

Daljinsko upravljanje i nadzor pneumatskog manipulatora

Vico, Anđelko

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:235:019866>

Rights / Prava: [Attribution 3.0 Unported](#)/[Imenovanje 3.0](#)

Download date / Datum preuzimanja: **2024-04-17**

Repository / Repozitorij:

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



SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Andelko Vico

Zagreb, godina 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Željko Šitum, dipl. ing.

Student:

Anđelko Vico

Zagreb, godina 2021.

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

Zahvaljujem se mentoru prof. dr. sc. Željku Šitumu, dipl.ing. na ukazanom povjerenju prihvaćanjem mentorstva te na odvojenom vremenu za savjete i pomoć.

Također, zahvaljujem se ko-mentor Juraju Beniću, mag.ing. na ukazanom povjerenju te za vođenje tijekom izrade rada s korisnim savjetima.

Zahvaljujem obitelji na neprestanoj potpori pruženoj kroz studij.

Andelko Vico



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: **ANDELKO VICO** Mat. br.: 0035201621

Naslov rada na hrvatskom jeziku: **Daljinsko upravljanje i nadzor pneumatskog manipulatora**

Naslov rada na engleskom jeziku: **Remote control and monitoring of the pneumatic manipulator**

Opis zadatka:

U Laboratoriju za automatiku i robotiku nalazi se pneumatski manipulator koji je prije tridesetak godina izrađen u tvrtki Festo za edukacijske svrhe. Sastoji se od devet pneumatskih cilindara, dvije vakuumske prihvatanice te od jedanaest elektromagnetskih ventila za upravljanje cilindrima i prihvatanicama. Cilindri omogućuju dva translacijska i dva rotacijska gibanja manipulatora te jedno translacijsko gibanje za svaku od dvije palete. Palete služe za odlaganje valjkastih predmeta koje manipulator prenosi iz jedne palete u drugu. Upravljanje je bilo izvedeno pomoću programabilnog logičkog kontrolera (PLC-a) proizvođača Festo. S obzirom da navedeni PLC više nije u upotrebljivom stanju, potrebno ga je zamijeniti novim upravljačkim uređajem i osuvremeniti upravljanje s ciljem daljinskog upravljanja i nadzora rada manipulatora primjenom tehnologije Interneta stvari prema načelima Industrije 4.0.

U radu je potrebno:

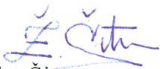
- pregledati sve pneumatske i elektroničke dijelove radne stanice, zamijeniti dotrajale komponente i dovesti radnu stanicu u funkcionalno stanje,
- načiniti upravljački program u Arduino jeziku za implementaciju u PLC,
- ostvariti upravljanje radne stanice putem Interneta korištenjem Modbus TCP/IP protokola, s mogućnošću prikaza podataka na korisničkom računalu,
- povezati upravljački program s aplikacijom za daljinsko upravljanje i nadzor sustava.

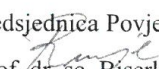
U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
6. svibnja 2021.

Rok predaje rada:
8. srpnja 2021.

Predviđeni datum obrane:
12. srpnja do 16. srpnja 2021.

Zadatak zadao: 
prof. dr. sc. Željko Šitum

Predsjednica Povjerenstva:

prof. dr. sc. Biserka Runje

SADRŽAJ

SADRŽAJ	I
POPIS SLIKA	I
POPIS TABLICA.....	I
POPIS TEHNIČE DOKUMENTACIJE	I
SAŽETAK.....	I
SUMMARY	II
1. UVOD.....	1
2. INTERNET STVARI	2
3. PREGLED PNEUMATSKE RADNE STANICE.....	4
3.1. Zatečeno stanje.....	4
3.2. Komponente za zamjenu	6
3.3. Nove komponente za ugradnju	12
3.4. Stanje nakon ugradnje novih komponenti.....	18
4. PROTOKOL ZA PODATKOVNU KOMUNIKACIJU MODBUS TCP/IP	19
5. UPRAVLJAČKI PROGRAM PNEUMATSKE RADNE STANICE.....	22
5.1. Arduino programski jezik	22
5.2. Razvojno okruženje Visual Studio 2019.....	23
5.3. Konfiguriranje razvojnog okruženja	24
5.4. Struktura programa	28
5.4.1. Inicijalizacijski dio programa	28
5.4.2. Izvršni dio programa	30
5.4.2.1. Automatski način	31
5.4.2.2. Point-to-point	33
5.4.2.3. Ručno upravljanje	34
6. PROGRAMSKO RJEŠENJE ZA UPRAVLJANJE I NADZOR PREKO WEB-A	35
6.1. Konfiguriranje Internet postavki rutera.....	35
6.2. Konfiguriranje video nadzora	37
6.3. Korištenje sučelja aplikacije <i>CyberHydraulic</i>	39
7. ZAKLJUČAK.....	43
LITERATURA.....	44
PRILOZI.....	46

