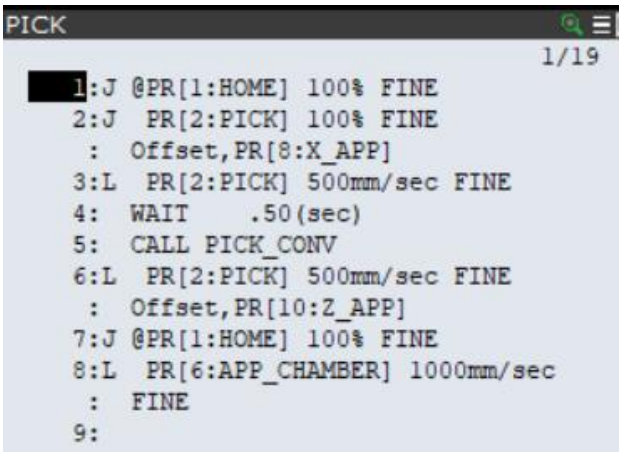


Slika 66. Program „P6“ na TP

Kao primjer je uzet program za P6 aparat. Robot ovisno o tome koji su digitalni ulazi DI aktivni prema objašnjenju sa slike 62. izuzima i odlaže vatrogasne aparate na mjesto komora za testiranje. Ukoliko su obje komore prazne robot odlaže aparat na mjesto prvo komore. Ako je prva komora prazna, a druga puna i testiranje je završeno tada robot popunjava mjesto prve komore, potom izuzima aparat se mjesta druge komore i odlaže dalje prema traci za paletizaciju.

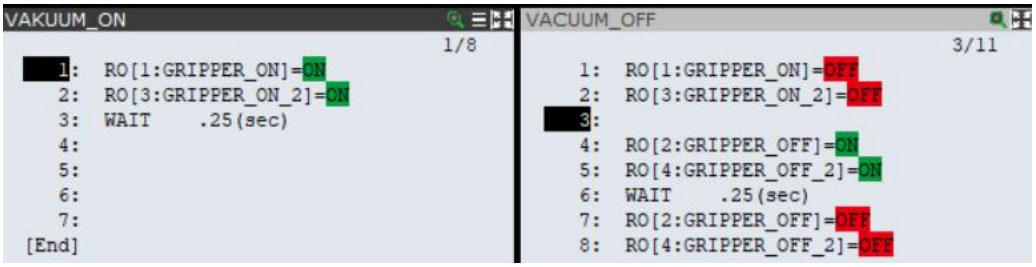
Slično, samo obratno vrijedi i ako je prva komora puna, a druga prazna. Ukoliko su obje komore pune i testiranje je u tijeku tada robot čeka prvu komoru koja završi, te sa nje izuzima aparat i šalje ga dalje prema paletizaciji.

U programima „PICK“ , „PLACE_HELIUM_1“ , „PLACE_HELIUM_2“ , „PICK_HELIUM_1“ , „PICK_HELIUM_2“ i „PLACE_TRAKA_KUTIJE“ programska logika je ista, a različiti su samo „targeti“ (točke kroz koje robot prolazi) i ponašanje hvataljke ovisno o tome da li se predmet izuzima ili odlaže. Za stavljanje aparata na mjesto testiranja i odlaganja na traku, napravljena su po dva programa zbog potrebe simulacije, u stvarnosti bi to bio jedan program. Kao primjer je pokazan program „PICK“ na slici 67.



Slika 67. Program „PICK“ na TP

Program za aktiviranje hvataljke „PICK_CONV“ je u ovom slučaju samo simulacijski program. Pravi program bi u slučaju P6 aparata aktivirao robotske izlaze RO1 i RO3 zato jer se on podiže sa svih osam kapica. U slučaju odlaganja predmeta RO1 i RO3 bi se ugasili, a RO2 i RO4 bi se samo impulsno upalili (stavili u „ON“ stanje na jako kratko vrijeme) jer oni služe za ispuštanje zraka kod spuštanja predmeta. Primjeri pravih programa za aktiviranje hvataljke kod podizanja i spuštanja prikazani su na slici 68.

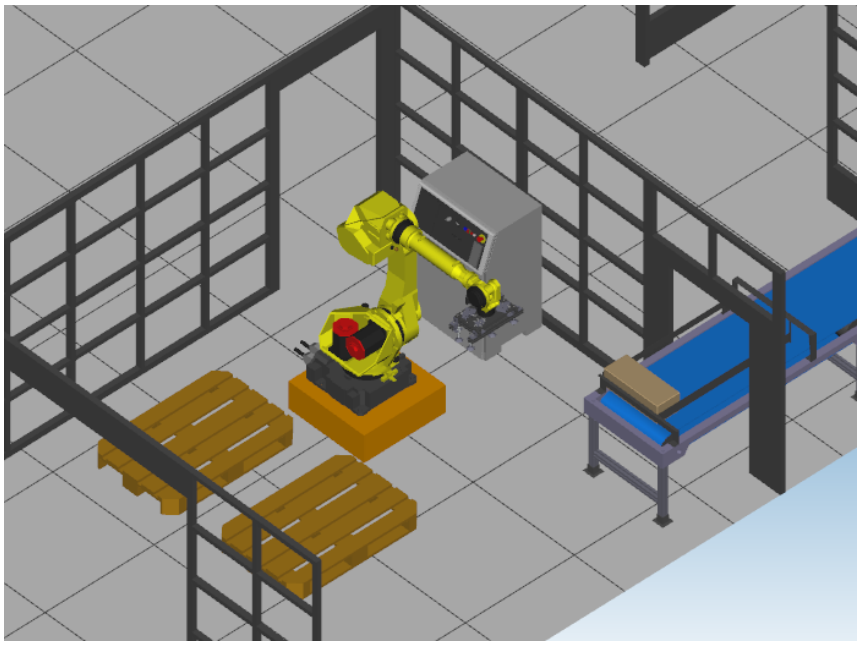


Slika 68. Programi za aktivaciju hvataljke na TP

Kompletni programi za prvu robotsku stanicu (robota za manipuliranje aparatima između traka i stroja za testiranje propusnosti) vidljivi su u prilogu.

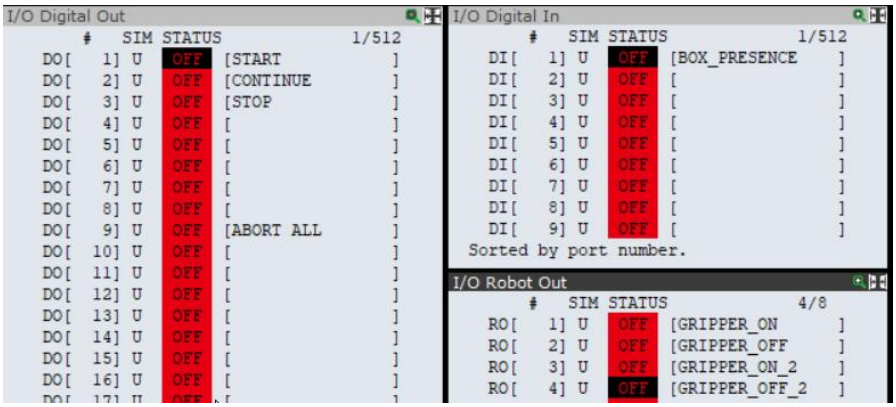
7.2 Programi za drugu robotsku stanicu

Druga robotska stanica sastoji se od trake na kojoj dolaze vatrogasni aparati zapakirani u kutije, Fanucovog M710iC – 50M robota i dvije palete za odlaganje kutija. Stanica je vidljiva na slici 69.



Slika 69. Druga robotska stanica u Roboguide-u

Kao i u poglavlju 7.1. nakon definiranja ćelije potrebno je postaviti digitalne ulaze i izlaze.



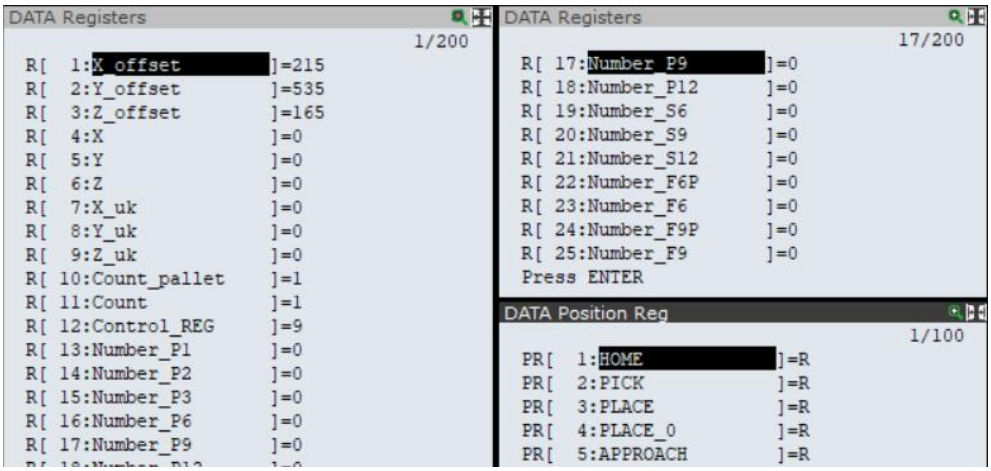
Slika 70. Definiranje digitalnih izlaza i ulaza – 2.robot

Na slici 70. vidimo da su digitalni izlazi DO jednaki kao i kod prve robotske stanice (vidjeti sliku 63.) izuzevši DO2 i DO18. Kao naredba „CONTINUE“ koristi se DO2. On je potreban

kada se napune obje palete, a želi se nastaviti raditi isti program. U tom slučaju će robot stati i čekati signal od aplikacije da su obje palete zamijenjene i da je sve spremno za nastavak rada.

DO18 koristit će se interno u programu za mijenjanje korisničkog koordinatnog sustava paleta, to jest neće biti kontroliran od strane aplikacije. On je iskorišten tako da se točka odlaganja treba snimiti samo prema jednoj paleti. Potom se odredi koordinatni sustav druge palete u odnosu na robota, unese u robotski kontroler i robot zna na koje točke treba odlagati predmete na drugu paletu.

Od digitalnih ulaza DI koji predstavljaju signale sa senzora imamo samo DI1. On nam predstavlja senzor koji detektira da je kutija došla na mjesto izuzimanja.



Slika 71. Numerički i pozicijski registri – 2.robot

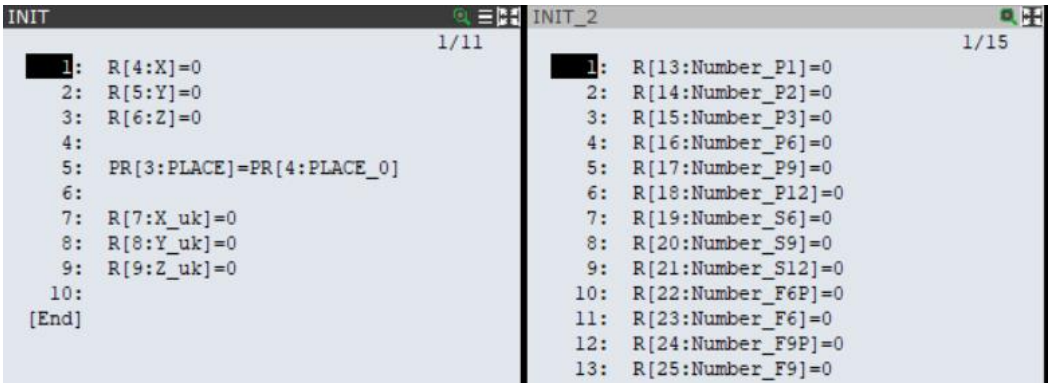
Nakon digitalnih signala potrebno je definirati registre. U numeričkim registrima imamo razmake po X,Y i Z osi koji predstavljaju udaljenosti između koordinatnih sustava kutija.

R4 – R6 nam služe kao inkrementalni brojači koji se koriste kod izračunavanja mjesta odlaganja kutija. R7 – R9 nam služe za spremanje ukupnih pomaka robota po X,Y i Z osi od definirane početne točke odlaganja. R10 nam služi za brojanje ukupnog broja kutija na jednoj paleti. To nam je bitno kada se jedna paleta u potpunosti napuni, da robot počne odlagati kutije na drugu paletu. R11 nam je registar koji broji ukupan broj napravljenih komada unutar jednog programa i njega je u bilo kojem trenutku moguće provjeriti i resetirati preko aplikacije. R12 nam je kontrolni registar i preko njega kontroliramo koji program će se izvoditi. Registri R13 – R25 koriste se za brojanje odrađenih aparata na dnevnoj bazi.

Od pozicijskih registara imamo: PR1 - početnu poziciju, PR2 - poziciju izuzimanja, PR3 - poziciju odlaganja koju izračunavamo unutar programa, PR4 - početnu poziciju odlaganja

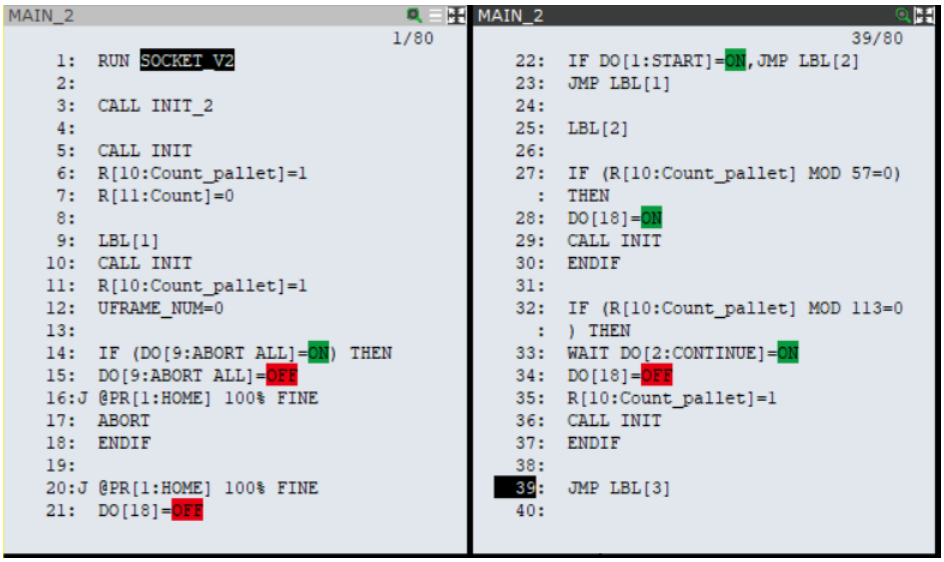
koju koristimo da prilikom pokretanja novog programa resetiramo PR3 i PR5 koji nam predstavlja udaljenost prilikom prilaska točkama izuzimanja i odlaganja.

Nakon definiranja parametara izrađeni su programi na TP.



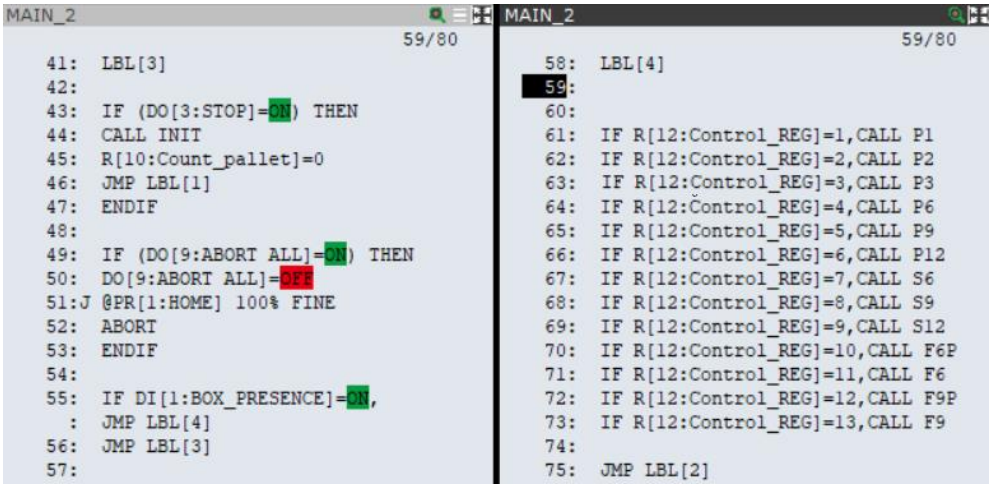
Slika 72. Program „INIT“ i „INIT_2“ na TP

Prvo je izrađen program za inicijalizaciju registara koji se koriste kod izračunavanja pozicije odlaganja i kod računanja broja komada na dnevnoj razini. Nakon toga je izrađen glavni program „MAIN_2“. Prvo se kao i kod prvog robota poziva Karel program za komunikaciju sa aplikacijom. Nakon toga se inicijaliziraju registri, te program ulazi u prvu beskonačnu petlju. U njoj se robot dovodi u početnu poziciju, gasi se DO18 što znači da će robot vršiti paletizaciju na prvu paletu i čeka se signal od aplikacije DO1 za pokretanje programa. Nakon signala, robot ulazi u drugi dio programa u kojem se provjerava da li je prva paleta napunjena pomoću naredbe MOD koja predstavlja cjelobrojno dijeljenje s ostatkom. Ako je ostatak 0 znači da je na paleti 56 kutija što je u ovom simulacijskom primjeru stavljeno kao maksimum kutija na jednoj paleti. Također se provjerava da li su obje palete napunjene i ako jesu tada program čeka signal aplikacije da može nastaviti.



Slika 73. Program „MAIN_2“ na TP – 1.dio

Nakon toga robot ulazi u treći dio programa. U beskonačnoj petlji se provjerava da li je od aplikacije došao signal za zaustavljanje programa i ako jest program se vraća na prvi dio – LBL[1]. Istovremeno se provjerava da li je DI1 aktivan to jest da li je senzor detektirao dolazak kutije na mjesto izuzimanja. Ako jest, program nastavlja na četvrti dio programa LBL[4] u kojem se ovisno o tome koji broj je spremljen u R12 poziva program za odgovarajući aparat.



Slika 74. Program „MAIN_2“ na TP – 2.dio

Kao i u slučaju prve robotske stanice i ovdje je kao primjer odabran referentni P6 aparat. Na slici 75. vidi se program za izračunavanje pozicije odlaganje kutije. Svaki put kad se program pozove, brojač po X-u se poveća za jedan. Brojač po Y-u se poveća za jedan tek kada se postave sve kutije po X-osi. Brojač po Z-u se povećava za jedan tek kada se napuni jedan cijeli red

palete. Potom se ovisno o stanjima brojača i ranije definiranim odmacima između kutija, izračunavaju pozicije odlaganja, te se pozivaju programi za izuzimanje i odlaganje kutija.

P6 1/39

```
  1:  UFRAME_NUM=1
  2:
  3:  R[7:X_uk]=R[1:X_offset]*R[4:X]
   :
  4:  R[4:X]=R[4:X]+1
  5:
  6:  IF (R[4:X]=5) THEN
  7:  R[5:Y]=R[5:Y]+1
  8:  R[4:X]=1
  9:  R[7:X_uk]=0
 10:  R[8:Y_uk]=R[2:Y_offset]*R[5:Y]
   :
 11:  ENDIF
 12:
 13:  IF (R[5:Y]=2) THEN
 14:  R[4:X]=1
 15:  R[5:Y]=0
```

P6 31/39

```
 16:  R[7:X_uk]=0
 17:  R[8:Y_uk]=0
 18:  R[6:Z]=R[6:Z]+1
 19:  R[9:Z_uk]=R[3:Z_offset]*R[6:Z]
   :
 20:  ENDIF
 21:
 22:
 23:  PR[3,1:PLACE]=PR[4,1:PLACE_0]+
   :  R[7:X_uk]
 24:  PR[3,2:PLACE]=PR[4,2:PLACE_0]+
   :  R[8:Y_uk]
 25:  PR[3,3:PLACE]=PR[4,3:PLACE_0]+
   :  R[9:Z_uk]
 26:
 27:  CALL PICK
 28:  CALL PLACE
```

Slika 75. Program „P6“ na TP – 2.robot

Program za izuzimanje predmeta je gotovo jednak programu kod prve robotske stanice. Kod programa za odlaganje prvo se provjerava stanje DO18. Ovisno o njemu aktivira se koordinatni korisnički sustav prve ili druge palete koji definira na koju paletu će robot odlagati kutije. U simulaciji se onda ovisno o odabranoj paleti, odabire i simulacijski program za otvaranje hvataljke. Kod stvarnog robota bi to sve bio jedan program „VACUUM_OFF“ kao što je prikazano kod prve robotske stanice u poglavlju 7.1. Ista stvar vrijedi i kod odlaganja predmeta.