POPIS SLIKA

Slika 1.	Vizualan prikaz koncepta Interneta stvari [1]	2
Slika 2.	Stanica pneumatskog manipulatora u zatečenom stanju	4
Slika 3.	Glava manipulatora	5
Slika 4.	Stanica manipulatora - prednji pogled	5
Slika 5.	Stanica manipulatora - bočni pogled	6
Slika 6.	Festo FPC 202 (gore) s proširenjem (dolje)	7
Slika 7.	Blok MFH-5-1/8 ventila	7
Slika 8.	Pneumatski 3/2 ventil s elektromagnetskom zavojnicom	8
Slika 9.	Dva VAD-1/4 generatora vakuuma s prigušnicama	8
Slika 10.	Glava za kontrolu vakuuma [8]	9
Slika 11.	Element LC-3-1/8 [9]	9
Slika 12.	Tlačna sklopka PE-1/8-1N [9]	10
Slika 13.	Induktivni senzori za poziciju klipa SME-1-LED-24	10
Slika 14.	Prigušni ventil GRL-1/4-PK-6-KU	11
Slika 15.	Upravljački uređaji: CONTROLLINO MINI, MAXI i MEGA [10]	12
Slika 16.	Upravljački uređaj CONTROLLINO MAXI Automation [12]	13
Slika 17.	Primjer VTUG ventilskog razvodnika s 8 ventila [13]	14
Slika 18.	Generator vakuuma VN-05-H-T3-PQ2-VQ2-RO1 [14]	15
Slika 19.	Senzor vakuuma i njegov konektor [15]	15
Slika 20.	Vakuumska hvataljka [16]	16
Slika 21.	Induktivni senzor i njegov držač [17]	16
Slika 22.	Novi prigušno nepovratni ventil [18]	16
Slika 23.	Pripremna grupa LFR-1/4-DB-7-MINI-KC [19]	17
Slika 24.	Stanje radne stanice nakon ugradnje novih komponenti	18
Slika 25.	Modbus <i>Client/Server</i> izmjena poruka [20]	19
Slika 26.	Struktura jednog Modbus TCP/IP paketa [21]	20
Slika 27.	Arduino IDE sučelje	22
Slika 28.	Visual Studio 2019 sučelje	23
Slika 29.	<i>IntelliSense</i> prijedlog prilikom tipkanja	23
Slika 30.	Preuzimanje proširenja Visual Micro	24
Slika 31.	Instaliranje dodatka Visual Micro	25
Slika 32.	Definiranje Arduino IDE instalacijske mape	25
Slika 33.	Instaliranje CONTROLLINO biblioteka	26
Slika 34.	Unos web poveznice u Arduino IDE-u	27
Slika 35.	Instaliranje CONTROLLINO biblioteka	27
Slika 36.	Uvodni dio kôda radne stanice	28
Slika 37.	Isječak iz <i>setup()</i> dijela kôda	29
Slika 38.	Isječak iz <i>setup()</i> dijela kôda	30
Slika 39.	Pregled <i>loop()</i> petlje	31
Slika 40.	Zaprimanje 32-bitnog broja	32
Slika 41.	Isječak algoritma za automatsko slaganje valjčića	32
Slika 42.	Struktura kôda drugog modula	33
Slika 43.	Struktura kôda trećeg modula rada programa	34
Slika 44.	Ruter <i>TP-Link Archer C20 AC750</i> [23]	35
Slika 45.	Postavljanje statičke IP adrese	36

Slika 46.	Rezervacija lokalnih IP adresa	36
Slika 47.	Postavljanje <i>Port-forwarding</i> -a za IP kameru	37
Slika 48.	Raspberry Pi 4 model B (lijevo) [24]; Logitech C170 web kamera (desno) [25] .	37
Slika 49.	Postavke unutar programa MotionEye	38
Slika 50.	Prikaz aplikacije <i>CyberHydraulic</i>	39
Slika 51.	<i>Dashboard</i> aplikacije <i>CyberHydraulic</i>	39
Slika 52.	Izbornik <i>System list</i>	40
Slika 53.	Prozor za dodavanje novog sustava.....	40
Slika 54.	Tablica registara sustava	41
Slika 55.	Pregled pneumatskih sustava.....	41
Slika 56.	Kontrolna ploča sustava	42

POPIS TABLICA

Tablica 1. Usporedba CONTROLLINO modela [11].....	13
Tablica 2. Tipovi Modbus tablica.....	20
Tablica 3. Najbitniji funkcijski kodovi.....	21

POPIS TEHNIČE DOKUMENTACIJE

PROGRAMSKI KÔD STANICE

SAŽETAK

U ovom zadatku opisana je obnova edukacijske radne stanice pneumatskog manipulatora. Opisan je koncept Interneta stvari, protokol Modbus TCP/IP i njihova primjena na radnu stanicu. Zatim je prikazan proces programiranja upravljačkog uređaja (PLC) stanice u Arduino jeziku. Naposljetku, opisana je konfiguracija Internet postavki i video nadzora kako bi se cijela radna stanica mogla upravljati i nadzirati preko razvijene web platforme *CyberHydraulic*.

Ključne riječi: Internet stvari, Modbus TCP/IP, pneumatika, Arduino, Raspberry Pi

SUMMARY

This work describes the renewal of a educational pneumatic manipulator station. The concept of the Internet of Things, the Modbus TCP / IP protocol and their application to the pneumatic station are also described. Afterwards, the process of programming the station's PLC in Arduino language is shown. Finally, the configuration of Internet and video surveillance settings is explained so that the entire station can be managed and monitored via the web platform *CyberHydraulic*.

Key words: Internet of Things, Modbus TCP/IP, pneumatics, Arduino, Raspberry Pi

1. UVOD

Napretkom tehnologije nezaobilazno dolazi do zastarenja ustaljenih načina proizvodnje, istraživanja, ali i razmišljanja. Prije ili kasnije javlja se potreba za modernizacijom. U ovom diplomskom radu načinjena je obnova jedne zastarjele edukacijske radne stanice pneumatskog manipulatora u Laboratoriju za automatiku i robotiku Fakulteta strojarstva i brodogradnje.

Oprema koja se trenutno nalazi na radnoj stanici datira iz 1980.-ih i potrebno ju je zamijeniti suvremenim dijelovima. Iako su mnogi dijelovi i dalje, zbog svoje robusnosti, funkcionalni kao npr. razvodnici, prigušivači i ventili, moderna oprema donosi niz drugih pogodnosti. Veličinom su puno manji, lakše je održavanje, a jednostavnija je instalacija i povezivanje s PLC-om. Ne treba zaboraviti kako današnji PLC-ovi nude nove mogućnosti nezamislive do prije 40 godina, koliko je stara oprema.

Kako se nalazimo u dobu u kojem je Internet sve dostupniji, uređaji sve pametniji i sve dostupniji cijenom, prirodno se javlja potreba za primjenom novih tehnologija i usvajanjem novih pristupa. Ovdje se naravno govori o Industriji 4.0. Ona ovisi o nizu novih i inovativnih tehnologija, a najvažnija među njima je Internet stvari, o čemu će biti riječ u nastavku. Stoga će se u ovom radu, osim osuvremenjivanjem opreme, obrađivati implementacija upravljanja i nadzora radne stanice putem Interneta.