PLACE

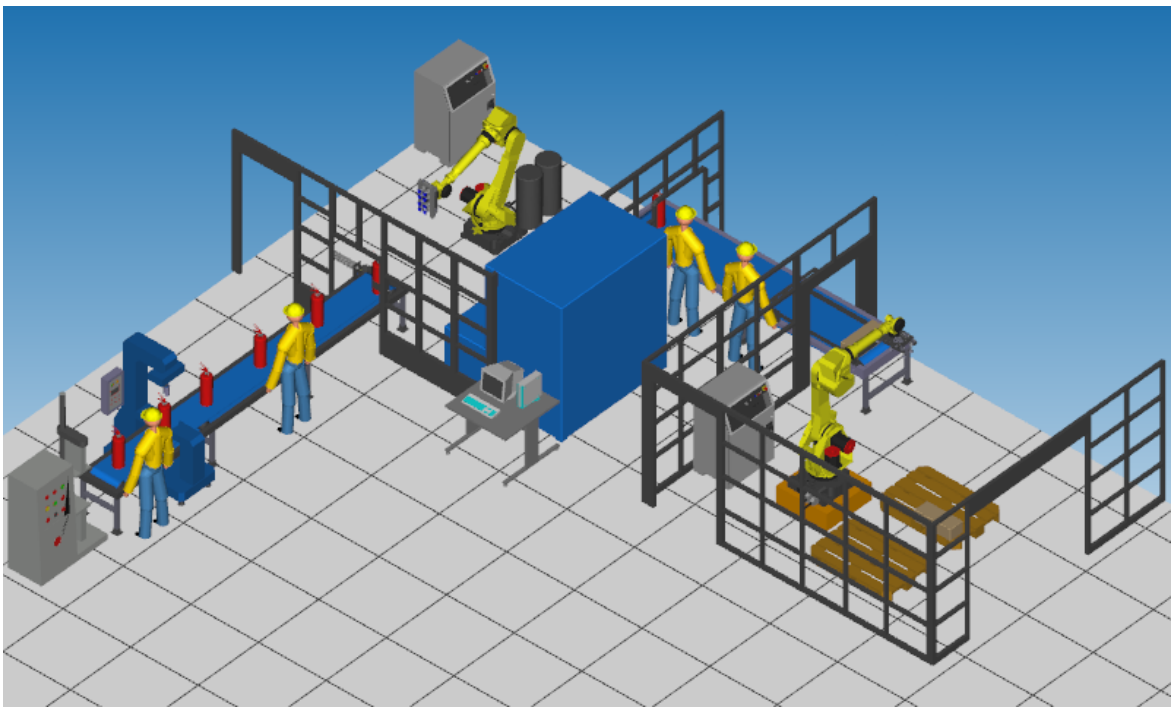
```
  1:  IF (DO[18]=OFF) THEN
  2:  UFRAME_NUM=1
  3:  ENDIF
  4:
  5:  IF (DO[18]=ON) THEN
  6:  UFRAME_NUM=2
  7:  ENDIF
  8:
  9:  PAYLOAD[1]
 10:  OVERRIDE=85%
 11:
 12:J  PR[3:PLACE] 100% FINE
   :  Offset,PR[5:APPROACH]
 13:L  PR[3:PLACE] 500mm/sec FINE
 14:
 15:  WAIT    .50(sec)
 16:
 17:  IF (DO[18]=OFF) THEN
 18:  CALL GRIPPER_OPEN_1
 19:  ENDIF
 20:
 21:  IF (DO[18]=ON) THEN
 22:  CALL GRIPPER_OPEN_2
 23:  ENDIF
 24:
 25:J  PR[3:PLACE] 100% FINE
   :  Offset,PR[5:APPROACH]
 26:  R[10:Count_pallet]=
   :  R[10:Count_pallet]+1
 27:
 28:  R[11:Count]=R[11:Count]+1
 29:
 30:  UFRAME_NUM=0
 31:J @PR[1:HOME] 100% FINE
```

Slika 76. Program „PLACE“ na TP – 2.robot

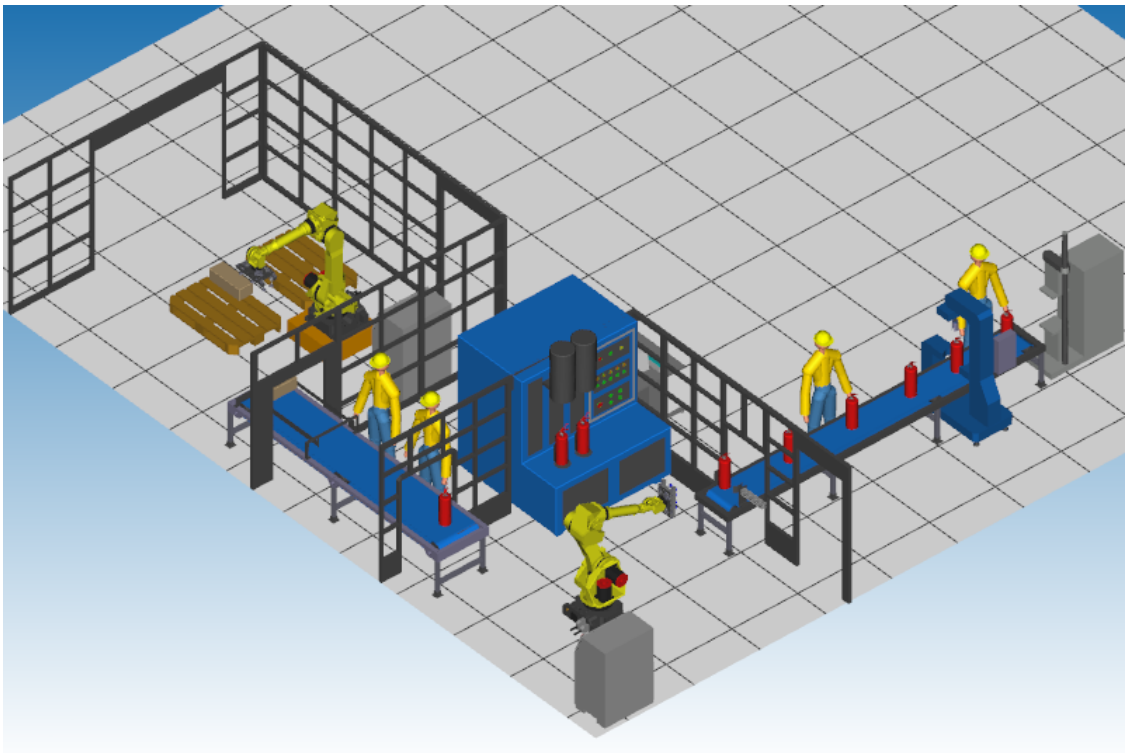
Programi napravljeni u simulacijskom okruženju u Roboguide-u se vrlo jednostavno mogu prebaciti na stvarne robote. Da bi programi točno funkcionirali u realnoj situaciji, potrebno je snimiti točne pozicije u koje će robot dolaziti.

Na kraju su obje robotske stanice ukomponirane u jednu i napravljena je simulacija upravljanja linijom pomoću aplikacije. Simulirano je povezivanje robota preko Etherneta, te slanje i primanje podataka unutar lokalne mreže.

Izgled kompletne linije u Roboguide-u sa ubačenim radnicima i računalom preko kojeg se linijom upravlja, prikazan je na slikama 77. i 78.



Slika 77. Proces završne montaže u bijeloj sobi - Roboguide



Slika 78. Proces završne montaže u bijeloj sobi – Roboguide, 2.pogled

8. APLIKACIJA ZA UPRAVLJANJE LINIJOM

Da bi se operaterima na proizvodnoj liniji olakšalo postavljanje parametara i upravljanje linijom, izrađena je aplikacija u programskom jeziku - Python. Aplikacija se otvara na stolnom računalo, laptopu ili tabletu koji je putem Ethernet kabela povezan sa robotima i po potrebi PLC-ovima koji upravljaju trakama i strojevima. Aplikacija je razvijena za kontroliranje dva Fanuc robota, ali se vrlo jednostavno može proširiti na više drugih robota ili PLC-ova. Komunikacija sa robotima odvija se pomoću TCP/IP protokola.

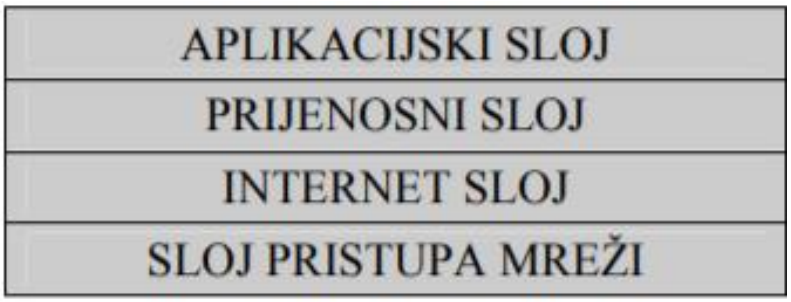
8.1 TCP/IP komunikacijski protokol

TCP/IP je jedan od najrasprostranjenijih komunikacijskih protokola koji se danas nalazi na gotovo svim računalima. Sastoji se od dva najčešće korištena protokola: TCP (Transmission Control Protocol) i IP (Internet Protocol). [12]

TCP protokol dijeli podatke u pakete koje mreža može učinkovito prenositi, potvrđuje da su svi paketi stigli na svoje odredište i ponovno sastavlja podatke [13].

IP protokol pakira i obrađuje podatke, omogućuje mreži da čita i prosljeđuje podatke na svoje odredište i da određuje koliko podataka stane u jedan paket. IP je odgovoran za usmjeravanje paketa između računala [13].

TCP/IP komunikacijski model dijeli se u četiri sloja prema slici 79.

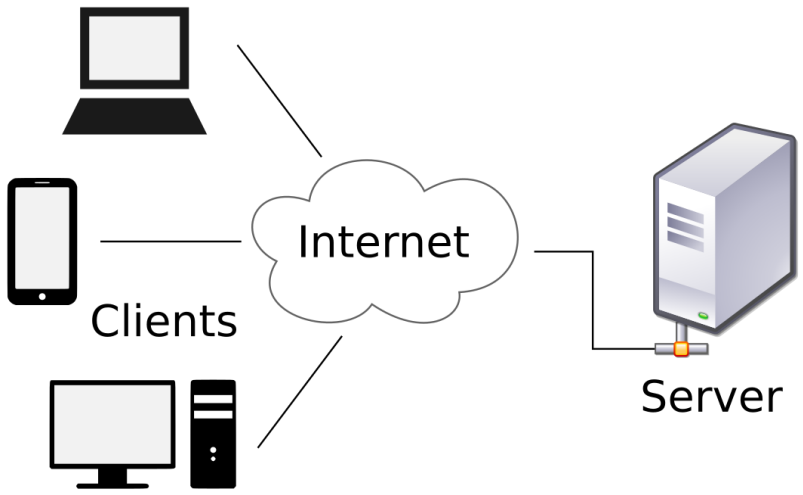


Slika 79. Arhitektura TCP/IP protokola [13]

Sloj pristupa mreži definira rutine za pristup fizičkom mediju, internet sloj definira blok podataka i upravlja usmjeravanjem podataka, Prijenosni sloj osigurava prijenos podataka s kraja na kraj mreže i aplikacijski sloj sadrži aplikacije i procese koji koriste mrežu.

Komunikacija robota sa aplikacijom vrši se preko „socketa“ . Socket je kombinacija IP adrese računala plus „port“ preko kojeg se odvija komunikacija. On djeluje kao početna i završna točka tijekom komunikacije između dva ili više računala.

U našem primjeru Python aplikacija će djelovati kao server na kojeg će se spajati klijenti. Klijenti mogu biti roboti ili PLC-ovi koji će nakon spajanja dobivati podatke od strane servera. Njihova zadaća je da interpretiraju te podatke i ovisno o njihovom sadržaju odrađuju određene zadatke.



Slika 80. Server – Clients konekcija

Prema slici 80. u našem primjeru kao server će djelovati Python aplikacija na PC-u na koju će se spajati dva klijenta (robota). Za takav način spajanja potrebni su nam Ethernet kabel i „Hub“ koji nam omogućuje spajanje više od dva računala (kontrolera).

8.2 Python

Za izradu aplikacije korišten je programski jezik Python. On je jedan od najpopularnijih i najraširenijih programskih jezika posebno popularan zbog svoje „user friendly“ strukture.
Prvi korak je definiranje potrebnih biblioteka, lista koje će se koristiti za spremanje klijenata koji se spoje na server i slanja dnevnog izvješća na email, definiranje parametara za slanje emaila i veličine prozora GUI aplikacije.

```
9   # kreiranje lista za spremanje klijenata
10  all_connections = []
11  all_address = []
12
13  # lista za kreiranje dnevnog izvješća o napravljanom broju aparata
14  list = []
15  list_B = ["Aparat P1 :","Aparat P2 :","Aparat P3 :","Aparat P6 :",
16  list.clear()
17
18  # parametri za slanje dnevnog izvješća putem emaila
19  port_s = 587  # For starttls
20  smtp_server = "smtp-mail.outlook.com"
21  sender_email = "kovadas@hotmail.com"
22  receiver_email = "dkeretic@hotmail.com"
23  password = 'rossonere'
24
25  # veličina GUI prozora
26  HEIGHT = 1500
27  WIDTH = 2000
```

Slika 81. Python aplikacija – 1.dio

Nakon toga, napravljene su funkcije za kreiranje i postavljanje „socketa“ preko kojeg će se vršiti komunikacija. Definirana je IP adresa i port na kojem će se nalaziti server (u ovom slučaju na lokalnoj mreži). Slijedeći korak je funkcija za prihvaćanje zahtjeva za spajanjem od strane klijenata. U ovom primjeru, funkcija je ograničena na dva klijenta s obzirom da se planira spajanje samo dva robota. Nakon spajanja dva klijenta na server, na aplikaciju dolazi informacija da su roboti spojeni i spremni primati podatke. Programski kodovi su vidljivi na slikama 82. i 83.

```
28
29  # funkcija za kreiranje socketa
30  def create_socket():
31      try:
32          global host
33          global port
34          global s
35          host = "127.0.0.10"
36          port = 8000
37          s = socket.socket()
38
39      except socket.error as msg:
40          print("Socket creation error: " + str(msg))
41
42
43  # funkcija postavljanje socketa i čekanja na konekciju klijenata
44  def bind_socket():
45      try:
46          global host
47          global port
48          global s
49          print("Binding the Port: " + str(port))
50
51          s.bind((host, port))
52          s.listen(5)
53
54      except socket.error as msg:
55          print("Socket Binding error" + str(msg) + "\n" + "Retrying...")
56          bind_socket()
```

Slika 82. Python aplikacija – 2.dio

```
44  # funkcija za prihvaćanje klijenata (u ovom primjeru dva robota)
45  def accepting_connections():
46      for c in all_connections:
47          c.close()
48
49      del all_connections[:]
50      del all_address[:]
51
52      while True and len(all_connections)<2:
53          if len(all_connections)<1:
54              conn, address = s.accept()
55
56              all_connections.append(conn)
57              all_address.append(address)
58
59          elif len(all_connections)<2:
60              conn1, address1 = s.accept()
61
62              all_connections.append(conn1)
63              all_address.append(address1)
64
65              response = tk.Label(root, text="SPOJENO", font=("Arial", 15), background="#32CD32")
66              response.place(x=95, y=100, relwidth=0.25)
```

Slika 83. Python aplikacija – 3.dio

Funkcija „disconnected“ služi za prekidanje veze između robota i servera, a funkcija „on_click“ služi za definiranje podataka koji se šalju ovisno o gumbu u aplikaciji koji je pritisnut. Dio koda je vidljiv na slici 84.

```python
# funkcija za odspajanje
def disconnected():
    conn = all_connections[0]
    conn1 = all_connections[1]
    conn.close()
    conn1.close()
    s.close()
    response10 = tk.Label(root, text="ODSPOJENO", font=("Arial", 15), background="#FF0000")
    response10.place(x=95, y=100, relwidth=0.25)

# funkcija za gumbe koji se nalaze u GUI prozoru aplikacije
def on_click(args):
    global aparat
    if args == 1:
        create_socket()
        bind_socket()
        accepting_connections()
        print("Spajanje")
    if args == 2:
        aparat = 'P1'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 3:
        aparat = 'P2'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
```

Slika 84. Python aplikacija – 4.dio

Na slici 85. vidi se na koji način se generiraju podaci koji se putem emaila šalju kao dnevni izvještaj o napravljenom broju komada. Provjeravaju se prvo odgovori od strane klijenata (robotskih kontrolera) i uzima se onaj koji je veći iz razloga što se kod prvog robota ti registri ne koriste (njihove vrijednosti su 0). Ako su registri jednaki to znači da su oba registra 0, te se u obliku stringa ispisuje vrijednost '0' .

```python
    if args == 21:
        global msg_full
        ukupno = 0
        aparat = 'CC'
        for i in range(0,13):
            send_commands()
            time.sleep(0.1)
            conn = all_connections[0]
            conn1 = all_connections[1]
            client_response = str(conn.recv(1024), "utf-8")
            client_response_1 = str(conn1.recv(1024), "utf-8")
            if int(client_response) > int(client_response_1):
                client_response_br = client_response
            elif int(client_response_1) > int(client_response):
                client_response_br = client_response_1
            else:
                client_response_br = ' 0'
            z = int(client_response_br)
            ukupno = ukupno + z
            time.sleep(0.1)
            list.append(list_B[i] + client_response_br)
            i=i+1
            time.sleep(0.1)
        ukupno_str = """\n
UKUPNO: """ + str(ukupno)
        message = """\
Subject: BIJELA SOBA - Dnevni izvjestaj

Broj napravljenih aparata:\n
"""
        list_transpose = ('\n'.join(list))
        msg_full = message + list_transpose + ukupno_str
        print(msg_full)
        send_email()
        i = 0
        list.clear()
```

Slika 85. Python aplikacija – 5.dio

Funkcija „send_commands“ služi za slanje podataka prema klijentima u obliku stringa, „send_email“ služi za slanje emaila, a funkcija „on_close“ služi za potpuno gašenje „socketa“ to jest konekcije i aplikacije (vidi se u prilogu).

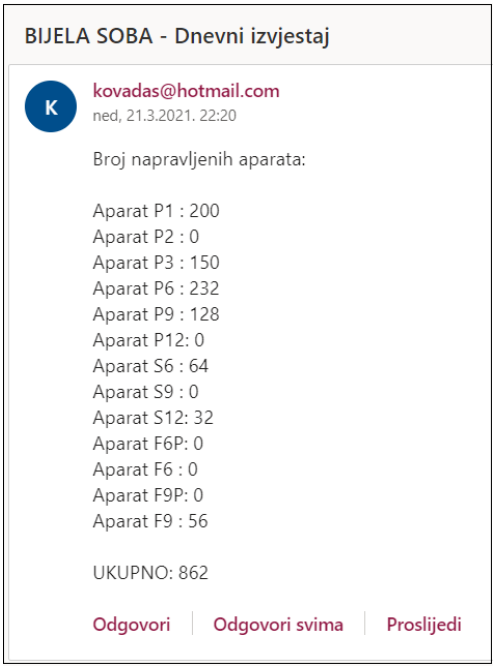
```
242  # funkcija za slanje podataka klijentima
243  def send_commands():
244      conn = all_connections[0]
245      conn1 = all_connections[1]
246      try:
247          if len(str.encode(aparat)) > 0:
248              conn.send(str.encode(aparat))
249              conn1.send(str.encode(aparat))
250      except:
251          disconnected()
252
253  # funkcija za slanje emaila
254  def send_email():
255      context = ssl.create_default_context()
256      with smtplib.SMTP(smtp_server, port_s) as server:
257          server.ehlo()  # Can be omitted
258          server.starttls(context=context)
259          server.ehlo()  # Can be omitted
260          server.login(sender_email, password)
261          server.sendmail(sender_email, receiver_email, msg_full)
```

Slika 86. Python aplikacija – 6.dio

Nakon definiranja svih funkcija napravljeno je GUI korisničko sučelje. Korištena je poznata Python biblioteka Tkinter i naredbe za oznake, tekst, slike, prozore za unos podataka i gumbe. Naredbe su prikazane na slici 87. Svakom gumbu pridodana je funkcija pomoću naredbe „command“. Kompletni programski kod nalazi se u prilogu, a izgled i detaljno funkcioniranje aplikacije objašnjeno je u poglavlju 8.4.

```
207  panel_1 = tk.Label(text="Pastor TVA", font=("Arial", 28), background="#FF4646")
208  panel_1.place(x=663, y=30)
220  img2 = ImageTk.PhotoImage(Image.open("logo2.jpg"))
221  panel_4 = tk.Label(root, image=img2)
222  panel_4.place(x=590, y=400)
247  button3 = tk.Button(root, text="APARAT P1", font=55, bg='#ACA9A9', command=lambda:on_click(2))
248  button3.place(x=100, y=450, relwidth=0.1)
268  entry = tk.Entry(font=("Arial", 20), bg='#ACA9A9')
269  entry.place(x=110, y=660)
```

Slika 87. Python aplikacija – 7.dio



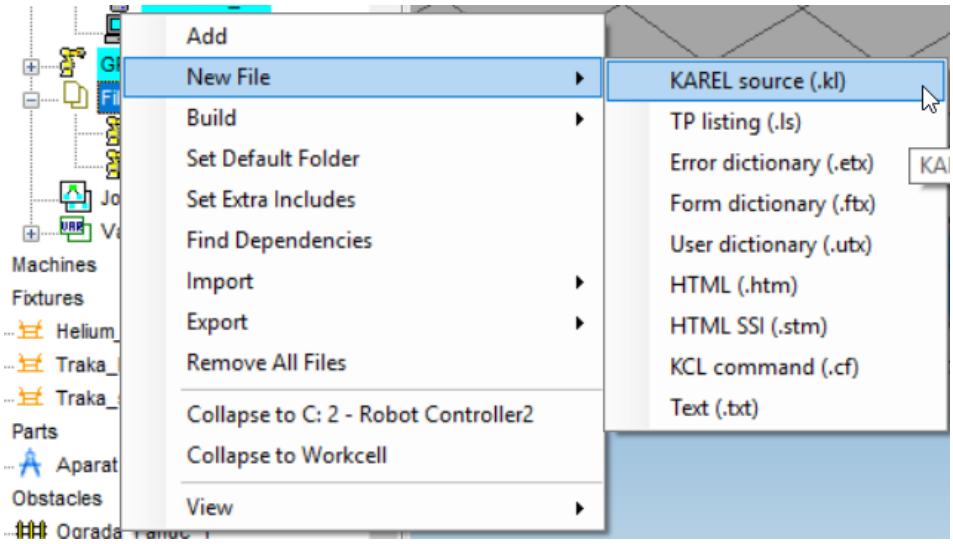
Slika 88. Dnevno izvješće koje se šalje putem maila na kraju radnog dana

8.3 Karel

Karel je niži programski jezik vrlo sličan Pascalu. Na Fanucovim robotima se koristi za programiranje svih onih zadataka koji su na TP kompliciranu ili ih se ne može izvesti.

Ima ugrađene funkcije za dohvaćanje i postavljanje registara, digitalnih signala, koordinatnih sustava, kretanja robota itd. U našem primjeru koristi će ga se kao poveznicu između aplikacije i robotskog kontrolera. Njegov zadatak će biti da omogući spajanje robota na server, dohvati podatke koji se šalju sa servera i odradi zadani zadatak. Konkretno će se koristiti za postavljanje digitalnih izlaza DO, te postavljanje i dohvaćanje registra.