2. INTERNET STVARI

Prije nego razmotrimo što je *Internet stvari*, dobro je shvatiti što točno riječ Internet označava. Internet je skraćenica od "network of intra-networks" ili jednostavno mreža više mreža. Bilo koje poduzeće ili kućanstvo može imati svoju lokalnu mrežu uređaja koji komuniciraju međusobno bez da su spojeni na Internet tj. s ostatkom svijeta. Ovakva mreža zove se intranet. Ako više intraneta umnožimo kroz neki zajednički protokol kao što je to Internet, dobivamo povezanost na globalnoj razini.

S druge strane Internet stvari, ili skraćeno IoT (eng. Internet of Things), odnosi se na sustav međusobno povezanih uređaja gdje svatko ima svoj unikatni identifikacijski broj i mogućnost da samostalno šalje podatke preko mreže. Glavni cilj interneta stvari jest imati sustav u kojem uređaji sami javljaju svoje stanje u stvarnom vremenu. IoT uređaji su opremljeni senzorima, aktuatorima, sustavom za komunikaciju i software-om koji dohvaća, filtrira i izmjenjuje podatke o sebi, svom stanju i stanju okoliša. Time se povećava cjelokupna efikasnost, jer sustav može sam donositi odluke i manje je ovisan o ljudskoj intervenciji. "Stvar" u imenu "Internet stvari" može, u teoriji, označavati bilo koju komponentu povezanu na Internet s IoT tehnologijom, bio to fizički objekt ili živo biće. Pojam IoT se uglavnom koristi za uređaje od kojih generalno ne očekujemo da imaju mogućnost spajanja na Internet. Dakle, pod tim pojmom ne podrazumijevamo laptope, pametne mobitele ili servere.



Slika 1. Vizualan prikaz koncepta Interneta stvari [1]

Na potrošačkoj razini se vidi sve veća dostupnost "pametnih uređaja". Prednosti internetske povezanosti za neke uređaje su očite, dok za druge nisu toliko da bi kupci bili odmah zainteresirani, nešto što se može reći, između ostalog, za pametne hladnjake. Pametni zvučnici nam omogućuju da putem glasovnih naredbi interaktivno komuniciramo s ugrađenim virtualnim asistentom koji nam može servirati informacije o vremenu, reproducirati glazbu, podsjetiti na obveze itd. Termostati mogu naučiti kada smo najčešće doma i time automatski regulirati temperaturu prostorije. Pametni automobili mogu sami izvršiti tehničku dijagnozu i naručiti potrebne dijelove [2].

Iako je tržište prepuno primjera kako Internet stvari postaje realnost, ono se i dalje nalazi u začetku što se tiče razvoja zbog tehničkih izazova koje je potrebno prebroditi. Uspješna primjena Interneta stvari podrazumijeva pronalazak najprije smislenog slučaja upotrebe, a potom prilagodbu postojećeg sustava i organizacije. Nakon uspješne tehničke provedbe javlja se potreba za kontinuiranim održavanjem cijelog sustava. Uz sve to, potrebno je istovremeno zadržati privatnost korisnika i kibernetičku sigurnost koja je *condicio sine qua non*, jer koliko pozitivnih stvari može donijeti toliko kod zlorabe može biti opasan.

Na koncu, važno je naglasiti da, slično kao i u slučaju pojave Interneta, samim povezivanjem uređaja ne stvara se nikakva vrijednost, nego ona nastaje tek kad se pronađe praktična primjena. U ovom radu, praktična primjena se očituje u omogućivanju daljinskog upravljanja i nadzora nad radnom stanicom korištenjem komunikacijskog protokola Modbus TCP/IP.

3. PREGLED PNEUMATSKE RADNE STANICE

U sklopu izrade ovog rada, korištena je postojeća pneumatska radna stanica koja se nalazi u Laboratoriju za automatiku i robotiku Fakulteta strojarstva i brodogradnje. Stanica može poslužiti za demonstraciju rada pneumatskih aktuatora u svrhu manipulacije objektima koji se nalaze na razdvojenim stolovima.

3.1. Zatečeno stanje

Na slici 2 prikazana je radna stanica u zatečenom stanju. Stanica se sastoji od pomične glave manipulatora s dvije vakuumske hvataljke i dva pomična stola, svaki sa po 9 mjesta za valjčiće. Na slici se također vidi da na stanici nedostaju pneumatske cijevi, da su senzori razmješteni i da se koriste stari tipovi elektromagnetskih ventila.



Slika 2. Stanica pneumatskog manipulatora u zatečenom stanju

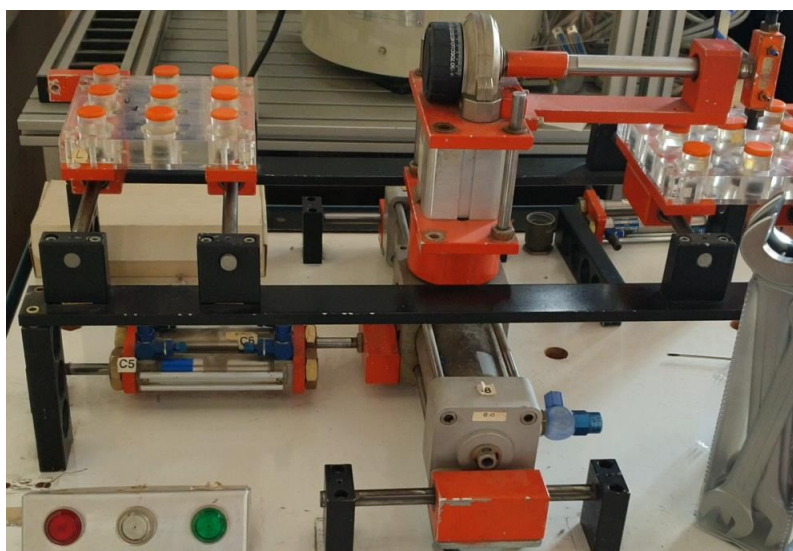
Manipulator radne stanice ima četiri stupnja slobode gibanja:

- **Rotacija glave manipulatora** – Omogućuje korištenje dviju vakuumske hvataljke na jednoj glavi. Zbog svoje istrošenosti silikonske hvataljke će biti zamijenjene novima.



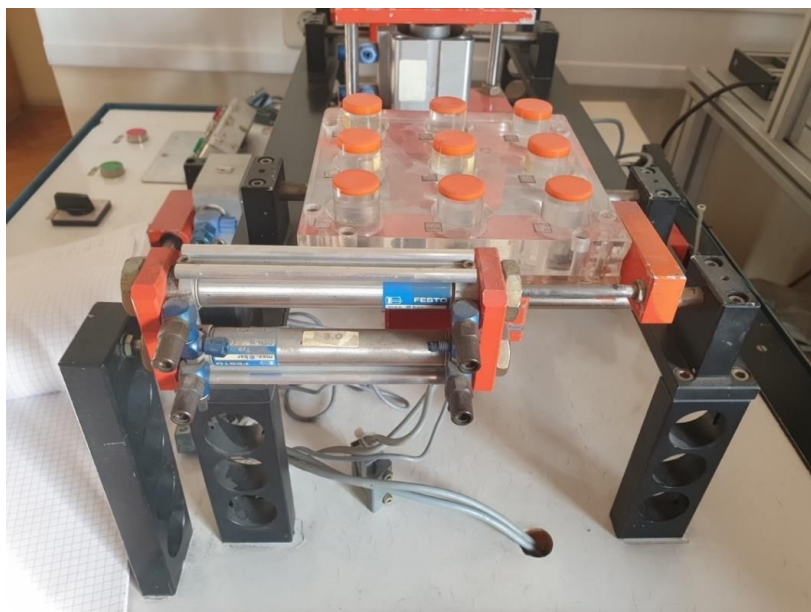
Slika 3. Glava manipulatora

- **Pomak glave gore-dolje** – Dvije moguće pozicije glave.
- **Rotacija postolja za 180 stupnjeva** – Dvije moguće pozicije postolja.
- **Pomak postolja lijevo-desno** – Korištenjem dva cilindra moguće je ostvariti tri različite pozicije postolja.