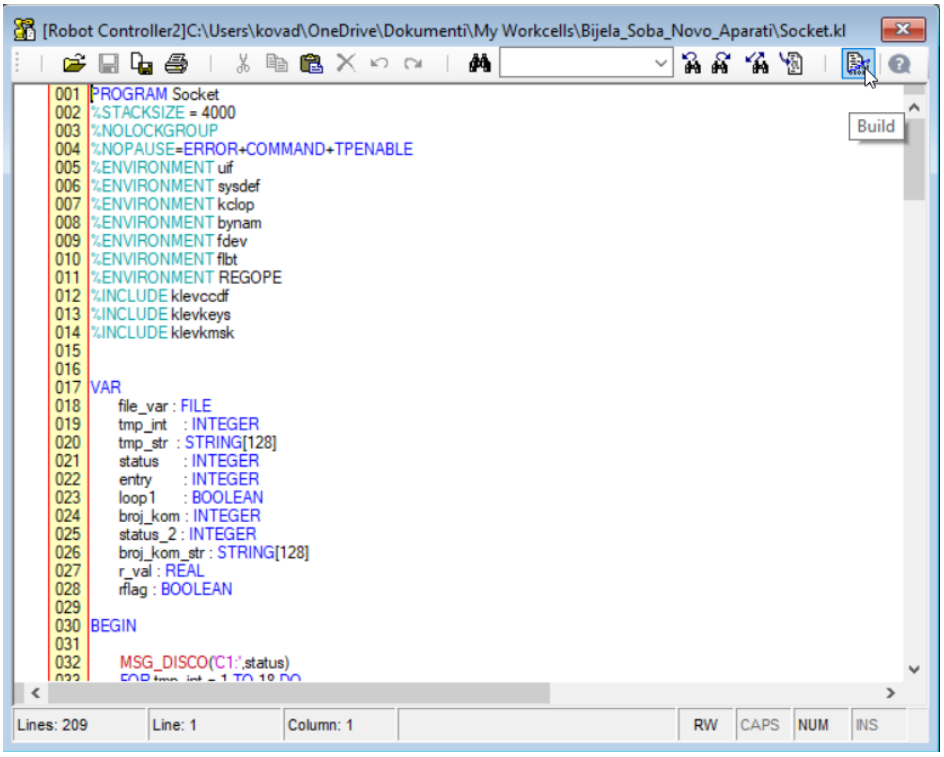
Da bi se Karel mogao koristiti, mora se imati ugrađenu opciju na robotskom kontroleru.



Slika 89. Pisanje novog Karel programa u Roboguideu

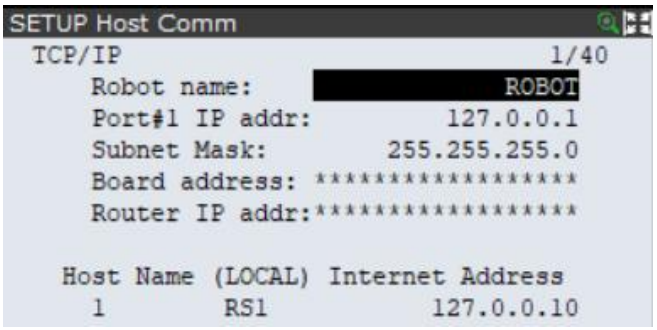
U Roboguide-u se ide pod polje „Files“ , klikne desni klik miša, potom na „New File“ i klikne se na „KAREL source (.kl)“ .

Potom se otvara prozor vidljiv na slici 90. U njega se upisuje Karel program i potom klikne na ikonu „Build“ koja .kl datoteku pretvara u .pc datoteku koja je onda spremna za učitavanje na robota.

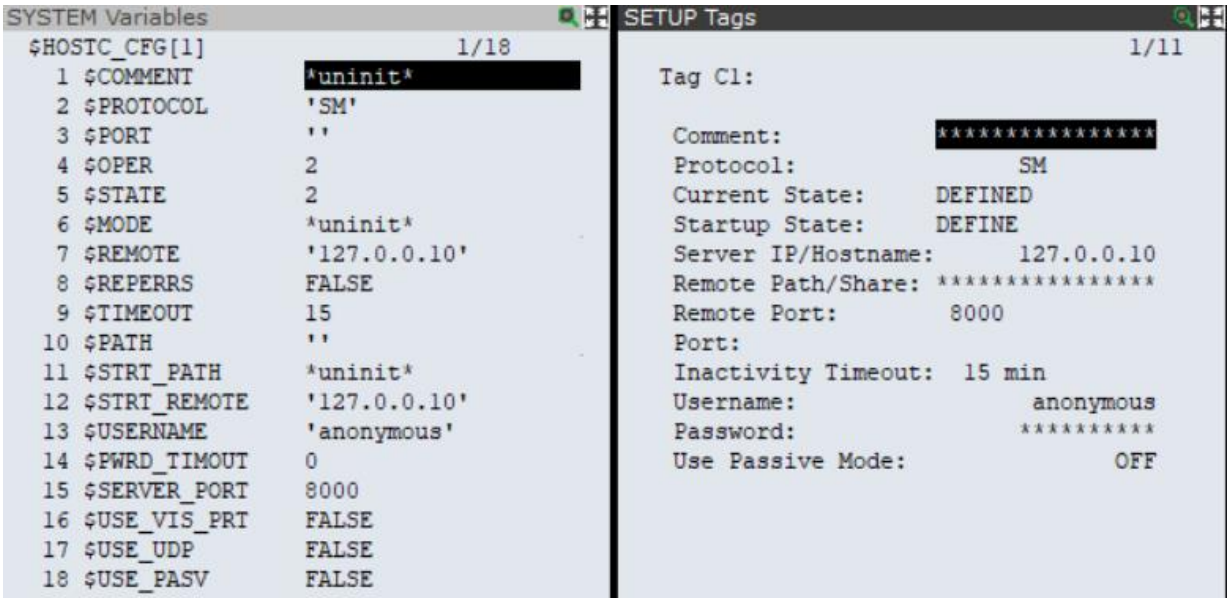


Slika 90. Pisanje novog Karel programa u Roboguideu – 2.dio

Prije samog pisanja programa na robotskom kontroleru preko TP-a treba postaviti određene opcije. Prvo se postavlja IP adresa robota prema slici 91. Prema uputama iz priručnika [14] u poglavlju 11.6.2. postavljeni su tagovi i adresa servera na koji se robot spaja (unutar lokalne mreže za potrebe simulacije).



Slika 91. Postavljanje servera i taga za korištenje unutar Karel programa [14]



Slika 92. Postavljanje servera i taga za korištenje unutar Karel programa [14]

Ovdje je konekcija napravljena lokalno, s obzirom da se spajanje vrši s virtualnim robotom unutar Roboguide-a. Postupak kod pravih robota prikazan je u poglavlju 9.

Nakon postavljanja servera, client taga i konekcije može se napisati Karel program. Dio koda vidljiv je na slici 93. Započinje s definiranjem svih potrebnih datoteka i varijabla koje ćemo koristiti u programu. Potom ugasimo konekciju i resetiramo sve DO i registre koji se koriste u programu u slučaju da su u njima ostali spremljeni podaci od zadnjeg korištenja Nakon toga se

ulazi u beskonačnu petlju, otvara se datoteka u koju se zapisuju podaci i pokušava se spojiti. Ukoliko je spajanje bilo uspješno, čekaju se podaci sa servera. Pri primanju podataka, interpretira ih se i ovisno o njihovom sadržaju postavlja registar R12 i uključuju određeni DO. Kompletni Karel kod vidljiv je u prilogu.

```
001 PROGRAM Socket_V2
002 %STACKSIZE = 4000
003 %NOLOCKGROUP
004 %NOPAUSE=ERROR+COMMAND+TPENABLE
005 %ENVIRONMENT uif
006 %ENVIRONMENT sysdef
007 %ENVIRONMENT kclop
008 %ENVIRONMENT bynam
009 %ENVIRONMENT fdev
010 %ENVIRONMENT flbt
011 %ENVIRONMENT REGOPE
012 %INCLUDE klevccdf
013 %INCLUDE klevkeys
014 %INCLUDE klevkmsk
015
016
017 VAR
018     file_var : FILE
019     tmp_int  : INTEGER
020     tmp_str  : STRING[128]
021     status   : INTEGER
022     entry    : INTEGER
023     loop1    : BOOLEAN
024     broj_kom : INTEGER
025     status_2 : INTEGER
026     number : INTEGER
027     broj_kom_str : STRING[128]
028     r_val : REAL
029     rflag : BOOLEAN
030     reg_rs : INTEGER
031
032
033 BEGIN
034     reg_rs = 11
035     number = 13
036     MSG_DISCO('C1:',status)
037     DOUT[1] = OFF
038     DOUT[2] = OFF
039     DOUT[3] = OFF
040     DOUT[9] = OFF
041     SET_REAL_REG(12,0, status)
042
043
044     label::
045
046     DELAY 100
047
048     SET_FILE_ATR(file_var, ATR_IA)
049     SET_VAR(entry, '*SYSTEM*','$HOSTC_CFG[1].$SERVER_PORT',8000,status)
050     WRITE(' VAR status = ',status,CR)
051     MSG_CONNECT('C1:',status)
052     WRITE(' Connect status = ',status,CR)
053     loop1 = TRUE
054     IF status = 0 THEN
055         WHILE loop1 = TRUE DO
056             DELAY 100
057             WRITE('Opening file...',CR)
058             OPEN FILE file_var('rw','C1:')
059             status = IO_STATUS(file_var)
060             IF status = 0 THEN
061                 WRITE('Waiting to read from server...')
062                 DELAY 100
063                 READ file_var(tmp_str::2)
064                 WRITE('Read: ',tmp_str::2,CR)
065                 DELAY 100
066
067                 IF tmp_str = 'P1' THEN
068                     SET_REAL_REG(12,1, status)
069                     DOUT[3] = OFF
070                     DOUT[1] = ON
071                     DELAY 50
072                 ENDIF
073
074                 IF tmp_str = 'P2' THEN
075                     SET_REAL_REG(12,2, status)
076                     DOUT[3] = OFF
077                     DOUT[1] = ON
078                     DELAY 50
079                 ENDIF
080
081                 IF tmp_str = 'P3' THEN
082                     SET_REAL_REG(12,3, status)
083                     DOUT[3] = OFF
084                     DOUT[1] = ON
```

Slika 93. Prvi dio Karel programa

8.4 Aplikacija

Python aplikacija za kontrolu linije prikazana je na slici 94.

Slika 94. Aplikacija za kontrolu linije za završnu montažu

Aplikacija funkcionira na sljedeći način. Operater po početku radnog vremena pali računalo na kojem se nalazi aplikacija i postavlja robote u automatski način rada. Roboti će biti tako postavljeni da će se na njima automatski pokrenuti „MAIN“ program na TP-u koji će potom paralelno pokrenuti i Karel program za komunikaciju s aplikacijom.

Nakon pokretanja, operater klikne na gumb „CONNECT“ i čeka da se svi roboti spoje. Pri uspješnom spajanju u polju 2 dobiva zelenu oznaku za tekstom „SPOJENO“. Iz polja gumba 4 ili 10 bira jedan od aparata koje će pustiti na liniju. U polju 12, iznad gumba „START“ pojavljuje se oznaka ovisno o tome koji gumb je pritisnut. Klikom na gumb „START“ pokreću se programi na robotima i linija je spremna za rad. U bilo kojem trenutku klikom na gumb „CHECK“ može se provjeriti koliko je komada stavljeno na paletu. Broj komada prikazat će se u polju 6. U slučaju promjene kupca ili vrste aparata operater može, ukoliko mu to odgovara, kliknuti na gumb „RESET“ . Tada se brojač aparata postavlja na 0, te ponovo kreće brojati ispočetka. U slučaju promjene programa, to jest vrste aparata koji se radi potrebno je sljedeće: kliknuti na gumb „QUIT(PREKID)“ (dolazi obavijest operateru unutar polja 1 da su se roboti odspojili), ponovo kliknuti na gumb „CONNECT (SPAJANJE)“ i čekati da se roboti povežu. Nakon povezivanja ponoviti postupak opisan ranije. Gumb „TODAY DATA“ služi za generiranje podataka o dnevnom broju napravljenih aparata (po vrsti i ukupno), te slanja emaila

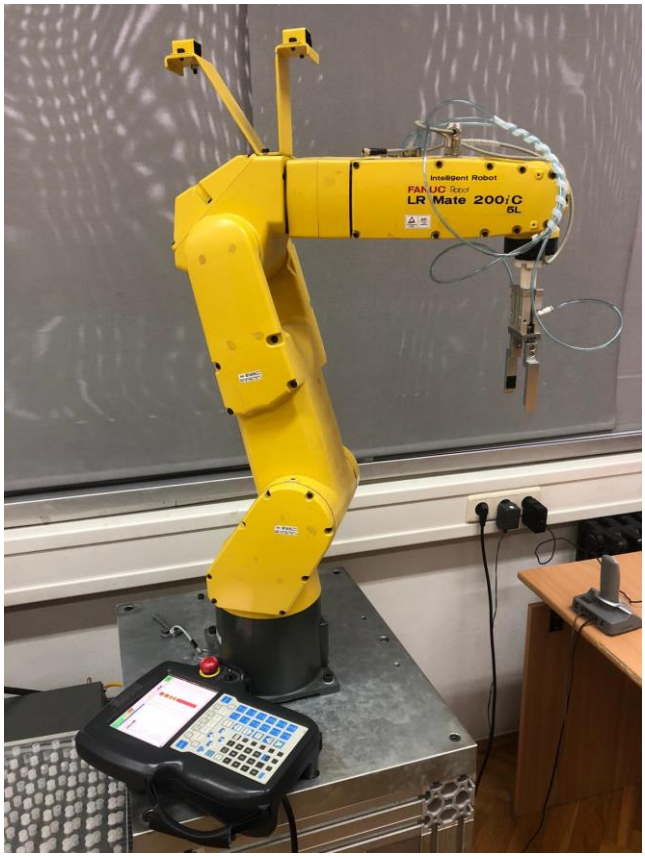
s izvješćem. Klikom na gumb „TURN OFF“ programi na robotu se potpuno gase i dnevno brojanje se resetira. U tom slučaju da bi se ponovo započelo s radom potrebno je resetirati robote. Gumb „CONTINUE“ se koristi iz sigurnosnih razloga u slučaju da je robot napunio obje palete. On tada staje i čeka da se palete zamjene. Nakon zamjene punih paleta sa praznima, klikom na gumb „CONTINUE“ robotu se daje indikacija da može nastaviti sa izvršavanjem programa.

Tablica 5. Dijelovi aplikacije za upravljanje linijom za završnu monažu

	DIJELOVI APLIKACIJE
1.	Gumb „TODAY DATA“ koji generira dnevno izvješće o broju napravljenih komada koje se potom šalje emailom.
2.	Oznaka za indikaciju da li su svi klijenti (robot, PLC) spojeni ili ne.
3.	Gumb „CONNECT (SPAJANJE)“ služi za uspostavljanje konekcije sa klijentima (robot, PLC).
4.	Polje sa gumbima P aparata. Klikom na jedan od gumba odabire se program za taj aparat koji je snimljen na robotskom kontroleru.
5.	Gumb „CHECK“ služi za provjeru registra R11 koji daje informaciju o broju aparata koji su stavljeni na paletu.
6.	Polje „BROJ KOMADA“ u kojem se pritiskom na gumb „CHECK“ pojavljuje broj spremljen u registru R11.
7.	Gumb „RESET“ služi za resetiranje brojača pri promjeni kupca ili programa (vrste aparata).
8.	Gumb „TURN OFF“ služi za gašenje konekcije i svih programa koji se izvode na robotima.
9.	Gumb „QUIT“ služi za gašenje konekcije između servera i klijenta.
10.	Polje sa gumbima S i F aparata. Klikom na jedan od gumba odabire se program za taj aparat koji je snimljen na robotskom kontroleru.
11.	Gumb „CONTINUE“ služi da se robotu da signal da su pune palete zamijenjene praznima i da može nastaviti sa programom.
12.	Oznaka iznad gumba „START“ koja označava koji gumb je trenutno kliknut.
13.	Gumb „START“ pokreće izvođenje odabranog programa.

9. TESTIRANJE

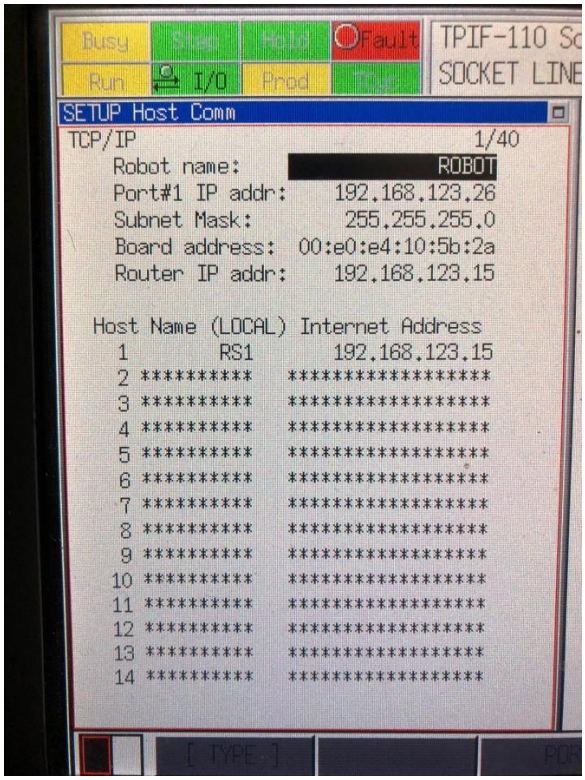
Testiranje aplikacije i uspostavljanja konekcije sa stvarnim robotom izvršeno je u Laboratoriju za Projektiranje izradbenih i montažnih sustava na Fakultetu strojarstva i brodogradnje u Zagrebu. Iskorišten je Fanucov robot LR Mate 200iC 5L koji se nalazi u laboratoriju. Riječ je o robotu drugačijih karakteristika sa starijom verzijom kontrolera u odnosu na odabrane robote. Ipak, za potrebe testiranja aplikacije njegovo ponašanje je identično kao i izabranih robota. Kontroler je R30iA Mate. Noviji roboti dolaze sa kontrolerom R30iB Plus koji je novija i naprednija verzija od R30iA Mate. S obzirom na različite karakteristike i hvataljku, na njemu nije moguće testirati prave programe koji su napravljeni u sklopu ovog projekta. Ideja je napraviti pokazne programe, uspostaviti konekciju sa robotom, pokrenuti jedan od njih i snimati ponašanje robota pri interakciji sa aplikacijom i pri promjeni programa.



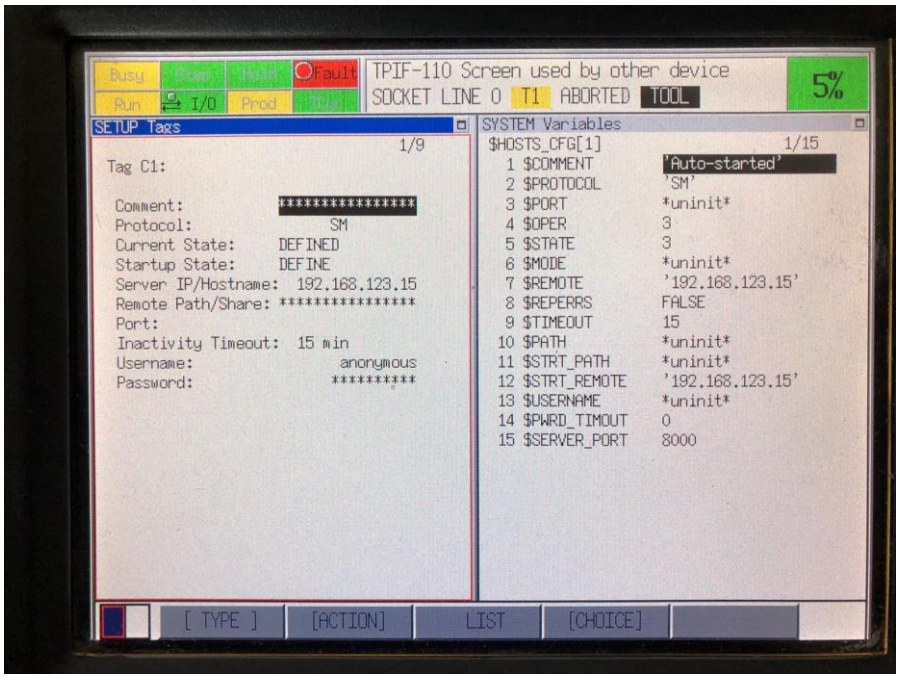
Slika 95. Fanuc LR Mate 200iC 5L

Slika 96. Fanuc R30iA Mate kontroler i privjesak za učenje (TP – Teach Pendant)

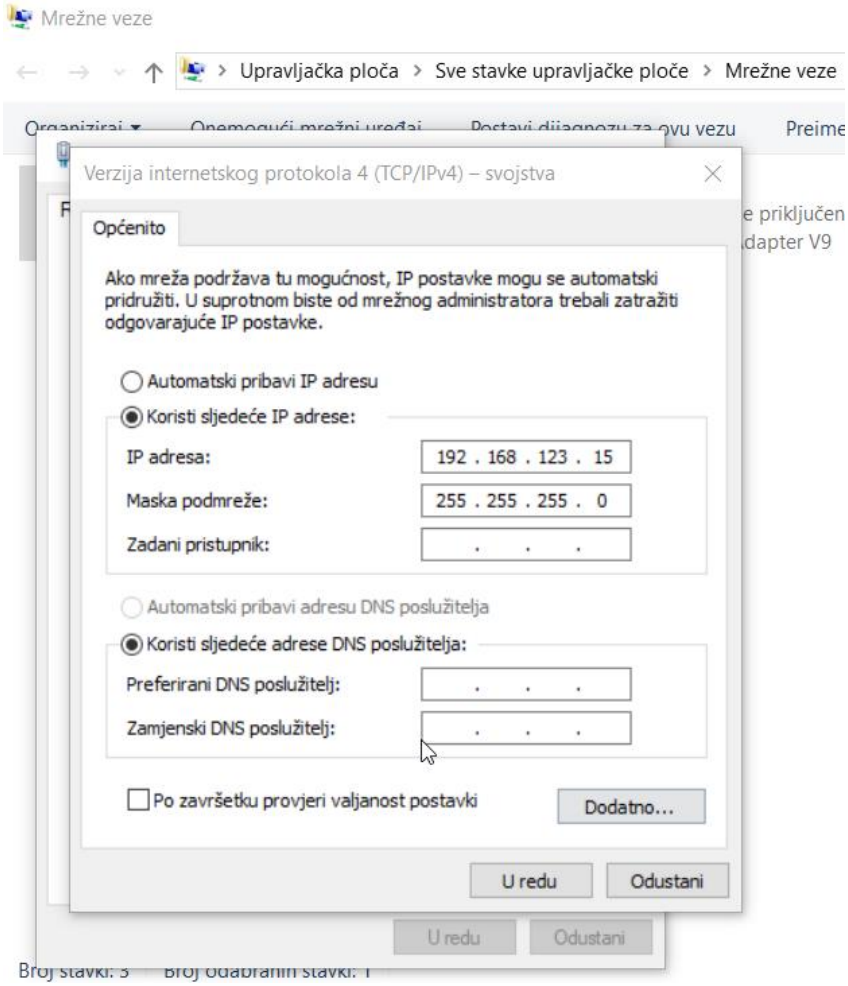
Za uspostavu konekcije potreban je Ethernet kabel. U slučaju spajanja na više robota ili klijenata potreban je i Hub. Nakon spajanja potrebno je postaviti parametre na robotu i računalu.