Slika 4. Stanica manipulatora - prednji pogled

Oba stola mogu se, uz pomoć dva cilindra, pravocrtno gibati u neku od tri moguće pozicije. Time se glavi manipulatora omogućuje pristup ka sva tri reda valjčića na stolu, kao što je to vidljivo na slici 5.



Slika 5. Stanica manipulatora - bočni pogled

U sklopu obnove stanice cilindri neće biti zamjenjivani novima, međutim priključci za vodove na njima hoće. Svaki konektor na stanici, koji ujedno služi kao prigušnica zraka, spaja se na vodove promjera 6 mm.

3.2. Komponente za zamjenu

U uvodnom dijelu je spomenuto kako neke zastarjele dijelove, iako su i dalje funkcionalni, je isplativo zamijeniti modernim varijantama. U ovom poglavlju opisat će se dijelovi koji će dobiti zamjenu.

Programabilni logički kontroler Festo FPC 202

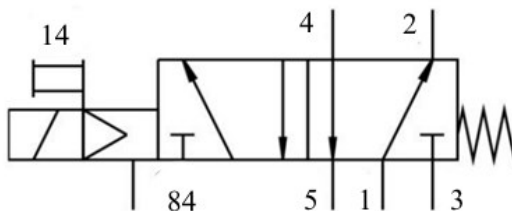
Na radnoj stanici je korišten Festo FPC 202 PLC koji datira iz 1980.-ih godina. Ima 16 ulaza, 8 tranzistorskih izlaza i 8 relejnih izlaza, a postoji mogućnost proširenja ulaza/izlaza uz spajanje drugog PLC-a. Na slici 6 se vidi da je takav pristup korišten. Za programiranje ovog PLC-a potrebno je računalo sa “FST200C” programom proizvođača Festo. Podržano je programiranje samo s Ladder dijagramima (LAD) i sa Statement list (STL) jezikom. PLC je mogao pohraniti do 8 programa u memoriji, a oni su se mijenjali uz pomoć tipki na prednjoj strani. [3]



Slika 6. Festo FPC 202 (gore) s proširenjem (dolje)

Pneumatski ventil MFH-5-1/8

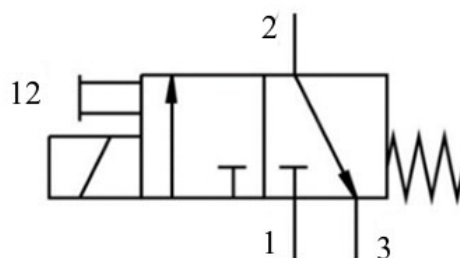
Ventil MFH-5-1/8 je monostabilni 5/2 ventil, električki aktuiran, nazivnog protoka 500 l/min s radnim tlakom u rasponu 1,8 do 8 bar [4]. Na slici 7 prikazan je blok od 9 takvih ventila. Za aktiviranje ventila koriste se elektromagnetske zavojnice MSFG-24/42-50/60-OD koje se moraju zasebno naručiti.



Slika 7. Blok MFH-5-1/8 ventila

Pneumatski ventil MFH-3-M5

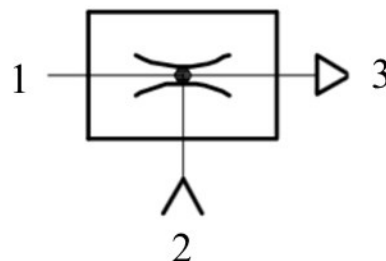
Neposredno pored prethodno spomenutih 9 ventila nalaze se dva MFH-3-M5 ventila. To su monostabilni 3/2 ventili, električki aktuirani sa nazivnim protokom u iznosu od 58 l/min [5]. Za rad ventila potrebno je zasebno nabaviti elektromagnetsku zavojnicu. Na ovoj stanici se koristila zavojnica MSFG-24/42-50/60-OD s radnim napon od 24 V [6].



Slika 8. Pneumatski 3/2 ventil s elektromagnetskom zavojnicom

Generator vakuuma VAD-1/4

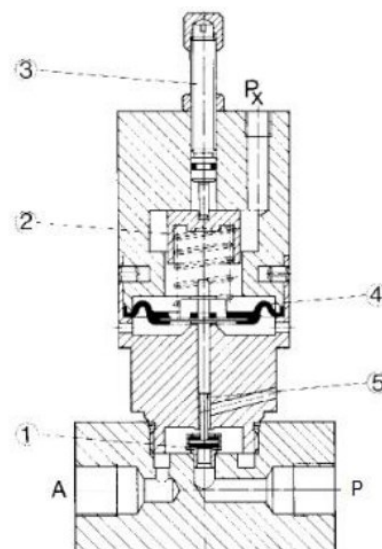
VAD-1/4 je kompaktni generator vakuuma koji djeluje prema Venturijevom principu nastajanja podtlaka. Oba generatora koriste prigušnice radi smanjena buke. Njihov radni tlak iznosi 1.5 - 10 bar i mogu stvoriti podtlak u iznosu do -1 bar [7].



Slika 9. Dva VAD-1/4 generatora vakuuma s prigušnicama

Glava za kontrolu vakuuma VUV

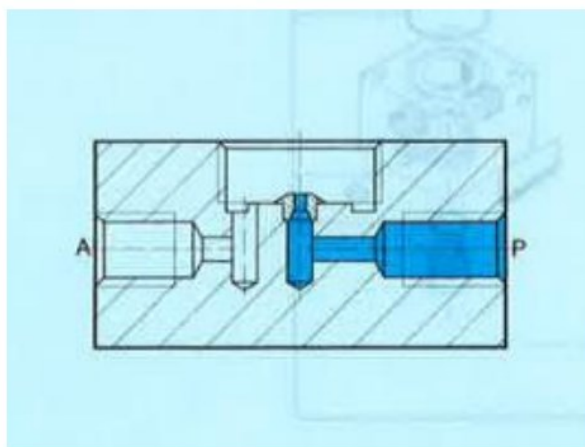
Glava za kontrolu vakuuma (njem. Vakuumschaltkopf) služi kao ventil koji se aktivira dovodom vakuumskeg signala. Uz pomoć vijka moguće je namjestiti silu opruge tj., potreban iznos vakuuma za aktivaciju.



Slika 10. Glava za kontrolu vakuuma [8]

Element LC-3-1/8

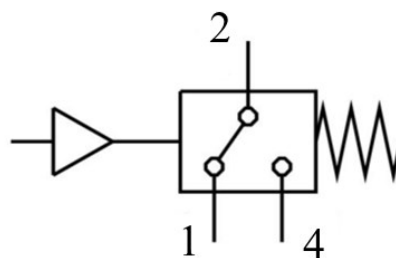
Funkcija ovog elementa ovisi o dodatku koji se koristi na njemu. U našem slučaju to je VUV glava za kontrolu vakuuma iz prethodnog poglavlja. Na slici 11 moguće je vidjeti ovaj dio bez dodataka.



Slika 11. Element LC-3-1/8 [9]

Tlačna sklopka PE-1/8-1N

Na slici 12 prikazana je tlačna sklopka koja za aktivaciju koristi zrak pod tlakom do 8 bar. U oznaci komponente „1N“ označen je tip šine na koju se spaja, a „1/8“ označava navoj priključka za crijevo zraka.



Slika 12. Tlačna sklopka PE-1/8-1N [9]