Slika 97. Postavljanje IP adrese robota i servera



Slika 98. Postavljanje client taga i servera na stvarnom robotu



Slika 99. Postavljanje IP adrese računala/servera

Osim prikazanih postavki potrebno je i onemogućiti „Windows defender“ ili neki slični zaštitnički program.

Nakon uspostavljanja konekcije, na robot je učitan Karel program iz poglavlja 8.3. Umjesto stvarnih P1 i P6 programa, napravljeni su pokazni koji samo simuliraju jednostavne robotske kretnje.

Slika 100. Postavljanje robota u automatski način rada – AUTO mode

Robot se treba postaviti u „AUTO“ mode rada, sa programom „MAIN“ postavljenim kao instrukcijom za izvođenje. Na slici 100. vidi se postupak stavljanja robota u automatski način rada.

Nakon što su postavljeni svi parametri i napravljeni svi programi, napravljeno je testiranje koje je snimano mobitelom. Rezultati su se pokazali identičnima kao i u softverskom okruženju čime je potvrđeno dobro funkcioniranje aplikacije.

10. OCR – PREPOZNAVANJE SERIJSKOG BROJA

Na liniju za završnu montažu planira se implementirati i kamera koja će okidati slike serijskog broja na spremniku vatrogasnog aparata. Ideja je da se razvija OCR algoritam koji će prepoznati brojeve koji će se nalaziti na slici i na temelju njih generirati barcode. Potom će se printati naljepnica s barcode-om koja će se lijepiti na spremnike. Osim naljepnica, paralelno će se prema serijskom broju u postojeću aplikaciju za servise vatrogasnih aparata slati podaci o datumu proizvodnje i tipu vatrogasnog aparata. To će omogućiti da kada aparat dođe na redovno servisiranje u ovlašteni servis (1 put godišnje), serviser će skenirati barcode i unutar aplikacije će mu se otvoriti stranica s podacima o točnom tom vatrogasnom aparatu. Nakon što obavi servis u aplikaciju će unijeti datum servisa i postupke koji su rađeni na aparatu. Na taj način kada aparat idući put dođe na servis, jednostavnim skeniranjem barcode-a, serviseru će se prikazati njegova povijest (tip, datum proizvodnje, datumi i opisi svih servisa).

Time će se implementirati digitalno praćenje životnog vijeka svakog Pastorovog vatrogasnog aparata.

Slika 101. Primjer serijskog broja na spremniku vatrogasnog aparata

Slika 102. Primjer serijskog broja na spremniku vatrogasnog aparata - 2

Na slikama 101. i 102. vide se primjeri serijskog broja na spremniku P2 aparata. Brojevi se sastoje od 6 znamenaka koje je potrebno prepoznati i prema njima generirati barcode.

Za testiranje same ideje i algoritma koristit će se neuronske mreže u Pythonu. Ideja je skupiti podatke okidanjem više stotina slika serijskih brojeva na raznim spremnicima. Nakon skupljanja, slike je potrebno obraditi, filtrirati i segmentirati po znamenkama. Potom se znamenke prosljeđuju neuronskoj mreži i započinje se s procesom učenja.

Primjer procesa obrade, filtriranja i segmentacije prikazat će se na slici 102.

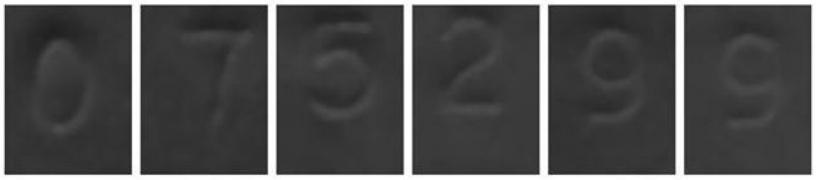
Nakon okidanja početne slike, potrebno je odrezati dio slike koji nas interesira.

Slika 103. Izrezani dio s početne slike koji nas interesira za daljnju obradu

Nakon toga izrezani dio prikazan na slici 103. prebacuje se u sliku sivih tonova (Grayscale sliku) i na nju se primjenjuju Gaussian i Median filteri koji služe za uglađivanje slike i reduciranje smetnji / šumova.

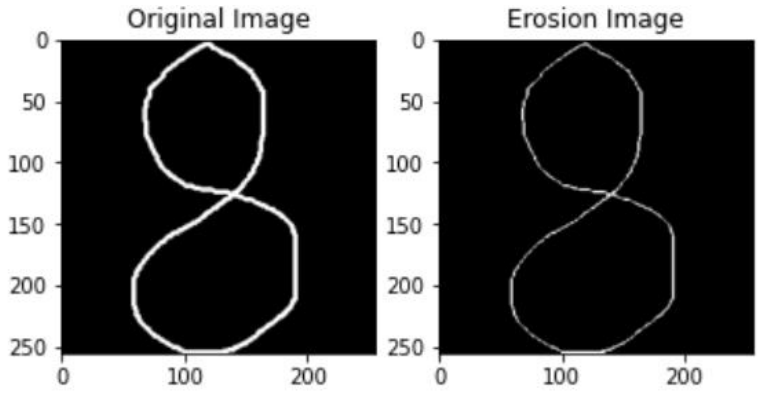
Slika 104. Prebacivanje u grayscale i filtriranje Gaussom i Medianom

Nakon toga vrši se segmentacija slike to jest odvajanje pojedinih znamenaka zasebno. Postupak je prikazan na slici 105.

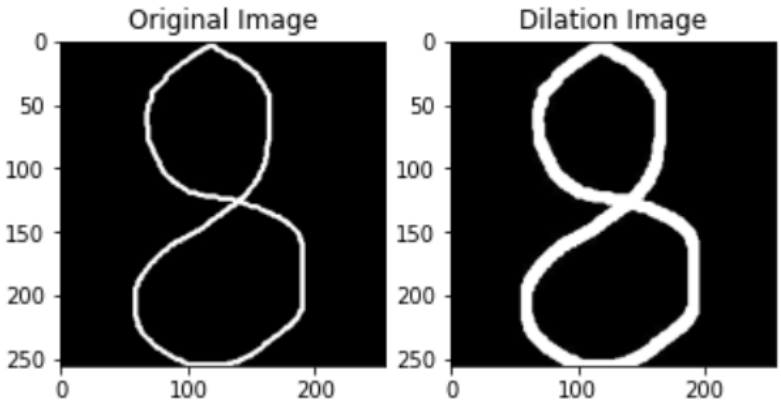


Slika 105. Segmentacija znamenaka

Sljedeća je na redu morfološka obrada slike. Koristit će se takozvano otvaranje (OPENING) i nakon toga dodatno erodiranje (EROSION). Otvaranje se sastoji od erodiranja koje je potom popraćeno s dilatacijom. Erodiranje je reduciranje piksela blizu granice uz uvjet da se uvijek zadrži baza objekta. Dilatacija je operacija koja je inverzna erodiranju. Kod nje se povećava broj istih piksela oko granice baze objekta. Primjeri erodiranja i dilatacije su slikovito prikazani na slikama 106. i 107. [15]



Slika 106. Primjer morfološke operacije erozije [15]

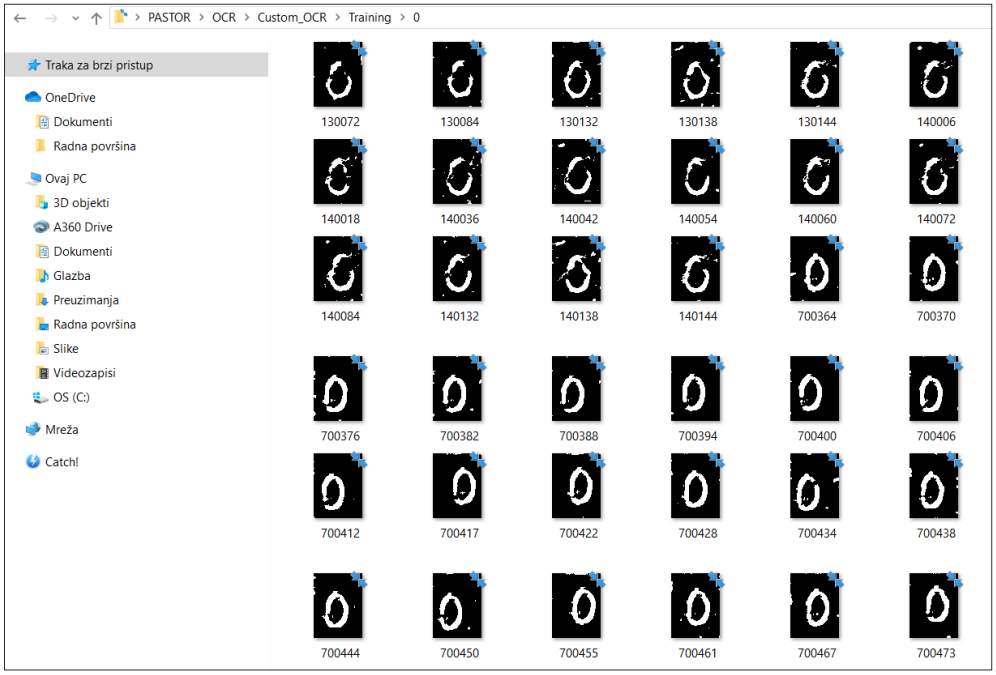


Slika 107. Primjer morfološke operacije dilatacije [15]

Primjer morfološke obrade slike 105. prikazan je na slici 108.

Slika 108. Segmentacija znamenaka

Kod za obradu, filtriranje i segmentiranje slike nalazi se u prilogu. Segmentirane znamenke potom odvajamo i spremamo u zasebne foldere. Npr. sve segmentirane nule stavljamo u folder 0 prikazan na slici 109. Isti postupak ponavljamo za sve znamenke od 0-9. Osim foldera za trening dio podataka odvajamo i u folder za validaciju.



Slika 109. Folder za trening s znamenkama nula

Te podatke ćemo potom proslijediti OCR algoritmu baziranom na neuronskim mrežama. U Pythonu koristimo biblioteke TensorFlow i Keras. TensorFlow je open source biblioteka za razne varijante strojnog učenja, dok je Keras biblioteka za neuronske mreže.

U ovom slučaju korišten je uobičajeni sekvencijalni model. On omogućava slaganje modela neuronske mreže sloj po sloj. Mreža i svaki sloj imaju samo po jedan ulaz i izlaz.

Na slici 110. prikazan je Python kod za sekvencijalnu neuronsku mrežu.

```
# kreiranje keras modela za duboko učenje (neuronske mreže)
# sekvencijalni model omogućuje dodavanje slojeva jedan po jedan
model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(60,80,3)),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
                                    tf.keras.layers.MaxPool2D(2, 2),

                                    tf.keras.layers.Flatten(),

                                    tf.keras.layers.Dense(512,activation='relu'),
                                    tf.keras.layers.Dense(9,activation='sigmoid')
                                    ])
```

Slika 110. Model sekvencijalne neuronske mreže

Mreža se sastoji od 2D konvolucijskih slojeva (Conv2D) koji stvaraju kernel matricu koja u kombinaciji s ulazima slojeva pomaže u stvaranju tenzora izlaza. Potrebno je definirati broj

filtera, veličinu kernel matrice, aktivacijsku funkciju i veličinu ulaza. MaxPool2D služi za smanjenje prostornih dimenzija izlaza, a funkcije Flatten i Dense služe da na izlazu dobijemo rješenje u obliku vektora tako da matricu pretvorimo u 1D polje.

Nakon definiranja modela izvršava se kompajliranje. Unutar compile funkcije potrebno je definirati funkciju gubitka. Gubitak se izračunava kako bi se dobili gradijenti u odnosu na težine modela i prema tome ažurirali prolazeći kroz mrežu unazad. Mreža se ažurira nakon svake iteracije sve dok ažuriranja modela ne dovedu do poboljšanja u željenoj mjernoj vrijednosti.

U ovom slučaju odabrana je funkcija 'sparse_categorical_crossentropy' koja se preporuča kada imamo 2 ili više klasa koje su označene kao cijelo brojne vrijednosti (integeri) što je ovdje slučaj.

Osim toga, moramo definirati i željenu metodu optimizacije. Ona nam služi da sa svakom iteracijom smanjujemo gubitke i podešavamo težine tako da se sve više približavamo željenom izlazu. U ovom slučaju korišten je SGD (Stochastic Gradient Descent**)** koji je najpoznatija metoda optimizacije koja se temelji na smanjivanju gradijenta.

Nakon toga zadajemo uvjete treninga, parametre koje želimo pratiti, trajanje i uvjet završetka treninga, te prosljeđujemo podatke i započinjemo s treniranjem mreže.

```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=tf.keras.optimizers.SGD(learning_rate=0.001),
              metrics=['accuracy'])

tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                                 min_delta=0,
                                 patience=0,
                                 verbose=0,
                                 mode='auto',
                                 baseline=None,
                                 restore_best_weights=False)

callback = [EarlyStopping(monitor='val_loss', patience=50),
            ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

model_fit = model.fit(train_dataset,
                      steps_per_epoch=20,
                      epochs=200,
                      validation_data=validation_dataset,
                      callbacks=callback)
```

Slika 111. Kompajliranje i zadavanje uvjeta treninga

```
Epoch 99/200
20/20 [======] - 1s 32ms/step - loss: 6.5453e-04 - accuracy: 1.0000 - val_loss: 0.1387 - val_accuracy: 0.9522
Epoch 100/200
20/20 [======] - 1s 32ms/step - loss: 0.0300 - accuracy: 0.9833 - val_loss: 0.3699 - val_accuracy: 0.9043
Epoch 101/200
20/20 [======] - 1s 31ms/step - loss: 0.0058 - accuracy: 1.0000 - val_loss: 0.2045 - val_accuracy: 0.9413
Epoch 102/200
20/20 [======] - 1s 32ms/step - loss: 6.6154e-04 - accuracy: 1.0000 - val_loss: 0.2005 - val_accuracy: 0.9391
Epoch 103/200
20/20 [======] - 1s 34ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.1906 - val_accuracy: 0.9391
Epoch 104/200
20/20 [======] - 1s 38ms/step - loss: 7.8505e-04 - accuracy: 1.0000 - val_loss: 0.1788 - val_accuracy: 0.9391
```

Slika 112. Rezultati mreže nakon treninga

Na slici 112. vidimo rezultate treninga. U ovom slučaju uzorak testiranja je bio oko 100-200 fotografija po znamenki. Što je veći uzorak za treniranje mreži to su i rezultati učenja mreže bolji. Sa slike možemo vidjeti da je mreža završila trening i prije zadanih 200 iteracija. To se dogodilo zato što smo stavili funkciju EarlyStopping koja prekida treniranje ako se pokaže da određeni broj iteracija (u ovom slučaju 50) mreža ne postiže značajniji napredak u smanjivanju gubitka.

Vrijednosti loss i accuracy pokazuju rezultate na setu za treniranje. S obzirom da mreža uspijeva prepoznati s 100% sve slike koje su dane u set za treniranje može se reći da je dizajnirani model mreže dovoljno dobar. Vrijednosti val_loss i val_accuracy pokazuju rezultat mreže na setu za validaciju. Tu je gubitak nešto veći i točnost nije 100% nego nešto više od 93%. S obzirom da je ovo testna mreža i da su fotografije snimane u raznim uvjetima, rezultati su zadovoljavajući. Točnost se može još više povećati kada bi se sve fotografije snimile u točno jednakim uvjetima i kada bi se mrežu treniralo na puno većem setu fotografija. Cijeli kod za treniranje neuronske mreže nalazi se u prilogu.

Na kraju je mreža i isprobana na jednom od primjera. Napravljen je algoritam koji najprije izrezuje sa slike dio koji nam treba, potom vrši obradu i filtraciju slike. Zatim segmentira sliku po znamenkama i te znamenke prosljeđuje OCR algoritmu (modelu neuronske mreže koji smo prethodno trenirali). Cijeli kod za testiranje neuronske mreže nalazi se u prilogu.

Primjer početne fotografije i rezultata prikazani su na slikama 113. i 114.

Slika 113. Početna fotografija za testiranje algoritma

```
1.jpeg
Ovo je broj: 0
2.jpeg
Ovo je broj: 1
3.jpeg
Ovo je broj: 6
4.jpeg
Ovo je broj: 5
5.jpeg
Ovo je broj: 5
6.jpeg
Ovo je broj: 1
Serijski broj je: 016551
```

Slika 114. Rezultati testiranja OCR modela

Daljnja razrada uključuje nabavu kamere i RaspberryPi uređaja pomoću kojih će se izraditi testna konstrukcija za skupljanje velike količine podataka (slika) na kojima će se izvršavati pravo treniranje. Nakon dobivanja dovoljno dobrih rezultata (točnost viša od 99%) planira se kupiti printer za naljepnice, profesionalni kontroler i kamera, te izvršiti implementacija cijelog sustava u proizvodni pogon.

11. OKVIRNA FINANCIJSKA ANALIZA

Na osnovi ponuda koje su prikupljene od raznih dobavljača i koje su pronađene na internetu, napravljena je okvirna financijska analiza. Cilj je vidjeti okvirnu investiciju i njezinu isplativost. U tablici 6. navedene su komponente linije i okvirni troškovi.

Tablica 6. Okvirne cijene pojedinih dijelova linije

	Komponenta	**Kom**	**Cijena**
1.	Stroj za ispitivanje propusnosti Helijem	1	€107,000.00
2.	Fanuc M710iC – 70	2	€100,000.00
3.	Pokretne trake	2	€30,000.00
4.	Stroj za stezanje ventila (Kina)	1	€30,000.00
5.	Siemens PLC + dodaci	1	€5,000.00
6.	Zaštitna ograda	1	€5,000.00
7.	Roboguide - Softver	1	€3,200.00
8.	Izrada limova i metalnih dijelova	1	€3,000.00
9.	Schmalz komponente za hvataljku 1	1	€2,600.00
10.	Schmalz komponente za hvataljku 2	1	€2,100.00
11.	Senzorika	1	€2,000.00
12.	Kamera + Raspberry Pi za OCR testiranje	1	€1,000.00
13.	Ostali nepredviđeni troškovi	1	€15,000.00
	UKUPNO		€305,900.00

S obzirom da bi projekt bio sufinanciran uz potporu fondova Europske Unije, možemo računati da će okvirna investicija Pastora biti otprilike oko 50% ukupne investicije.
Prema tome ispada da je investicija Pastora oko 153,000.00 €. Neki troškovi u tablici su vjerojatno i malo pretjerani, ali stavljeni su takvi s obzirom da se želi doći do nekog maksimuma pa će se onda sve ispod toga smatrati prihvatljivim.

Što se tiče povećanja troškova, porasti će cijena aktivacijskog plina zbog upotrebe helija.
Koristit će se smjesa dušika i 5% Helija. Proračun je rađen na aparatu P6.
Ukupan volumen P6 aparata je:

$$V_{P6} = \ 6{,}65\ l \tag{2.1}$$

Od čega aktivacijski plin popunjava oko 40-50% spremnika, dakle volumen dušika možemo uzeti da je:

$$V_{N2} = \ 3{,}0\ l \tag{2.2}$$

Gustoća dušika, molarna masa, tlak i temperatura punjenja su:

$$\varphi_{N2} = \ 1{,}251\ \frac{kg}{m^3}$$

$$M_{N2} = \ 28\ \frac{g}{mol}$$

$$p = \ 15\ bar$$

$$\vartheta = \ 20\ °C$$

$$T = \ 293{,}15\ K \tag{2.3}$$

Jednadžba stanja idealnog plina glasi:

$$p \cdot V_{N2} = \ n_{N2} \cdot R \cdot T \tag{2.4}$$

Gdje je n_{N2} – množina tvari izražena u molovima i R – univerzalna plinska konstanta koja iznosi $R = \ 8{,}314\ \frac{J}{mol\ K}$. Uvrštavanjem vrijednosti (2.3) u izraz (2.4) dobivamo množinu dušika u spremniku:

$$n_{N2} = \ \frac{p \cdot V_{N2}}{R \cdot T} = \frac{15 \cdot 10^5 \cdot 3 \cdot 10^{-3}}{8{,}314 \cdot 293{,}15}$$

$$n_{N2} = \ 1{,}846\ mol \tag{2.5}$$

Nakon toga preko molarne mase i množine izračunavamo masu dušika koja se nalazi u spremniku:

$$m_{N2} = \ M_{N2} \cdot n_{N2} = 28 \cdot 10^{-3} \cdot 1{,}846$$

$$m_{N2} = \ 0{,}0517\ kg \tag{2.6}$$

Prema informacijama od dobavljača, cijena po kilogramu dušika iznosi $C_{N2} = \ 6{,}542\ \frac{kn}{kg}$, pa je onda cijena dušika po aparatu (P6) jednaka:

$$C_{N2,P6} = \ 0{,}338\ \frac{kn}{aparat} \tag{2.7}$$

Ako uzmemo da se godišnje izradi 100 000 komada P6 aparata (ukupno se izradi 150 000

aparata ali veliki udio su S i neki od F aparata koji se ne pune aktivacijskim plinom) slijedi da je cijena dušika na godišnjoj razini:

$$C_{N2,ukupno} = \ 100000 \cdot 0{,}338 = 33800 \ \frac{kn}{god} \tag{2.8}$$

S obzirom, da bi promjenom načina testiranja, u aktivacijsku smjesu (dušik + helij) trebalo dodati 5% volumnog udjela helija, slijedi prilagođeni proračun:

$$y_{N2} = \ 0{,}95$$
$$y_{He} = \ 0{,}05$$
$$V_{N2+He} = \ 3{,}0 \ l \tag{2.9}$$

Gustoća i molarna masa helija su:

$$\varphi_{He} = \ 0{,}1786 \ \frac{kg}{m^3}$$
$$M_{He} = \ 4 \ \frac{g}{mol} \tag{2.10}$$

Iz formule za volumni udio slijede volumeni dušika i helija u aktivacijskoj smjesi:

$$y_{He} = \ \frac{V_{He}}{V_{P6}}, y_{N2} = \ \frac{V_{N2}}{V_{P6}}$$
$$V_{He} = \ 1{,}5 \cdot 10^{-4} \ m^3 \ , \ V_{N2} = \ 2{,}85 \cdot 10^{-3} \ m^3 \tag{2.11}$$

Iz čega se uvrštavanjem u formulu (2.5) dobivaju množine helija i dušika u spremniku:

$$n_{He} = \ 0{,}0923 \ mol$$
$$n_{N2} = \ 1{,}754 \ mol \tag{2.12}$$

Nakon toga, uvrštavanjem u formulu (2.6) dobivamo mase helija i dušika:

$$m_{He} = \ 3{,}692 \cdot 10^{-4} \ kg$$
$$m_{N2} = \ 0{,}049 \ kg \tag{2.13}$$

Prema informacijama od dobavljača, cijena po kilogramu helija iznosi $C_{N2} = \ 983{,}02 \ \frac{kn}{kg}$, pa je onda cijena helija i dušika po aparatu:

$$C_{N2,P6} = \ 0{,}3206 \ \frac{kn}{aparat}$$

$$C_{He,P6} = \ 0{,}363 \ \frac{kn}{aparat} \tag{2.14}$$

Dakle ukupan trošak aktivacijskog plina po aparatu jest:

$$C_{uk,P6} = \ 0{,}68356 \ \frac{kn}{aparat} \tag{2.15}$$

Što u primjeru sa 100 000 aparata na godišnjoj razini iznosi:

$$C_{uk} = \ 68356 \ \frac{kn}{aparat} \tag{2.16}$$

Možemo zaključiti da promjenom načina testiranja i nabavom aktivacijskog plina koji je mješavina helija i dušika dobivamo poskupljenje sa otprilike 30 000 na 70 000 kn godišnje.

Ako sada izračunamo uštedu samo na osnovi toga za koliko smo smanjili broj radnika na liniji dobivamo slijedeće. Broj radnika na liniji smanjen je za 2-3 radnika (uzimamo najgori slučaj). Davanja poslodavca za njih na godišnjoj razini sa svim doplatama i porezima može se uzeti da je otprilike 200 – 300,000.00 kn. S obzirom da imamo povećanje troškova aktivacijskog plina, uzet ćemo najgori slučaj - 200,000.00 kn. Uz investiciju od 153,000.00 € što je otprilike 1,150,000.00 kn. Dobivamo povrat investicije u roku od maksimalno 5-6 godina (također uzeto s rezervom).

Također, bitno je napomenuti da u ovom okvirnom proračunu nisu uzeti u obzir faktori povećanja efikasnosti, subvencioniranje plaća ljudi koji će raditi na projektu od strane EU, povećanja produktivnosti, smanjenja fizičkog napora, modernizacije linije, ulazak u tehnologiju 21. stoljeća itd. Od navedenih faktora, nisu svi vezani direktno uz financije, ali su navedeni kao samo neke od prednosti implementacije ove linije.
Kao zadnja napomena u ovoj kratkoj analizi mora se napomenuti da se za uspješnu implementaciju linije mora analizirati kvaliteta i cijena kutija za pakiranje.
Postojeće kutije ne zadovoljavaju niti dizajnom, niti kvalitetom. Tako da je to jedan od problema sa kojima bi se, u slučaju implementacije, još trebalo pozabaviti i naći potencijalnu alternativu (nove kutije, novi dobavljač, učvršćivanje postojećih kutija itd.).

ZAKLJUČAK

U sklopu diplomskog rada, napravljen je koncept i razrada automatske linije za završnu montažu vatrogasnih aparata za tvrtku Pastor TVA. Odrađena je izrada modela, simulacija i programa. Odabrani su roboti i strojevi koji će se koristiti. Osmišljeno je poboljšanje procesa koje donosi reduciranje potrebnih radnika na procesu završne montaže sa 7 na 4, ubrzanje ciklusa za 15-20%, modernizaciju testiranja propusnosti, digitalizaciju upravljanja i praćenja linije i godišnju uštedu od oko 200 000 kn.

Primijenjena su znanja stečena tijekom studiranja na realni industrijski projekt. U projektu je korišten širok spektar vještina naučenih u prethodnih 5 godina. Rad se sastoji od izrade nacrta, modeliranja u CAD alatima, programiranja, izrade proračuna, testiranja i analize isplativosti. Tijekom izrade je naučeno jako puno novih stvari koje su potom uklopljene u projekt. Kroz cijeli proces se pojavilo puno problema, čijim rješavanjem se dobiva dojam koliko je teško, dugotrajno i naporno razvijati i uklapati nove stvari u postojeću proizvodnju. Potencijalna dugoročna vizija za dalje, uključuje implementaciju sličnih automatskih linija u cijeli proizvodni pogon. Izbacivanje papira i olovke provođenjem potpune digitalizacije praćenja proizvodnje. Automatizacija nije stvar budućnosti, nego sadašnjosti i trendovi pokazuju da će kompanije koje ne uspiju implementirati digitalizaciju i robotizaciju u narednih nekoliko godina, vrlo vjerojatno nestati s tržišta u bliskoj budućnosti.

Ova tema diplomskog rada i primjena u stvarnom industrijskog okruženju omogućila mi je da naučim jako puno o problemima i načinima implementacije teoretskog znanja sa fakulteta u industriji. Planiram se dalje posvetiti i razvijati u ovom području jer smatram da je zanimljivo i aktualno. Uz sve veći rast novih tehnologija, interneta i robotike smatram da će u idućih 20-30 godina većina toga biti automatizirano i robotizirano.

LITERATURA

[1] Web stranica tvrtke Pastor TVA, https://www.pastor.hr/ 29.1.2021.

[2] Stroj za stezanje ventila, tvrtka Yuyao Chaocheng Machinery Manufacturing Co, web stranica, https://yycc.en.alibaba.com/ 29.1.2021.

[3] Industrijski robot Fanuc M710iC-50, tvrtka Fanuc, službena web stranica, https://www.fanuc.eu/it/en/robots/robot-filter-page/m-710-series/m-710ic-50 1.2.2021.

[4] Lekcija o testiranju propusnosti helijem, TQC online lekcija, https://www.tqc.co.uk/our-services/leak-testing/helium/guide-to-helium-leak-testing/ 1.2.2021.

[5] Testiranje propusnosti helijem, online course, https://www.youtube.com/watch?v=i7JOCsEugGY&ab_channel=INFICON 1.2.2021.

[6] Stroj za ispitivanje propusnosti helijem, tvrtka Fritz – EMDE, službena web stranica, https://www.fritz-emde.com/en/products/helium-leak-detection-unit/ 1.2.2021.

[7] Proračun vakuumskih hvataljki, tvrtka Schmalz, službena web stranica, https://www.schmalz.com/en/vacuum-knowledge/the-vacuum-system-and-its-components/system-design-calculation-example/ 1.2.2021.

[8] Predavanja iz kolegija Projektiranje automatskih montažnih sustava, gradivo hvataljke, portal e-učenje, Fakultet strojarstva i brodogradnje u Zagrebu

[9] Vakuumske komponente, tvrtka Schmalz, službena web stranica, https://www.schmalz.com/en/vacuum-technology-for-automation/vacuum-components 1.2.2021.

[10] RoboDK upute, online priručnik, https://robodk.com/doc/en/Basic-Guide.html 4.2.2021.

[11] RoboDK forum, web stranica, https://robodk.com/forum/ 4.2.2021.

[12] Skripta „Računalne mreže“, Prirodoslovno matematički fakultet, Split https://mapmf.pmfst.unist.hr/~lada/rm/rm-pog7.pdf 6.3.2021.

[13] Web članak o računalnim mrežama, https://pcchip.hr/helpdesk/obitelj-internet-protokola-skupina-tcpip-protokola 6.3.2021.

[14] Fanuc Karel, referentni priručnik, http://therobotguyllc.com/wp-content/uploads/2015/05/KAREL-Programming-Guide.pdf 7.3.2021.

[15] Web članak o morfološkoj obradi slike,

https://medium.com/analytics-vidhya/morphological-transformations-of-images-using-opencv-image-processing-part-2-f64b14af2a38 27.8.2021.

PRILOG

a) Python program za simuliranje senzora
b) Python program za simuliranje prve trake
c) Python program za simuliranje pneumatskog cilindra
d) Python program za simuliranje prve robotske stanice
e) Python program za simuliranje druge trake
f) Python program za simuliranje stroja za ispitivanje propusnosti
g) Python program za simuliranje druge robotske stanice
h) Python program za simuliranje hvatanja različitih veličina kutija
i) Python program za simuliranje hvatanja različitih veličina aparata
j) Python aplikacija za kontrolu linije
k) Karel program
l) TP programi – prva robotska stanica
m) TP programi – druga robotska stanica
n) Nacrti ploča
o) Program za obradu fotografija
p) Program za modeliranje i treniranje neuronske mreže za OCR
q) Program za testiranje OCR modela

---

a)

```
from robolink import *
from robodk import *
RDK = Robolink()
RDK.Render(False)

SENSOR_NAME = 'Sensor SICK WL4S_2'
SENSOR_VARIABLE = 'SENSOR2'

PART_KEYWORD = 'Ventil '

RECHECK_PERIOD = 0.001

SENSOR = RDK.Item(SENSOR_NAME, ITEM_TYPE_OBJECT)

all_objects = RDK.ItemList(ITEM_TYPE_OBJECT, False)
part_objects = []
for obj in all_object:
    if obj.Name().count(PART_KEYWORD) > 0:
part_objects.append(obj)

            detected_status = -1
            while True:
                detected = 0
                for obj in part_objects:
                    # Provjerava prisutnost objekta tako da provjerava da li dolazi do kolizije
```

```
            if SENSOR.Collision(obj) > 0:
                detected = 1
                break
        if detected_status != detected:
            detected_status = detected
            RDK.setParam(SENSOR_VARIABLE, detected_status)
            print('Object detected status --> %i' % detected_status)
        pause(RECHECK_PERIOD)
```

b)

```
--------------------------------------------------------------------------------------------
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)

distance = 608.33
SENSOR_VARIABLE_3 = 'SENSOR3'

traka_2            = RDK.Item('Traka_2', ITEM_TYPE_ROBOT)
frame_conv_2       = RDK.Item('Traka_2', ITEM_TYPE_FRAME)
frame_conv_moving_2 = RDK.Item('Pomicni_koord_dva', ITEM_TYPE_FRAME)
spremnik           = RDK.Item('Spremnik', ITEM_TYPE_OBJECT)

# gather targets
target_home_conv = RDK.Item('Home_conv_2', ITEM_TYPE_TARGET)
target_move_conv = RDK.Item('Move_conv_2', ITEM_TYPE_TARGET)

traka_2.setSpeed(250)

n = -1
i = 0

while i > n:
if RDK.RunMode() == RUNMODE_SIMULATE:
if RDK.getParam(SENSOR_VARIABLE_3) == 0:
distance_new = i*distance
traka_2.setPoseFrame(frame_conv_2)
target_move_conv_new = target_move_conv.Pose()*transl(distance_new,0,0)
traka_2.MoveJ(target_move_conv_new)
pause(10)
i=i+1
else:
pause(0.001)
--------------------------------------------------------------------------------------------
```

c)

```
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)
distance = 100
SENSOR_VARIABLE = 'SENSOR'

Trigger_mech       = RDK.Item('Trigger_mech', ITEM_TYPE_ROBOT)
frame_conv_trigg   = RDK.Item('Trigger_frame_main', ITEM_TYPE_FRAME)
fanuc_base         = RDK.Item('Fanuc M-710iC/45M Base',ITEM_TYPE_FRAME)
tool               = RDK.Item('Tool 2', ITEM_TYPE_TOOL)
Move_trigg         = RDK.Item('Move_trigg', ITEM_TYPE_TARGET)
Home_trigg         = RDK.Item('Home_trigg', ITEM_TYPE_TARGET)

def TCP_On(toolitem):
   toolitem.AttachClosest()
```

```
    toolitem.RDK().RunMessage('Set air valve on')
    toolitem.RDK().RunProgram('TCP_On()');

def TCP_Off(toolitem, itemleave=0):
    toolitem.DetachAll(itemleave)
    toolitem.RDK().RunMessage('Set air valve off')
    toolitem.RDK().RunProgram('TCP_Off()');

Trigger_mech.setSpeed(350)
Trigger_mech.setPoseFrame(frame_conv_trigg)
i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        while RDK.getParam(SENSOR_VARIABLE) == 0:
            pause(0.001)
        if RDK.getParam(SENSOR_VARIABLE) == 1:
            pause(0.5)
            TCP_On(tool)
            Trigger_mech.MoveJ(Move_trigg)
            TCP_Off(tool,fanuc_base)
            Trigger_mech.MoveJ(Home_trigg)
            pause(5)
```

---

d)

```
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)
APPROACH = 200
SENSOR_VARIABLE_3 = 'SENSOR3'
SENSOR_VARIABLE_4 = 'SENSOR4'
SENSOR_VARIABLE_5 = 'SENSOR5'
SENSOR_VARIABLE_9 = 'SENSOR9'
SENSOR_VARIABLE_10 = 'SENSOR10'

speedJ = 120
speedL = 450
accJ = 800
accL = 800

robot             = RDK.Item('Fanuc M-710iC/45M', ITEM_TYPE_ROBOT)
tool              = RDK.Item('Fanuc_gripper', ITEM_TYPE_TOOL)
frame_fanuc       = RDK.Item('Fanuc M-710iC/45M', ITEM_TYPE_FRAME)
frame_conv_3      = RDK.Item('Pomicni_koord_tri', ITEM_TYPE_FRAME)

target_home = RDK.Item('Home_Fanuc', ITEM_TYPE_TARGET)
target_pick = RDK.Item('Fanuc_pick', ITEM_TYPE_TARGET)
target_pass = RDK.Item('Fanuc_pass', ITEM_TYPE_TARGET)

target_pick_app = target_pick.Pose()*transl(0,0,-APPROACH)
target_pick_app_2 = target_pick.Pose()*transl(0,0.70*APPROACH,-2.2*APPROACH)
target_pick_up = target_pick.Pose()*transl(0,1.35*APPROACH,0)
target_pick_app_up = target_pick.Pose()*transl(0,1.35*APPROACH,-APPROACH)

target_place_1 = RDK.Item('Fanuc_place', ITEM_TYPE_TARGET)
target_place_2 = RDK.Item('Fanuc_place_2', ITEM_TYPE_TARGET)
target_conv_3_app = RDK.Item('Fanuc_conv_3_app', ITEM_TYPE_TARGET)
target_conv_3_place = RDK.Item('Fanuc_conv_3_place', ITEM_TYPE_TARGET)

target_app_3 = target_conv_3_place.Pose()*transl(0,APPROACH,-APPROACH)
target_up_3 = target_conv_3_place.Pose()*transl(0,APPROACH,0)
target_back_3 = target_conv_3_place.Pose()*transl(0,0,-APPROACH)
```

```
robot.setPoseFrame(frame_fanuc)
robot.setSpeed(-1,speedJ)
robot.setSpeed(speedL)
robot.setAcceleration(accL)
robot.setAccelerationJoints(accJ)

def TCP_On(toolitem):
    toolitem.AttachClosest()
    pause(0.1)
    toolitem.RDK().RunMessage('Set air valve on')
    toolitem.RDK().RunProgram('TCP_On()');
def TCP_Off(toolitem, itemleave=0):
    toolitem.DetachAll(itemleave)
    pause(0.1)
    toolitem.RDK().RunMessage('Set air valve off')
    toolitem.RDK().RunProgram('TCP_Off()');

def Pick_conv(robot):
    pause(0.5)
    robot.MoveJ(target_pick_app)
    robot.MoveJ(target_pick)
    TCP_On(tool)
    robot.MoveL(target_pick_up)
    robot.MoveL(target_pick_app_up)
    robot.MoveJ(target_home);

def Place_chamber(robot,target):
    target_app = target.Pose()*transl(0,50,-1.5*APPROACH)
    target_up = target.Pose()*transl(0,50,0)
    target_back = target.Pose()*transl(0,0,-1.5*APPROACH)
    robot.MoveL(target_app)
    robot.MoveL(target_up)
    robot.MoveL(target)
    TCP_Off(tool)
    robot.MoveL(target_back);

def Take_chamber(robot,target):
    target_app = target.Pose()*transl(0,50,-1.5*APPROACH)
    target_up = target.Pose()*transl(0,50,0)
    target_back = target.Pose()*transl(0,0,-1.5*APPROACH)
    robot.MoveL(target_back)
    robot.MoveL(target)
    TCP_On(tool)
    robot.MoveL(target_up)
    robot.MoveL(target_app);

def Place_conv(robot):
    robot.MoveJ(target_pass)
    robot.MoveL(target_app_3)
    robot.MoveL(target_up_3)
    robot.MoveL(target_conv_3_place)
    TCP_Off(tool,frame_conv_3)
    robot.MoveL(target_back_3);


i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        while RDK.getParam(SENSOR_VARIABLE_3) == 0:
            pause(0.001)
        if RDK.getParam(SENSOR_VARIABLE_3) == 1:
            if i < 2:
                i = i + 1
                pause(3.0)
            robot.MoveJ(target_home)
```

```
        Pick_conv(robot)
        if RDK.getParam(SENSOR_VARIABLE_4) == 0:
            print("Load chamber 1")
            Place_chamber(robot,target_place_1)
            while  RDK.getParam(SENSOR_VARIABLE_5) == 1 : #& (if test_2_finished==1)
                if RDK.getParam(SENSOR_VARIABLE_10) == 1:
                    print("Unload chamber 2 and put it on conv_3")
                    Take_chamber(robot,target_place_2)
                    Place_conv(robot)

        elif RDK.getParam(SENSOR_VARIABLE_5) == 0:
            print("Load chamber 2")
            Place_chamber(robot,target_place_2)

            while  RDK.getParam(SENSOR_VARIABLE_4) == 1 : #& (if test_1_finished==1)"""
                if RDK.getParam(SENSOR_VARIABLE_9) == 1:
                    print("Unload chamber 1 and put it on conv_3")
                    Take_chamber(robot,target_place_1)
                    Place_conv(robot)
```

---

e)

```
from robolink import *
from robodk import *

RDK = Robolink()

SENSOR_VARIABLE_6 = 'SENSOR6' # station variable

traka_3            = RDK.Item('Traka_3', ITEM_TYPE_ROBOT)
frame_conv_3       = RDK.Item('Traka_3', ITEM_TYPE_FRAME)
frame_conv_moving_3 = RDK.Item('Pomicni_koord_tri', ITEM_TYPE_FRAME)

target_home_conv = RDK.Item('Home_conv_3', ITEM_TYPE_TARGET)
target_move_conv = RDK.Item('Move_conv_3', ITEM_TYPE_TARGET)

traka_3.setSpeed(-1,45)
i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        if RDK.getParam(SENSOR_VARIABLE_6) == 1:
            pause(0.1)
            traka_3.MoveJ(target_move_conv)
        else:
            pause(0.001)
```

---

f)

```
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)
SENSOR_VARIABLE_4 = 'SENSOR4'
SENSOR_VARIABLE_6 = 'SENSOR6'
SENSOR_VARIABLE_9 = 'SENSOR9'

Komora_1           = RDK.Item('Komora_1', ITEM_TYPE_ROBOT)
Komora_1_1         = RDK.Item('Komora_1', ITEM_TYPE_FRAME)

target_home_1 = RDK.Item('Home_komora_1', ITEM_TYPE_TARGET)
target_move_1 = RDK.Item('Move_komora_1', ITEM_TYPE_TARGET)
```

```
Komora_1.setSpeed(250)
Komora_1.setPoseFrame(Komora_1_1)

i = 0
n = -1

while i>n:
    if RDK.RunMode() == RUNMODE_SIMULATE:
        if RDK.getParam(SENSOR_VARIABLE_4) == 1:
            Komora_1.MoveJ(target_move_1)
            pause(25)
            Komora_1.MoveJ(target_home_1)
            while RDK.getParam(SENSOR_VARIABLE_9) == 1:
                pause(0.001)
                if RDK.getParam(SENSOR_VARIABLE_6) == 1:
                    break
        else:
            pause(0.001)
```

---

g)

```
from robolink import *
from robodk import *

# Use RoboDK API as RL
RDK = Robolink()
RDK.Render(False)
APPROACH = 300
SENSOR_VARIABLE_8 = 'SENSOR8' # station variable
distance = 200
speedJ = 120
speedL = 350
height = 200

robot             = RDK.Item('Fanuc M-710iC/45M_2', ITEM_TYPE_ROBOT)
tool              = RDK.Item('Vakuum_hvataljka', ITEM_TYPE_TOOL)
frame_robot       = RDK.Item('Fanuc M-710iC/45M Base_2', ITEM_TYPE_FRAME)
frame_paleta      = RDK.Item('Frame_paleta', ITEM_TYPE_FRAME)

target_home = RDK.Item('Home', ITEM_TYPE_TARGET)
target_pick = RDK.Item('Pick', ITEM_TYPE_TARGET)
target_upp_pick = target_pick.Pose()*transl(0,0,-height)
target_place = RDK.Item('Place',ITEM_TYPE_TARGET)
target_place_2 = RDK.Item('Place_2',ITEM_TYPE_TARGET)

def TCP_On(toolitem):
    pause(0.25)
    toolitem.AttachClosest()
    toolitem.RDK().RunMessage('Set air valve on')
    toolitem.RDK().RunProgram('TCP_On()');

def TCP_Off(toolitem, itemleave=0):
    pause(0.25)
    toolitem.DetachAll(itemleave)
    toolitem.RDK().RunMessage('Set air valve off')
    toolitem.RDK().RunProgram('TCP_Off()');

def Pick_box(robot):
    robot.setPoseFrame(frame_robot)
    robot.MoveJ(target_upp_pick)
    robot.MoveL(target_pick)
    pause(0.2)
    TCP_On(tool)
    robot.MoveL(target_upp_pick)
    robot.MoveJ(target_home);
```

```
def Place_box(robot,target):
   robot.setPoseFrame(frame_paleta)
   robot.MoveJ(target_app_place)
   robot.MoveL(target)
   pause(0.2)
   TCP_Off(tool,frame_paleta)
   robot.MoveL(target_app_place)
   robot.MoveJ(target_home);

robot.setSpeed(-1,speedJ)
robot.setSpeed(speedL)

i = 0
n = -1
m = 0
z = 0
j = 0
x = -1
y = 0

while i>n:
   if RDK.RunMode() == RUNMODE_SIMULATE:
      while RDK.getParam(SENSOR_VARIABLE_8) == 0:
         pause(0.001)

      if RDK.getParam(SENSOR_VARIABLE_8) == 1:
         x = x + 1
         if x == 4:
            x = 0
            j = j + 1
         if j == 2:
            j = 0
            target_place_hor = target_place_2.Pose()*transl(z*900,0,-m*155)
            target_app_place = target_place_2.Pose()*transl(z*900,0,-m*155-height)
            Pick_box(robot)
            Place_box(robot,target_place_hor)
            z = z + 1
            y = y + 1
         if z == 2:
            z = 0
            m = m + 1
         if m == 8:
            m = 0
            robot.MoveJ(target_home)

         if y != 1:
            target_place_1 = target_place.Pose()*transl(-j*520,-x*200-z*900,-m*155)
            target_app_place = target_place.Pose()*transl(-j*520,-x*200-z*900,-m*155-height)
            Pick_box(robot)
            Place_box(robot,target_place_1)
         if y == 1:
            y = 0
            x = x – 1
```

---

h)

```
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)

APPROACH = 300
SENSOR_VARIABLE_8 = 'SENSOR8'
distance = 200
distance_hor = 630
```

```
speedJ = 120
speedL = 350
height = 200

robot             = RDK.Item('Fanuc M-710iC/45M_2', ITEM_TYPE_ROBOT)
tool              = RDK.Item('Vakuum_hvataljka', ITEM_TYPE_TOOL)
frame_robot       = RDK.Item('Fanuc M-710iC/45M Base_2', ITEM_TYPE_FRAME)
frame_paleta      = RDK.Item('Frame_paleta', ITEM_TYPE_FRAME)

target_home = RDK.Item('Home', ITEM_TYPE_TARGET)
target_pick_1 = RDK.Item('Pick_1', ITEM_TYPE_TARGET)
target_pick_2 = RDK.Item('Pick_2', ITEM_TYPE_TARGET)
target_pick_3 = RDK.Item('Pick_3', ITEM_TYPE_TARGET)
target_pick_4 = RDK.Item('Pick_4', ITEM_TYPE_TARGET)
target_pick_5 = RDK.Item('Pick_5', ITEM_TYPE_TARGET)
target_pick_6 = RDK.Item('Pick_6', ITEM_TYPE_TARGET)
target_place = RDK.Item('Place', ITEM_TYPE_TARGET)
target_place_velika = RDK.Item('Place_2', ITEM_TYPE_TARGET)

def TCP_On(toolitem):
   toolitem.AttachClosest()
   toolitem.RDK().RunMessage('Set air valve on')
   toolitem.RDK().RunProgram('TCP_On()');

def TCP_Off(toolitem, itemleave=0):
   #toolitem.DetachClosest(itemleave)
   toolitem.DetachAll(itemleave)
   toolitem.RDK().RunMessage('Set air valve off')
   toolitem.RDK().RunProgram('TCP_Off()');

def Pick_box(robot,target):
   robot.setPoseFrame(frame_robot)
   target_upp = target.Pose()*transl(0,0,-height)
   robot.MoveJ(target_upp)
   robot.MoveL(target)
   TCP_On(tool)
   pause(0.2)
   robot.MoveL(target_upp)
   robot.MoveJ(target_home);

def Place_box(robot,target,target_app):
   robot.setPoseFrame(frame_paleta)
   robot.MoveJ(target_app)
   robot.MoveL(target)
   TCP_Off(tool,frame_paleta)
   pause(0.2)
   robot.MoveL(target_app)
   robot.MoveJ(target_home);

robot.setSpeed(-1,speedJ)
robot.setSpeed(speedL)

i = 0
n = -1
m = 0

while i>n:
   if RDK.RunMode() == RUNMODE_SIMULATE:
      while RDK.getParam(SENSOR_VARIABLE_8) == 0:
         pause(0.001)
      if RDK.getParam(SENSOR_VARIABLE_8) == 1:
         i = i + 1
         if i == 1:
            Pick_box(robot,target_pick_1)
            target_app = target_place.Pose()*transl(0,0,-height)
            Place_box(robot,target_place,target_app)
         elif i == 2:
```

```
        Pick_box(robot,target_pick_2)
        target_place_2 = target_place.Pose()*transl(25,-115,-30)
        target_app = target_place.Pose()*transl(25,-115,-30-height)
        Place_box(robot,target_place_2,target_app)
    elif i == 3:
        Pick_box(robot,target_pick_3)
        target_place_3 = target_place.Pose()*transl(85,-237,-24)
        target_app = target_place.Pose()*transl(85,-237,-24-height)
        Place_box(robot,target_place_3,target_app)
    elif i == 4:
        Pick_box(robot,target_pick_4)
        target_app = target_place_velika.Pose()*transl(0,0,-height)
        Place_box(robot,target_place_velika,target_app)
    elif i == 5:
        Pick_box(robot,target_pick_5)
        target_place_5 = target_place_velika.Pose()*transl(196,60,-15)
        target_app = target_place_velika.Pose()*transl(196,60,-15-height)
        Place_box(robot,target_place_5,target_app)
    elif i == 6:
        Pick_box(robot,target_pick_6)
        target_place_6 = target_place_velika.Pose()*transl(405,85,-30)
        target_app = target_place_velika.Pose()*transl(405,85,-30-height)
        Place_box(robot,target_place_6,target_app)
```

---

i)

```
from robolink import *
from robodk import *

RDK = Robolink()
RDK.Render(False)
APPROACH = 200
SENSOR_VARIABLE_3 = 'SENSOR3'
SENSOR_VARIABLE_4 = 'SENSOR4'
SENSOR_VARIABLE_5 = 'SENSOR5'
SENSOR_VARIABLE_9 = 'SENSOR9'
SENSOR_VARIABLE_10 = 'SENSOR10'
SENSOR_VARIABLE_12 = 'SENSOR10'

speedJ = 120
speedL = 450
accJ = 800
accL = 800

robot             = RDK.Item('Fanuc M-710iC/45M', ITEM_TYPE_ROBOT)
tool              = RDK.Item('Fanuc_gripper', ITEM_TYPE_TOOL)
frame_fanuc       = RDK.Item('Fanuc M-710iC/45M', ITEM_TYPE_FRAME)
frame_conv_3      = RDK.Item('Pomicni_koord_tri', ITEM_TYPE_FRAME)

target_home = RDK.Item('Home_Fanuc', ITEM_TYPE_TARGET)
target_pick_P1 = RDK.Item('Fanuc_pick_P1',ITEM_TYPE_TARGET)
target_pick_P2 = RDK.Item('Fanuc_pick_P2',ITEM_TYPE_TARGET)
target_pick_P3 = RDK.Item('Fanuc_pick_P3',ITEM_TYPE_TARGET)
target_pick_P6 = RDK.Item('Fanuc_pick_P6', ITEM_TYPE_TARGET)
target_pick_P9 = RDK.Item('Fanuc_pick_P9',ITEM_TYPE_TARGET)
target_pick_P12 = RDK.Item('Fanuc_pick_P12',ITEM_TYPE_TARGET)
target_pass = RDK.Item('Fanuc_pass', ITEM_TYPE_TARGET)
target_place_1 = RDK.Item('Fanuc_place', ITEM_TYPE_TARGET)
target_place_2 = RDK.Item('Fanuc_place_2', ITEM_TYPE_TARGET)

target_conv_3_app = RDK.Item('Fanuc_conv_3_app', ITEM_TYPE_TARGET)
target_conv_3_place = RDK.Item('Fanuc_conv_3_place', ITEM_TYPE_TARGET)

target_app_3 = target_conv_3_place.Pose()*transl(0,APPROACH,-APPROACH)
target_up_3 = target_conv_3_place.Pose()*transl(0,APPROACH,0)
target_back_3 = target_conv_3_place.Pose()*transl(0,0,-APPROACH)
```

```
robot.setPoseFrame(frame_fanuc)
robot.setSpeed(-1,speedJ)
robot.setSpeed(speedL)
robot.setAcceleration(accL)
robot.setAccelerationJoints(accJ)

def TCP_On(toolitem):
    pause(0.5)
    toolitem.AttachClosest()
    pause(0.1)
    toolitem.RDK().RunMessage('Set air valve on')
    toolitem.RDK().RunProgram('TCP_On()');

def TCP_Off(toolitem, itemleave=0):
    #toolitem.DetachClosest(itemleave)
    pause(0.5)
    toolitem.DetachAll(itemleave)
    pause(0.1)
    toolitem.RDK().RunMessage('Set air valve off')
    toolitem.RDK().RunProgram('TCP_Off()');

def Pick_conv(robot,target):
    pause(0.5)
    target_pick_app = target.Pose()*transl(0,0,-APPROACH)
    target_pick_app_2 = target.Pose()*transl(0,0.70*APPROACH,-2.2*APPROACH)
    target_pick_up = target.Pose()*transl(0,1.35*APPROACH,0)
    target_pick_app_up = target.Pose()*transl(0,1.35*APPROACH,-APPROACH)
    robot.MoveJ(target_pick_app)
    robot.MoveJ(target)
    TCP_On(tool)
    robot.MoveL(target_pick_up)
    robot.MoveL(target_pick_app_up)
    robot.MoveJ(target_home);

def Place_chamber(robot,target,z):
    target_app = target.Pose()*transl(0,50-10*z,-1.5*APPROACH-5*z-10)
    target_up = target.Pose()*transl(0,50-10*z,-5*z-10)
    target_back = target.Pose()*transl(0,-10*z,-1.5*APPROACH-5*z-10)
    target = target.Pose()*transl(0,-10*z,-3*z-10)
    robot.MoveL(target_app)
    robot.MoveL(target_up)
    robot.MoveL(target)
    TCP_Off(tool)
    robot.MoveL(target_back);

def Take_chamber(robot,target,z):
    target_app_1 = target.Pose()*transl(0,50-10*m,-1.5*APPROACH-5*m-10)
    target_up_1 = target.Pose()*transl(0,50-10*m,-5*m-10)
    target_back_1 = target.Pose()*transl(0,-10*m,-1.5*APPROACH-5*m-10)
    target = target.Pose()*transl(0,-10*m,-5*m-10)
    robot.MoveL(target_back_1)
    robot.MoveL(target)
    TCP_On(tool)
    robot.MoveL(target_up_1)
    robot.MoveL(target_app_1);

def Place_conv(robot,z):
    target = target_conv_3_place.Pose()*transl(0,-10*m,0)
    target_app_3 = target_conv_3_place.Pose()*transl(0,APPROACH-10*m,-APPROACH)
    target_up_3 = target_conv_3_place.Pose()*transl(0,APPROACH-10*m,0)
    target_back_3 = target_conv_3_place.Pose()*transl(0,-10*m,-APPROACH)
    robot.MoveJ(target_pass)
    robot.MoveL(target_app_3)
    robot.MoveL(target_up_3)
    robot.MoveL(target)
    TCP_Off(tool,frame_conv_3)
```

```
   robot.MoveL(target_back_3);

i = 1
n = -1
z = 1

Pick_conv(robot,target_pick_P1)
Place_chamber(robot,target_place_1,z)
pause(5.0)

while i>n:
   if RDK.RunMode() == RUNMODE_SIMULATE:
      # Simulate the sensor by waiting for the SENSOR status to turn to 1 (object present)
      while RDK.getParam(SENSOR_VARIABLE_3) == 0:
         pause(0.001)
         if RDK.getParam(SENSOR_VARIABLE_12) == 1:
            if RDK.getParam(SENSOR_VARIABLE_4) == 0:
               Take_chamber(robot,target_place_2,z)
               Place_conv(robot,z)
      if RDK.getParam(SENSOR_VARIABLE_3) == 1:
         i = i + 1
         pause(0.5)
         if i == 2:
            z = -2
            m = 1
            robot.MoveJ(target_home)
            Pick_conv(robot,target_pick_P2)
         elif i == 3:
            z = -13.5
            m = -2
            robot.MoveJ(target_home)
            Pick_conv(robot,target_pick_P3)
         elif i == 4:
            z = 0
            m = -13.5
            robot.MoveJ(target_home)
            Pick_conv(robot,target_pick_P6)
         elif i == 5:
            z = -1
            m = 0
            robot.MoveJ(target_home)
            Pick_conv(robot,target_pick_P9)
         elif i == 6:
            z = -8
            m = -1
            robot.MoveJ(target_home)
            Pick_conv(robot,target_pick_P12)

         if RDK.getParam(SENSOR_VARIABLE_4) == 0:
            print("Load chamber 1")
            Place_chamber(robot,target_place_1,z)

            while  RDK.getParam(SENSOR_VARIABLE_5) == 1 : #& (if test_2_finished==1)
               if RDK.getParam(SENSOR_VARIABLE_10) == 1:
                  print("Unload chamber 2 and put it on conv_3")
                  Take_chamber(robot,target_place_2,z)
                  Place_conv(robot,z)
         elif RDK.getParam(SENSOR_VARIABLE_5) == 0:
            print("Load chamber 2")
            Place_chamber(robot,target_place_2,z)
            while  RDK.getParam(SENSOR_VARIABLE_4) == 1 : #& (if test_1_finished==1)"""
               if RDK.getParam(SENSOR_VARIABLE_9) == 1:
                  print("Unload chamber 1 and put it on conv_3")
                  Take_chamber(robot,target_place_1,z)
                  Place_conv(robot,z)
```

j)

PASTOR - Aplikacija za upravljanje linijom.py

```python
# ubacivanje potrebnih biblioteka
import socket
import sys
import tkinter as tk
from PIL import ImageTk, Image
import time
import smtplib, ssl

# kreiranje lista za spremanje klijenata
all_connections = []
all_address = []

# lista za kreiranje dnevnog izvješća
list = []
list_B = ["Aparat P1 :","Aparat P2 :","Aparat P3 :"]
list.clear()

# parametri za slanje dnevnog izvješća putem emaila
port_s = 587  # For starttls
smtp_server = "smtp-mail.outlook.com"
sender_email = "kovadas@hotmail.com"
receiver_email = "dkeretic@hotmail.com"
password = 'rossonere'

# veličina GUI prozora
HEIGHT = 1500
WIDTH = 2000

# funkcija za kreiranje socketa
def create_socket():
    try:
        global host
        global port
        global s
        host = "127.0.0.10"
        port = 8000
        s = socket.socket()

    except socket.error as msg:
        print("Socket creation error: " + str(msg))


# funkcija postavljanje socketa i čekanja na konekciju klijenata
def bind_socket():
    try:
        global host
        global port
        global s
        print("Binding the Port: " + str(port))

        s.bind((host, port))
        s.listen(5)

    except socket.error as msg:
        print("Socket Binding error" + str(msg) + "\n" + "Retrying...")
        bind_socket()
```

```python
# funkcija za prihvaćanje klijenata (u ovom primjeru dva robota
def accepting_connections():
    for c in all_connections:
        c.close()

    del all_connections[:]
    del all_address[:]

    while True and len(all_connections)<2:
            if len(all_connections)<1:
                conn, address = s.accept()

                all_connections.append(conn)
                all_address.append(address)

            elif len(all_connections)<2:
                conn1, address1 = s.accept()

                all_connections.append(conn1)
                all_address.append(address1)

                response = tk.Label(root, text="SPOJENO", font
                response.place(x=95, y=100, relwidth=0.25)
```

```python
# funkcija za odspajanje
def disconnected():
    conn = all_connections[0]
    conn1 = all_connections[1]
    conn.close()
    conn1.close()
    s.close()
    response10 = tk.Label(root, text="ODSPOJENO", font=("Arial", 15),
    response10.place(x=95, y=100, relwidth=0.25)

# funkcija za gumbe koji se nalaze u GUI prozoru aplikacije
def on_click(args):
    global aparat
    if args == 1:
        create_socket()
        bind_socket()
        accepting_connections()
        print("Spajanje")
    if args == 2:
        aparat = 'P1'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 3:
        aparat = 'P2'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
```

```
    if args == 4:
        aparat = 'P3'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 5:
        aparat = 'P6'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 6:
        aparat = 'P9'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 7:
        aparat = '12'
        print(aparat)
        response15 = tk.Label(root, text='P12', font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 8:
        aparat = 'S6'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 9:
        aparat = 'S9'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 10:
    if args == 10:
        aparat = '13'
        print(aparat)
        response15 = tk.Label(root, text='S12', font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 11:
        aparat = '14'
        print(aparat)
        response15 = tk.Label(root, text='F6P', font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 12:
        aparat = 'F6'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 13:
        aparat = '15'
        print(aparat)
        response15 = tk.Label(root, text='F9P', font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 14:
        aparat = 'F9'
        print(aparat)
        response15 = tk.Label(root, text=aparat, font=("Arial", 15))
        response15.place(x=1300, y=675, relwidth=0.1)
    if args == 15:
        print("Start")
        send_commands()
```

```
    if args == 16:
        aparat = '99'
        print("Disconnect")
        send_commands()
        disconnected()
    if args == 17:
        aparat = 'CO'
        print("Continue")
        send_commands()
    if args == 18:
        aparat = 'BR'
        send_commands()
        conn = all_connections[0]
        conn1 = all_connections[1]
        client_response = str(conn.recv(1024), "utf-8")
        client_response_1 = str(conn1.recv(1024), "utf-8")
        if int(client_response) > int(client_response_1):
            count2 = tk.Label(root, text=client_response, font=("Arial", 15), ba
            count2.place(x=110, y=660, relwidth=0.2)
            print(client_response_1)
        elif int(client_response_1) > int(client_response):
            count2 = tk.Label(root, text=client_response_1, font=("Arial", 15), 
            count2.place(x=110, y=660, relwidth=0.2)
        else:
            count2 = tk.Label(root, text='0', font=("Arial", 15), background='#A
            count2.place(x=110, y=660, relwidth=0.2)

    if args == 19:
        aparat = 'RS'
            send_commands()
            count2 = tk.Label(root, text='0', font=("Arial", 15), b
            count2.place(x=110, y=660, relwidth=0.2)
        if args == 20:
            aparat = 'QU'
            send_commands()
            disconnected()
        if args == 21:
            global msg_full
            ukupno = 0
            aparat = 'CC'
            for i in range_(0,13):
                send_commands()
                time.sleep(0.1)
                conn = all_connections[0]
                conn1 = all_connections[1]
                client_response = str(conn.recv(1024), "utf-8")
                client_response_1 = str(conn1.recv(1024), "utf-8")
                if int(client_response) > int(client_response_1):
                    client_response_br = client_response
                elif int(client_response_1) > int(client_response):
                    client_response_br = client_response_1
                else:
                    client_response_br = ' 0'
                z = int(client_response_br)
                ukupno = ukupno + z
                time.sleep(0.1)
                list.append(list_B[i] + client_response_br)
```

```
                list.append(list_B[i] + client_response_br)
                i=i+1
                time.sleep(0.1)
        ukupno_str = """\n
UKUPNO: """ + str(ukupno)
        message = """\
Subject: BIJELA SOBA - Dnevni izvjestaj

Broj napravljenih aparata:\n
"""
        list_transpose = ('\n'.join(list))
        msg_full = message + list_transpose + ukupno_str
        print(msg_full)
        send_email()
        list.clear()

# funkcija za slanje podataka klijentima
def send_commands():
    conn = all_connections[0]
    conn1 = all_connections[1]
    try:
        if len(str.encode(aparat)) > 0:
            conn.send(str.encode(aparat))
            conn1.send(str.encode(aparat))
    except:
        disconnected()
```

```
def send_email():
    context = ssl.create_default_context()
    with smtplib.SMTP(smtp_server, port_s) as server:
        server.ehlo()  # Can be omitted
        server.starttls(context=context)
        server.ehlo()  # Can be omitted
        server.login(sender_email, password)
        server.sendmail(sender_email, receiver_email, msg_full)

# funkcija za zatvaranje GUI prozora
def on_close():
    if len(all_connections)==1:
        conn = all_connections[0]
        conn.close()
        s.close()
        sys.exit()
    elif len(all_connections)==2:
        conn = all_connections[0]
        conn1 = all_connections[1]
        conn.close()
        conn1.close()
        s.close()
        sys.exit()

    root.destroy()
```

```
# tkinter GUI za interakciju aplikacije i operatera
root = tk.Tk()
root.title("Pastor TVA - Linija za završnu montažu")
# veličina GUI prozora
WIDTH, HEIGHT = root.winfo_screenwidth(), root.winfo_screenheight()
root.geometry('%sx%s' % (WIDTH, HEIGHT))
root.configure(background='#DDDDDD')

img_naslov = ImageTk.PhotoImage(Image.open("Naslov.png"))
panel_3 = tk.Label(root, image=img_naslov,background="#DDDDDD" )
panel_3.place(x=520, y=30)

panel_2 = tk.Label(text="ZAVRŠNA MONTAŽA APARATA", font=("Times New Roman Bold", 26),background="#DDDDDD")
panel_2.place(x=500, y=120)


img = ImageTk.PhotoImage(Image.open("logo1.png"))
panel_3 = tk.Label(root, image=img,background="#DDDDDD" )
panel_3.place(x=650, y=180)

img2 = ImageTk.PhotoImage(Image.open("logo2.jpg"))
panel_4 = tk.Label(root, image=img2)
panel_4.place(x=585, y=400)

img3 = ImageTk.PhotoImage(Image.open("Slikice.png"))
panel_7 = tk.Label(root, image=img3,background="#DDDDDD")
panel_7.place(x=65, y=250)
```

```
img4 = ImageTk.PhotoImage(Image.open("Slikice2.png"))
panel_8 = tk.Label(root, image=img4,background="#DDDDDD")
panel_8.place(x=970, y=250)

button1 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(1))
button1["border"] = "0"
img10 = PhotoImage(file="CONNECT.png")
button1.config(image=img10)
button1.place(x=67, y=120, relwidth=0.25)

button_send = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(21))
button_send["border"] = "0"
img101 = PhotoImage(file="TODAY_DATA.png")
button_send.config(image=img101)
button_send.place(x=67, y=30, relwidth=0.25)

response10 = tk.Label(root, text="ODSPOJENO", font=("Times New Roman", 18), background="#FF0000")
response10.place(x=84, y=85, relwidth=0.23)

button2 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(16))
button2["border"] = "0"
img22 = PhotoImage(file="QUIT.png")
button2.config(image=img22)
button2.place(x=1080, y=125, relwidth=0.25)

panel_5 = tk.Label(text="P APARATI (STALNI TLAK)", font=("Times New Roman", 24,"bold","underline"),background="#DDDDDD")
panel_5.place(x=75, y=200)
```

```
panel_6 = tk.Label(text="S i F APARATI", font=("Times New Roman", 24,"bold","underline"),background="#DDDDDD")
panel_6.place(x=1140, y=210)

button3 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(2))
button3["border"] = "0"
img28 = PhotoImage(file="P1.png")
button3.config(image=img28)
button3.place(x=90, y=430, relwidth=0.12)

button4 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(3))
button4["border"] = "0"
img29 = PhotoImage(file="P2.png")
button4.config(image=img29)
button4.place(x=90, y=495, relwidth=0.12)

button5 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(4))
button5["border"] = "0"
img30 = PhotoImage(file="P3.png")
button5.config(image=img30)
button5.place(x=90, y=560, relwidth=0.12)

button6 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(5))
button6["border"] = "0"
img31 = PhotoImage(file="P6.png")
button6.config(image=img31)
button6.place(x=300, y=430, relwidth=0.12)
```

```
button7 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(6))
button7["border"] = "0"
img32 = PhotoImage(file="P9.png")
button7.config(image=img32)
button7.place(x=300, y=495, relwidth=0.12)

button8 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(7))
button8["border"] = "0"
img33 = PhotoImage(file="P12.png")
button8.config(image=img33)
button8.place(x=300, y=560, relwidth=0.12)

panel_9 = tk.Label(text="BROJ KOMADA:", font=("Times New Roman", 18),background="#DDDDDD" )
panel_9.place(x=195, y=625)

entry = tk.Entry(font=("Arial", 20), bg='#ACA9A9')
entry.place(x=125, y=660)

button181 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(18))
button181["border"] = "0"
img232 = PhotoImage(file="CHECK.png")
button181.config(image=img232)
button181.place(x=110, y=705, relwidth=0.105)

button19 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(19))
button19["border"] = "0"
img24 = PhotoImage(file="RESET.png")
button19.config(image=img24)
```

```
button9 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(8))
button9["border"] = "0"
img34 = PhotoImage(file="S6.png")
button9.config(image=img34)
button9.place(x=1050, y=430, relwidth=0.12)

button10 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(9))
button10["border"] = "0"
img35 = PhotoImage(file="S9.png")
button10.config(image=img35)
button10.place(x=1050, y=495, relwidth=0.12)

button11 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(10))
button11["border"] = "0"
img36 = PhotoImage(file="S12.png")
button11.config(image=img36)
button11.place(x=1050, y=560, relwidth=0.12)

button12 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(11))
button12["border"] = "0"
img37 = PhotoImage(file="F6.png")
button12.config(image=img37)
button12.place(x=1260, y=430, relwidth=0.12)

button13 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(12))
button13["border"] = "0"
img38 = PhotoImage(file="F9.png")
button13.config(image=img38)
```

```
    button15 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(14))
    button15["border"] = "0"
    img40 = PhotoImage(file="F9.png")
    button15.config(image=img40)
    button15.place(x=1155, y=625, relwidth=0.12)

    button16 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(17))
    button16["border"] = "0"
    img25 = PhotoImage(file="CONTINUE.png")
    button16.config(image=img25)
    button16.place(x=1040, y=705, relwidth=0.145)

    button17 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(15))
    button17["border"] = "0"
    img26 = PhotoImage(file="START.png")
    button17.config(image=img26)
    button17.place(x=1285, y=705, relwidth=0.105)

    button18 = Button(root, activebackground="black", bg='#DDDDDD', command=lambda: on_click(20))
    button18["border"] = "0"
    img18 = PhotoImage(file="TURN_OFF.png")
    button18.config(image=img18)
    button18.place(x=1080, y=55, relwidth=0.25)

    root.protocol('WM_DELETE_WINDOW', on_close)

    root.mainloop()
```

k)

```
PROGRAM Socket_V2
%STACKSIZE = 4000
%NOLOCKGROUP
%NOPAUSE=ERROR+COMMAND+TPENABLE
%ENVIRONMENT uif
%ENVIRONMENT sysdef
%ENVIRONMENT kclop
%ENVIRONMENT bynam
%ENVIRONMENT fdev
%ENVIRONMENT flbt
%ENVIRONMENT REGOPE
%INCLUDE klevccdf
%INCLUDE klevkeys
%INCLUDE klevkmsk


VAR
    file_var : FILE
    tmp_int  : INTEGER
    tmp_str  : STRING[128]
    status   : INTEGER
    entry    : INTEGER
    loop1    : BOOLEAN
    broj_kom : INTEGER
    status_2 : INTEGER
    number : INTEGER
    broj_kom_str : STRING[128]
    r_val : REAL
    rflag : BOOLEAN
    reg_rs : INTEGER


BEGIN
    reg_rs = 11
    number = 13
    MSG_DISCO('C1:',status)
    DOUT[1] = OFF
    DOUT[2] = OFF
    DOUT[3] = OFF
    DOUT[9] = OFF
    SET_REAL_REG(12,0, status)


label::

DELAY 100

SET_FILE_ATR(file_var, ATR_IA)
SET_VAR(entry, "'SYSTEM*','$HOSTC_CFG[1].$SERVER_PORT',8000,status)
WRITE(' VAR status = ',status,CR)
MSG_CONNECT('C1:',status)
WRITE(' Connect status = ',status,CR)
loop1 = TRUE
IF status = 0 THEN
    WHILE loop1 = TRUE DO
        DELAY 100
        WRITE('Opening file...',CR)
        OPEN FILE file_var('rw','C1:')
        status = IO_STATUS(file_var)
        IF status = 0 THEN
            WRITE('Waiting to read from server...')
            DELAY 100
            READ file_var(tmp_str::2)
            WRITE('Read: ',tmp_str::2,CR)
            DELAY 100

            IF tmp_str = 'P1' THEN
                SET_REAL_REG(12,1, status)
                DOUT[3] = OFF
                DOUT[1] = ON
                DELAY 50
            ENDIF

            IF tmp_str = 'P2' THEN
                SET_REAL_REG(12,2, status)
                DOUT[3] = OFF
                DOUT[1] = ON
                DELAY 50
            ENDIF

            IF tmp_str = 'P3' THEN
                SET_REAL_REG(12,3, status)
                DOUT[3] = OFF
                DOUT[1] = ON
```

```
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'P6' THEN
            SET_REAL_REG(12,4, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'P9' THEN
            SET_REAL_REG(12,5, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = '12' THEN
            SET_REAL_REG(12,6, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'S6' THEN
            SET_REAL_REG(12,7, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'S9' THEN
            SET_REAL_REG(12,8, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = '13' THEN
            SET_REAL_REG(12,9, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = '14' THEN
            SET_REAL_REG(12,10, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'F6' THEN
            SET_REAL_REG(12,11, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = '15' THEN
            SET_REAL_REG(12,12, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'F9' THEN
            SET_REAL_REG(12,13, status)
            DOUT[3] = OFF
            DOUT[1] = ON
            DELAY 50
        ENDIF

        IF tmp_str = 'CO' THEN
        DOUT[2] = ON
        DELAY 1000
        DOUT[2] = OFF
        ENDIF

        IF tmp_str = '99' THEN
            DOUT[3] = ON
            DELAY 3000
            DOUT[1] = OFF
```

```
            DOUT[2] = OFF
            DELAY 200

            CLOSE FILE file_var
            MSG_DISCO('C1:',status)
            GOTO label
        ENDIF

        IF tmp_str = 'QU' THEN
            DOUT[9] = ON
            DOUT[90] = ON
            DELAY 1000
            FOR tmp_int = 13 TO 25 DO
                SET_REAL_REG(tmp_int,0, status_2)
            ENDFOR
            CLOSE FILE file_var
            MSG_DISCO('C1:',status)
            number = 13
            ABORT
        ENDIF

        IF tmp_str = 'BR' THEN
            GET_REG(reg_rs, rflag, broj_kom, r_val, status_2)
            CNV_INT_STR(broj_kom, 0, 0, broj_kom_str)
            WRITE file_var(broj_kom_str)
            CLOSE FILE file_var
        ENDIF

        IF tmp_str = 'RS' THEN
            DOUT[99] = ON
            SET_REAL_REG(reg_rs, 0, status)
            DELAY 50
        ENDIF

        IF tmp_str = 'CC' THEN
            GET_REG(number, rflag, broj_kom, r_val, status_2)
            CNV_INT_STR(broj_kom, 0, 0, broj_kom_str)
            WRITE file_var(broj_kom_str)
            CLOSE FILE file_var
            number = number + 1
            IF number = 26 THEN
                number = 13
            ENDIF
        ENDIF

    CLOSE FILE file_var
    ELSE
        GOTO label
    ENDIF
  ENDWHILE
ENDIF
DELAY 200
GOTO label

END Socket_V2
```

l)

MAIN 1/52

```
  1:  RUN SOCKET
  2:
  3:  UFRAME_NUM=0
  4:  PAYLOAD[10]
  5:  OVERRIDE=80%
  6:
  7:  LBL[1]
  8:J @PR[1:HOME] 100% FINE
  9:  IF DO[1:START]=ON,JMP LBL[2]
 10:  IF (DO[9:ABORT ALL]=ON) THEN
 11:  DO[9:ABORT ALL]=OFF
 12:J @PR[1:HOME] 100% FINE
 13:  ABORT
 14:  ENDIF
 15:  JMP LBL[1]
 16:
 17:  LBL[2]
 18:
 19:  IF DO[3:STOP]=ON,JMP LBL[1]
 20:  IF (DO[9:ABORT ALL]=ON) THEN
 21:  DO[9:ABORT ALL]=OFF
```

MAIN 41/52

```
 22:J @PR[1:HOME] 100% FINE
 23:  ABORT
 24:  ENDIF
 25:
 26:  IF R[12:Control_Reg]=1,CALL P1
 27:  IF R[12:Control_Reg]=2,CALL P2
 28:  IF R[12:Control_Reg]=3,CALL P3
 29:  IF R[12:Control_Reg]=4,CALL P6
 30:  IF R[12:Control_Reg]=5,CALL P9
 31:  IF R[12:Control_Reg]=6,CALL P12
 32:  IF R[12:Control_Reg]=7,CALL S6
 33:  IF R[12:Control_Reg]=8,CALL S9
 34:  IF R[12:Control_Reg]=9,CALL S12
 35:  IF R[12:Control_Reg]=10,CALL F6P
 36:  IF R[12:Control_Reg]=11,CALL F6
 37:  IF R[12:Control_Reg]=12,CALL F9P
 38:  IF R[12:Control_Reg]=13,CALL F9
 39:
 40:
 41:  IF DO[3:STOP]=ON,JMP LBL[1]
 42:  JMP LBL[2]
```

P6 1/43

```
  1:J @PR[1:HOME] 100% FINE
  2:
  3:  IF (DI[1:AP_DETECTION]=ON) THEN
  4:  CALL PICK
  5:
  6:  IF (DI[2:CH_1_PRESENCE]=OFF)
   :  THEN
  7:  CALL PLACE_HELIUM_1
  8:
  9:  IF (DI[3:CH_2_PRESENCE]=ON) THEN
 10:  IF (DO[9:ABORT ALL]=ON) THEN
 11:  DO[9:ABORT ALL]=OFF
 12:J @PR[1:HOME] 100% FINE
 13:  ABORT
 14:  ENDIF
 15:
 16:  WAIT DI[5:CH_2_OPEN]=ON
 17:  CALL PICK_HELIUM_2
 18:  CALL PLACE_TRAKA_KUTIJE
 19:
 20:  ENDIF
```

P6 22/43

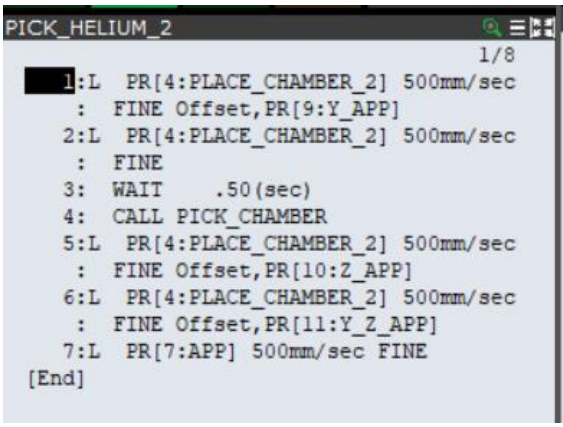
```
 21:
 22:  ELSE
 23:  IF (DI[3:CH_2_PRESENCE]=OFF)
   :  THEN
 24:  CALL PLACE_HELIUM_2
 25:
 26:
 27:  IF (DI[2:CH_1_PRESENCE]=ON) THEN
 28:  IF (DO[9:ABORT ALL]=ON) THEN
 29:J @PR[1:HOME] 100% FINE
 30:  WAIT    .25(sec)
 31:  ABORT
 32:  ENDIF
 33:
 34:  WAIT DI[4:CH_1_OPEN]=ON
 35:  CALL PICK_HELIUM_1
 36:  CALL PLACE_TRAKA_KUTIJE
 37:
 38:  ENDIF
 39:  ENDIF
 40:  ENDIF
```

PICK 1/19

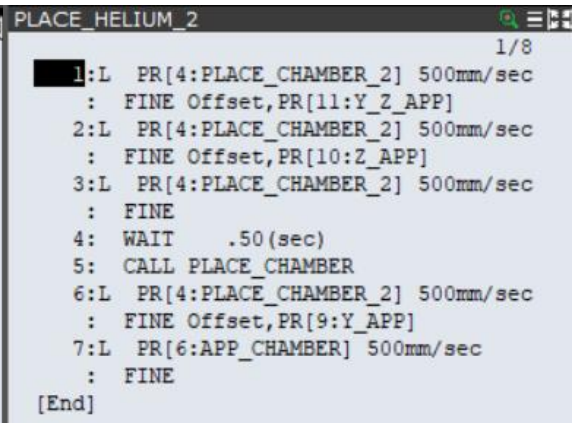
```
  1:J @PR[1:HOME] 100% FINE
  2:J  PR[2:PICK] 100% FINE
   :  Offset,PR[8:X_APP]
  3:L  PR[2:PICK] 500mm/sec FINE
  4:  WAIT    .50(sec)
  5:  CALL PICK_CONV
  6:L  PR[2:PICK] 500mm/sec FINE
   :  Offset,PR[10:Z_APP]
  7:J @PR[1:HOME] 100% FINE
  8:L  PR[6:APP_CHAMBER] 1000mm/sec
   :  FINE
  9:
```

PICK_HELIUM_1 1/8

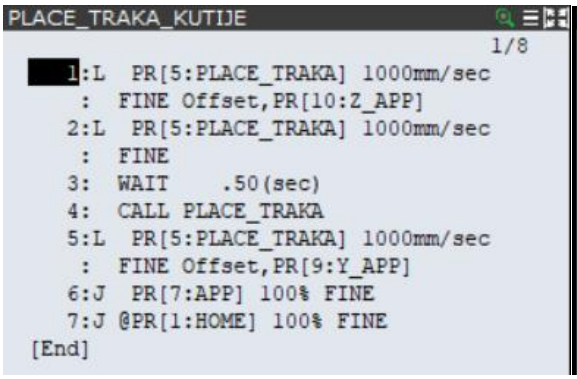
```
  1:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
   :  FINE Offset,PR[9:Y_APP]
  2:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
   :  FINE
  3:  WAIT    .50(sec)
  4:  CALL PICK_CHAMBER
  5:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
   :  FINE Offset,PR[10:Z_APP]
  6:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
   :  FINE Offset,PR[11:Y_Z_APP]
  7:L  PR[7:APP] 1000mm/sec FINE
[End]
```



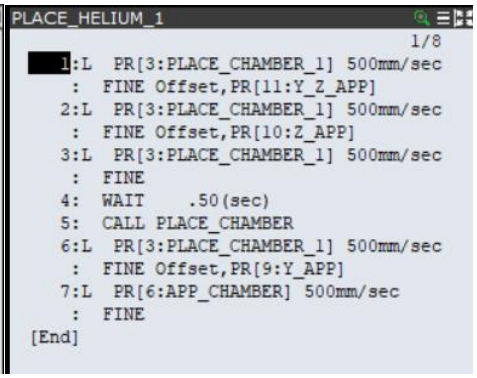
```
PICK_HELIUM_2                                    1/8
    1:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[9:Y_APP]
    2:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE
    3:  WAIT    .50(sec)
    4:  CALL PICK_CHAMBER
    5:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[10:Z_APP]
    6:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[11:Y_Z_APP]
    7:L  PR[7:APP] 500mm/sec FINE
[End]
```



```
PLACE_HELIUM_2                                   1/8
    1:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[11:Y_Z_APP]
    2:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[10:Z_APP]
    3:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE
    4:  WAIT    .50(sec)
    5:  CALL PLACE_CHAMBER
    6:L  PR[4:PLACE_CHAMBER_2] 500mm/sec
     :  FINE Offset,PR[9:Y_APP]
    7:L  PR[6:APP_CHAMBER] 500mm/sec
     :  FINE
[End]
```



```
PLACE_TRAKA_KUTIJE                               1/8
    1:L  PR[5:PLACE_TRAKA] 1000mm/sec
     :  FINE Offset,PR[10:Z_APP]
    2:L  PR[5:PLACE_TRAKA] 1000mm/sec
     :  FINE
    3:  WAIT    .50(sec)
    4:  CALL PLACE_TRAKA
    5:L  PR[5:PLACE_TRAKA] 1000mm/sec
     :  FINE Offset,PR[9:Y_APP]
    6:J  PR[7:APP] 100% FINE
    7:J @PR[1:HOME] 100% FINE
[End]
```



```
PLACE_HELIUM_1                                   1/8
    1:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
     :  FINE Offset,PR[11:Y_Z_APP]
    2:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
     :  FINE Offset,PR[10:Z_APP]
    3:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
     :  FINE
    4:  WAIT    .50(sec)
    5:  CALL PLACE_CHAMBER
    6:L  PR[3:PLACE_CHAMBER_1] 500mm/sec
     :  FINE Offset,PR[9:Y_APP]
    7:L  PR[6:APP_CHAMBER] 500mm/sec
     :  FINE
[End]
```

m)

INIT 1/11

```
  1:  R[4:X]=0
  2:  R[5:Y]=0
  3:  R[6:Z]=0
  4:
  5:  PR[3:PLACE]=PR[4:PLACE_0]
  6:
  7:  R[7:X_uk]=0
  8:  R[8:Y_uk]=0
  9:  R[9:Z_uk]=0
 10:
[End]
```

INIT_2 1/15

```
  1:  R[13:Number_P1]=0
  2:  R[14:Number_P2]=0
  3:  R[15:Number_P3]=0
  4:  R[16:Number_P6]=0
  5:  R[17:Number_P9]=0
  6:  R[18:Number_P12]=0
  7:  R[19:Number_S6]=0
  8:  R[20:Number_S9]=0
  9:  R[21:Number_S12]=0
 10:  R[22:Number_F6P]=0
 11:  R[23:Number_F6]=0
 12:  R[24:Number_F9P]=0
 13:  R[25:Number_F9]=0
```

MAIN_2 1/80

```
  1:  RUN SOCKET_V2
  2:
  3:  CALL INIT_2
  4:
  5:  CALL INIT
  6:  R[10:Count_pallet]=1
  7:  R[11:Count]=0
  8:
  9:  LBL[1]
 10:  CALL INIT
 11:  R[10:Count_pallet]=1
 12:  UFRAME_NUM=0
 13:
 14:  IF (DO[9:ABORT ALL]=ON) THEN
 15:  DO[9:ABORT ALL]=OFF
 16:J @PR[1:HOME] 100% FINE
 17:  ABORT
 18:  ENDIF
 19:
 20:J @PR[1:HOME] 100% FINE
 21:  DO[18]=OFF
```

MAIN_2 39/80

```
 22:  IF DO[1:START]=ON,JMP LBL[2]
 23:  JMP LBL[1]
 24:
 25:  LBL[2]
 26:
 27:  IF (R[10:Count_pallet] MOD 57=0)
   :  THEN
 28:  DO[18]=ON
 29:  CALL INIT
 30:  ENDIF
 31:
 32:  IF (R[10:Count_pallet] MOD 113=0
   :  ) THEN
 33:  WAIT DO[2:CONTINUE]=ON
 34:  DO[18]=OFF
 35:  R[10:Count_pallet]=1
 36:  CALL INIT
 37:  ENDIF
 38:
 39:  JMP LBL[3]
 40:
```

MAIN_2 59/80

```
 41:  LBL[3]
 42:
 43:  IF (DO[3:STOP]=ON) THEN
 44:  CALL INIT
 45:  R[10:Count_pallet]=0
 46:  JMP LBL[1]
 47:  ENDIF
 48:
 49:  IF (DO[9:ABORT ALL]=ON) THEN
 50:  DO[9:ABORT ALL]=OFF
 51:J @PR[1:HOME] 100% FINE
 52:  ABORT
 53:  ENDIF
 54:
 55:  IF DI[1:BOX_PRESENCE]=ON,
   :  JMP LBL[4]
 56:  JMP LBL[3]
 57:
```

MAIN_2 59/80

```
 58:  LBL[4]
 59:
 60:
 61:  IF R[12:Control_REG]=1,CALL P1
 62:  IF R[12:Control_REG]=2,CALL P2
 63:  IF R[12:Control_REG]=3,CALL P3
 64:  IF R[12:Control_REG]=4,CALL P6
 65:  IF R[12:Control_REG]=5,CALL P9
 66:  IF R[12:Control_REG]=6,CALL P12
 67:  IF R[12:Control_REG]=7,CALL S6
 68:  IF R[12:Control_REG]=8,CALL S9
 69:  IF R[12:Control_REG]=9,CALL S12
 70:  IF R[12:Control_REG]=10,CALL F6P
 71:  IF R[12:Control_REG]=11,CALL F6
 72:  IF R[12:Control_REG]=12,CALL F9P
 73:  IF R[12:Control_REG]=13,CALL F9
 74:
 75:  JMP LBL[2]
```

```
PLACE

  1:  IF (DO[18]=OFF) THEN
  2:  UFRAME_NUM=1
  3:  ENDIF
  4:
  5:  IF (DO[18]=ON) THEN
  6:  UFRAME_NUM=2
  7:  ENDIF
  8:
  9:  PAYLOAD[1]
 10:  OVERRIDE=85%
 11:
 12:J  PR[3:PLACE] 100% FINE
    :  Offset,PR[5:APPROACH]
 13:L  PR[3:PLACE] 500mm/sec FINE
 14:
 15:  WAIT    .50(sec)
 16:
 17:  IF (DO[18]=OFF) THEN
 18:  CALL GRIPPER_OPEN_1
 19:  ENDIF
 20:
 21:  IF (DO[18]=ON) THEN
 22:  CALL GRIPPER_OPEN_2
 23:  ENDIF
 24:
 25:J  PR[3:PLACE] 100% FINE
    :  Offset,PR[5:APPROACH]
 26:  R[10:Count_pallet]=
    :  R[10:Count_pallet]+1
 27:
 28:  R[11:Count]=R[11:Count]+1
 29:
 30:  UFRAME_NUM=0
 31:J @PR[1:HOME] 100% FINE
```

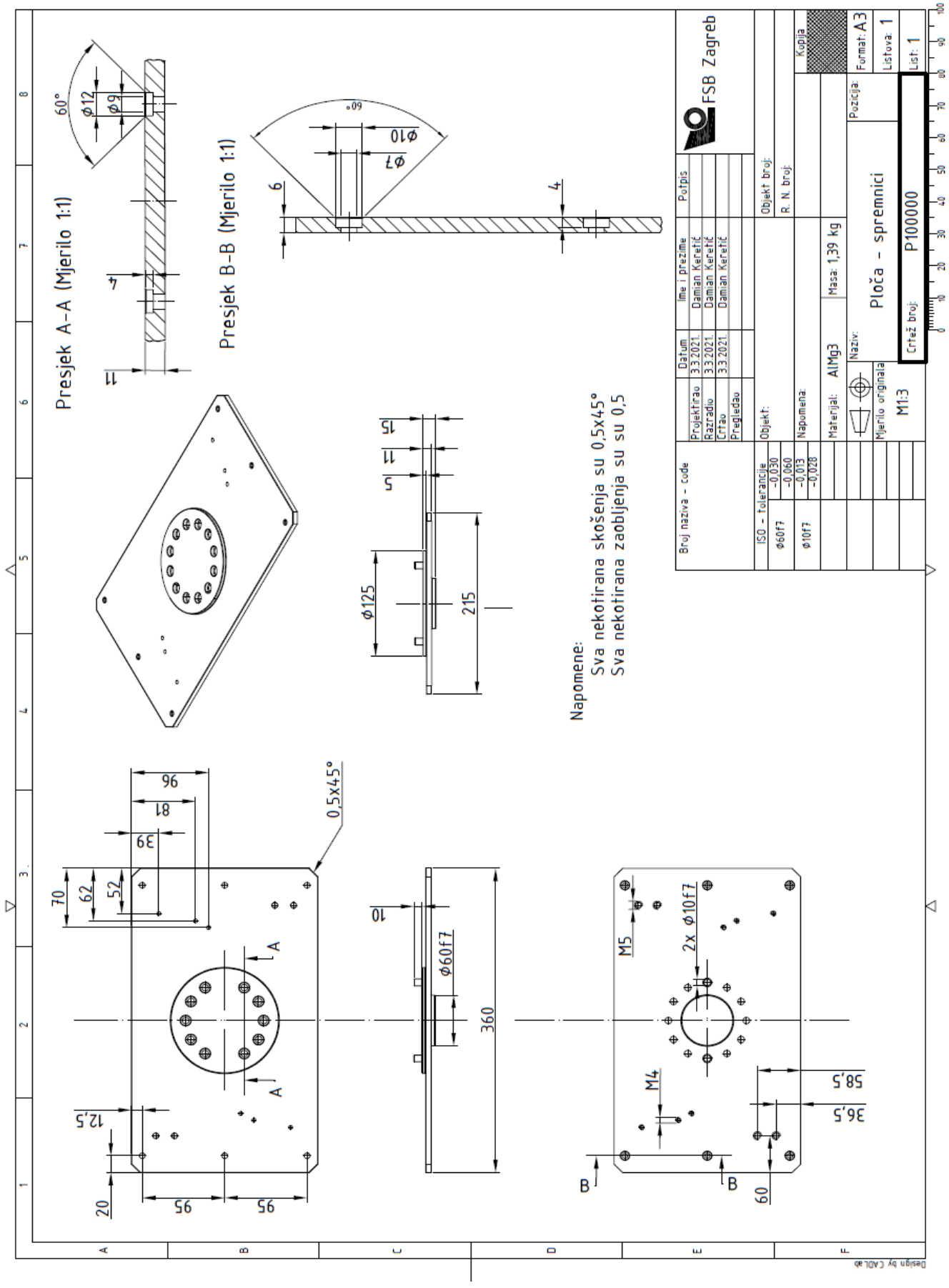
```
P6                                  15/39
  1:  UFRAME_NUM=1
  2:
  3:  R[7:X_uk]=R[1:X_offset]*R[4:X]
    :
  4:  R[4:X]=R[4:X]+1
  5:
  6:  IF (R[4:X]=5) THEN
  7:  R[5:Y]=R[5:Y]+1
  8:  R[4:X]=1
  9:  R[7:X_uk]=0
 10:  R[8:Y_uk]=R[2:Y_offset]*R[5:Y]
    :
 11:  ENDIF
 12:
 13:  IF (R[5:Y]=2) THEN
 14:  R[4:X]=1
 15:  R[5:Y]=0
```

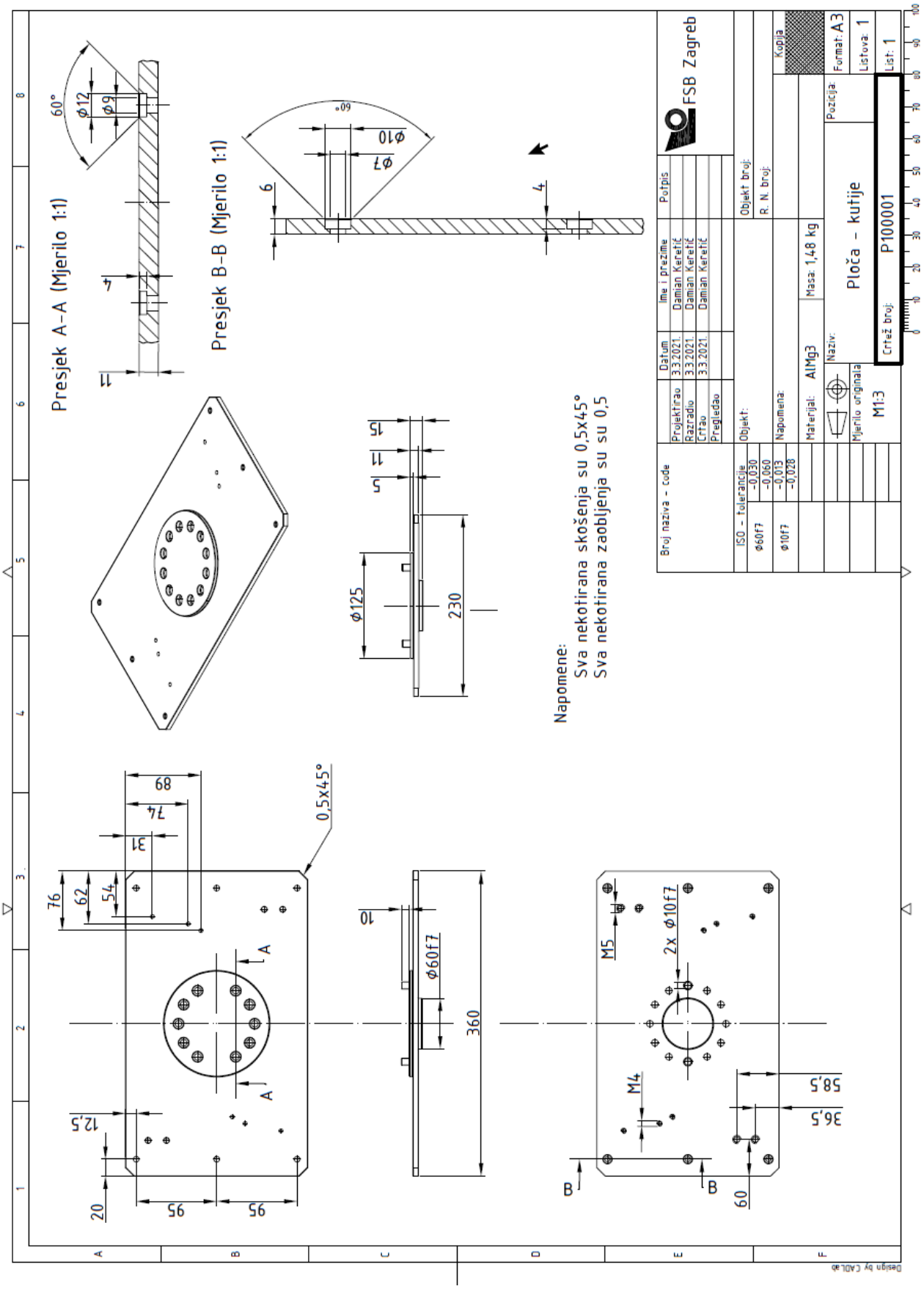
```
P6                                  28/39
 16:  R[7:X_uk]=0
 17:  R[8:Y_uk]=0
 18:  R[6:Z]=R[6:Z]+1
 19:  R[9:Z_uk]=R[3:Z_offset]*R[6:Z]
    :
 20:  ENDIF
 21:
 22:
 23:  PR[3,1:PLACE]=PR[4,1:PLACE_0]+
    :  R[7:X_uk]
 24:  PR[3,2:PLACE]=PR[4,2:PLACE_0]+
    :  R[8:Y_uk]
 25:  PR[3,3:PLACE]=PR[4,3:PLACE_0]+
    :  R[9:Z_uk]
 26:
 27:  CALL PICK
 28:  CALL PLACE
```

```
VAKUUM_ON                           1/8
  1:  DO[4:GRIPPER_CLOSE]=ON
  2:  DO[6:GRIPPER_CLOSE_2]=ON
  3:  WAIT    .25(sec)
  4:
  5:
  6:
  7:
```

```
VACUUM_OFF                          1/6
  1:  DO[4:GRIPPER_CLOSE]=OFF
  2:  DO[6:GRIPPER_CLOSE_2]=OFF
  3:  DO[5:GRIPPER_OPEN]=PULSE,0.5sec
  4:  DO[7:GRIPPER_OPEN_2]=
    :  PULSE,0.5sec
  5:  WAIT    .25(sec)
[End]
```

n)


Presjek A-A (Mjerilo 1:1)
Presjek B-B (Mjerilo 1:1)
Napomene:
Sva nekotirana skošenja su 0,5x45°
Sva nekotirana zaobljenja su su 0,5
FSB Zagreb
Ploča - spremnici
P100000
Format: A3
Masa: 1,39 kg
Materijal: AlMg3
M1:3
0,5x45°
2x Ø10f7
Ø60f7
M5
M4
360


Presjek A-A (Mjerilo 1:1)
Presjek B-B (Mjerilo 1:1)
Napomene:
Sva nekotirana skošenja su 0,5x45°
Sva nekotirana zaobljenja su su 0,5
FSB Zagreb
Broj naziva - code
Projektirao
Razradio
Crtao
Pregledao
Datum
3.3.2021.
Ime i prezime
Damian Keretić
Potpis
Objekt:
Objekt broj:
R. N. broj:
Napomena:
Materijal: AlMg3
Masa: 1,48 kg
Naziv:
Ploča - kutije
Mjerilo originala
M1:3
Pozicija:
Format: A3
Listova: 1
List: 1
Crtež broj: P100001
Kopija
ISO - tolerancije
Ø60f7 -0,030 -0,060
Ø10f7 -0,013 -0,028
0,5x45°
2x Ø10f7
Ø60f7
M5
M4
360
230
Ø125

o)

```python
import cv2
import os
import glob
from PIL import Image

n = 0
images = []
obradene = []
data = []

path = glob.glob("OCR/ocr/venv/slike1/*.jpeg")
for img in path:
    n = cv2.imread(img)
    data.append(n)

for img in data:
    n = n + 1
    img = img[280:650, 250:380]
    images.append(img)
    cv2.imshow(str(n), img)

for img in images:
    n = n + 1
    img1 = img[20:80, 20:100]
    cv2.imshow(str(n),img1)
    obradene.append(img1)

    n = n + 1
    img2 = img[75:135, 20:100]
    cv2.imshow(str(n),img2)
    obradene.append(img2)

    n = n + 1
    img3 = img[130:190, 25:105]
    cv2.imshow(str(n),img3)
    obradene.append(img3)

    n = n + 1
    img4 = img[180:240, 30:110]
    cv2.imshow(str(n),img4)
    obradene.append(img4)

    n = n + 1
    img5 = img[240:300, 30:110]
    cv2.imshow(str(n),img5)
    obradene.append(img5)

    n = n + 1
    img6 = img[290:350, 30:110]
    cv2.imshow(str(n),img6)
    obradene.append(img6)

for img in obradene:
    n = n + 1

    image2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    cv2.imshow("sss",image2)
    image2 = cv2.GaussianBlur(image2, (5, 5), 0)
    # image_ = cv2.GaussianBlur(image_, (5, 5), 0)
    image2 = cv2.medianBlur(image2, 5)
    # image_ = cv2.medianBlur(image_, 5)
    image2 = cv2.rotate(image2, cv2.ROTATE_90_COUNTERCLOCKWISE)
    img = cv2.medianBlur(image2, 5)
    filterSize = (7, 7)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, filterSize)
    corrected = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
    BW1 = cv2.threshold(corrected, 2, 255, cv2.THRESH_BINARY)[1]
    # cv2.imshow("corrected",corrected)

    filterSize2 = (2, 1)
    kernel2 = cv2.getStructuringElement(cv2.MORPH_RECT, filterSize2)
    marker = cv2.erode(corrected, kernel2)
    # cv2.imshow("marker",marker)
    BW2 = cv2.threshold(marker, 1, 255, cv2.THRESH_BINARY)[1]

    cv2.imwrite('slike1/' + str(n) + ".jpeg", BW2)

cv2.waitKey()
cv2.destroyAllWindows()
```

p)

```python
## treniranje neuronske mreže na binarnim slikama znamenaka 0-9 za OCR prepoznavanje serijskog broja

import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import os
import numpy as np
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.models import load_model


## normalizacija inputa ako slika nije u binarnom formatu (dobijemo brojke od 0-1)
## u ovom primjeru slike su već binarizirane tako da je skaliranje postavljeno kao 1/1
train = ImageDataGenerator(rescale=1/1)
validation = ImageDataGenerator(rescale=1/1)
```

```
## učitavanje trening i validation podataka, batch size je koliko se primjera prođe u jednoj iteraciji
train_dataset = train.flow_from_directory("Training",
                                          target_size=(60,80),
                                          batch_size = 3,
                                          class_mode = 'sparse')

validation_dataset = validation.flow_from_directory("Validation",
                                          target_size=(60,80),
                                          batch_size = 3,
                                          class_mode = 'sparse')

# pritanje pronađenih klasa iz training podataka
print(train_dataset.class_indices)
```

```
# kreiranje keras modela za duboko učenje (neuronske mreže)
# sekvencijalni model omogućuje dodavanje slojeva jedan po jedan
model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(60,80,3)),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
                                    tf.keras.layers.MaxPool2D(2, 2),

                                    tf.keras.layers.Flatten(),

                                    tf.keras.layers.Dense(512,activation='relu'),
                                    tf.keras.layers.Dense(9,activation='sigmoid')
                                    ])

model.compile(loss = 'sparse_categorical_crossentropy',
              optimizer = tf.keras.optimizers.SGD(learning_rate=0.001),
              metrics=['accuracy'])
```

```
tf.keras.callbacks.EarlyStopping(monitor = "val_loss",
                                 min_delta=0,
                                 patience=0,
                                 verbose = 0,
                                 mode = 'auto',
                                 baseline=None,
                                 restore_best_weights=False)

callback = [EarlyStopping(monitor='val_loss', patience=50),
            ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

model_fit = model.fit(train_dataset,
                      steps_per_epoch=20,
                      epochs=200,
                      validation_data=validation_dataset,
                      callbacks=callback)

model.save('Custom OCR full.verzija.h5')
```

g)

```python
import cv2
import os
import numpy as np
from keras.preprocessing.image import img_to_array, array_to_img, load_img
from keras.models import load_model

img = cv2.imread('Original_photos/1.jpeg')
cv2.imshow("Original",img)
image_ = img[300:660, 200:330]
#cv2.imshow("Prva",image_)
images = []
obradene = []

n = 1
img1 = image_[20:80, 20:100]
#cv2.imshow(str(n),img1)
images.append(img1)

n = n + 1
img2 = image_[75:135, 20:100]
#cv2.imshow(str(n),img2)
images.append(img2)

n = n + 1
img3 = image_[130:190, 25:105]
#cv2.imshow(str(n),img3)
images.append(img3)

n = n + 1
img4 = image_[180:240, 30:110]
#cv2.imshow(str(n),img4)
images.append(img4)

n = n + 1
img5 = image_[240:300, 30:110]
#cv2.imshow(str(n),img5)
images.append(img5)

n = n + 1
img6 = image_[290:350, 30:110]
#cv2.imshow(str(n),img6)
images.append(img6)
img6 = image_[290:350, 30:110]
```

```python
for img in images:
    number = number + 1

    image2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    image2 = cv2.GaussianBlur(image2, (5, 5), 0)
    # image_ = cv2.GaussianBlur(image_, (5, 5), 0)
    image2 = cv2.medianBlur(image2, 5)
    # image_ = cv2.medianBlur(image_, 5)
    image2 = cv2.rotate(image2, cv2.ROTATE_90_COUNTERCLOCKWISE)

    img = cv2.medianBlur(image2, 5)
    filterSize = (7, 7)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, filterSize)
    corrected = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
    BW1 = cv2.threshold(corrected, 2, 255, cv2.THRESH_BINARY)[1]
    # cv2.imshow("corrected",corrected)

    filterSize2 = (2, 1)
    kernel2 = cv2.getStructuringElement(cv2.MORPH_RECT, filterSize2)
    marker = cv2.erode(corrected, kernel2)
    # cv2.imshow("marker",marker)
    BW2 = cv2.threshold(marker, 1, 255, cv2.THRESH_BINARY)[1]
    cv2.imwrite('Nova mapa/' + str(number) + ".jpeg", BW2)

model = load_model('Custom OCR full.verzija.h5')

string = ""

dir_path = "Nova mapa"
for i in os.listdir(dir_path):
    img = load_img(dir_path + '//' + i, target_size=(60, 80))
    x = img_to_array(img)
    x = np.expand_dims(x,axis=0)

    val = np.argmax(model.predict(x), axis=-1)
    print(i)

    if (str(val)) == "[0]":
        print("Ovo je broj: 0")
        string = string + '0'
    if (str(val)) == "[1]":
        print("Ovo je broj: 1")
        string = string + '1'
```

```python
    if (str(val)) == "[4]":
        print("Ovo je broj: 4")
        string = string + '4'
    if (str(val)) == "[5]":
        print("Ovo je broj: 5")
        string = string + '5'
    if (str(val)) == "[6]":
        print("Ovo je broj: 6")
        string = string + '6'
    if (str(val)) == "[7]":
        print("Ovo je broj: 7")
        string = string + '7'
    if (str(val)) == "[8]":
        print("Ovo je broj: 8")
        string = string + '8'
    if (str(val)) == "[9]":
        print("Ovo je broj: 9")
        string = string + '9'

print("Serijski broj je:",string)
cv2.waitKey()
cv2.destroyAllWindows()
```