Senzor SME-1-LED-24

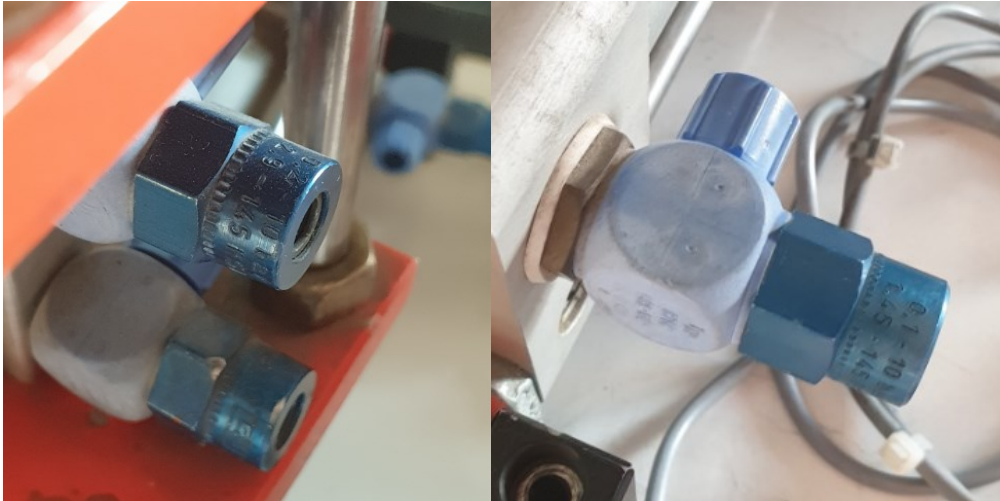
Ovaj senzor služi za detekciju položaja klipa cilindra. Prilikom detekcije, uključuje se LED lampica na senzoru. Nakon testiranja ispravnosti induktivnih senzora na cilindrima uspostavljeno je da su dva neispravna te će stoga biti naručen odgovarajući novi.



Slika 13. Induktivni senzori za poziciju klipa SME-1-LED-24

Raznovrsni prigušni ventili starog tipa

Na radnoj stanici koristili su se prigušni ventili različitih veličina, a zajedničko im je bilo to što se protok u jednom smjeru može regulirati. U svrhu pojednostavljenja, prilikom narudžbe novih prigušnih ventila pazilo se da, koliko god je moguće, budu iste veličine. Kod modernih varijanti pojednostavljen je način spajanja pneumatskih vodova time što više nije potrebno stezati navoj.



Slika 14. Prigušni ventil GRL-1/4-PK-6-KU

3.3. Nove komponente za ugradnju

Upravljački uređaj CONTROLLINO MAXI Automation

Za potrebe ovog projekta odabran je CONTROLLINO industrijsko računalo. CONTROLLINO je industrijski PLC u potpunosti kompatibilan s Arduinoom. Osnovna značajka ovog proizvoda je kombinacija pouzdanosti PLC-a i fleksibilnost i otvorenost Arduino ekosustava. Proizvod je austrijske tvrtke CONELCOM GmbH sa sjedištem u Innsbruck-u. U svojoj ponudi nude 3 linije proizvoda:

- CONTROLLINO MINI
- CONTROLLINO MAXI
- CONTROLLINO MEGA



Slika 15. Upravljački uređaji: CONTROLLINO MINI, MAXI i MEGA [10]

CONTROLLINO je moguće programirati u bilo kojem okruženju koje podržava Arduino.

Najpoznatija razvojna okruženja za Arduino su:

- Arduino IDE
- Atmel Studio
- Visual Studio (uz proširenje VisualMicro)
- Eclipse for Arduino
- EmbedXcode (sa MacOS sustav)

U sklopu ovog projekta koristit će se model „MAXI Automation“, jedini model koji ima analogne izlaze i ulaze. U Tablici 1. prikazana je usporedba glavnih karakteristika pojedinog CONTROLLINO modela.

Tablica 1. Usporedba CONTROLLINO modela [11]

	MINI	MAXI	MAXI Automation	MEGA
Analogno/digitalni ulazi	6	10	12	16
Isključivo digitalni ulazi	2	2	6	5
Analogni 0-10V ulazi	-	-	2	2
Digital 2A izlazi	8	12	8	24
Analogni 0-10V izlazi	-	-	2	-
Releji	6	10	10	16
Serijski TTL protokol	DA	DA	DA	DA
I ² C protokol	DA	DA	DA	DA
RS485 protokol	NE	DA	NE	DA
Ethernet podrška	NE	DA	DA	DA

Ethernet veza omogućit će nam implementaciju razvijenih web-aplikacija npr., daljinsko upravljanje i očitavanje senzora. Deset galvanski izoliranih relejnih izlaza prikladni su za veća opterećenja ili za situacije gdje se koriste uređaji koji rade s 230 V izmjenične struje.

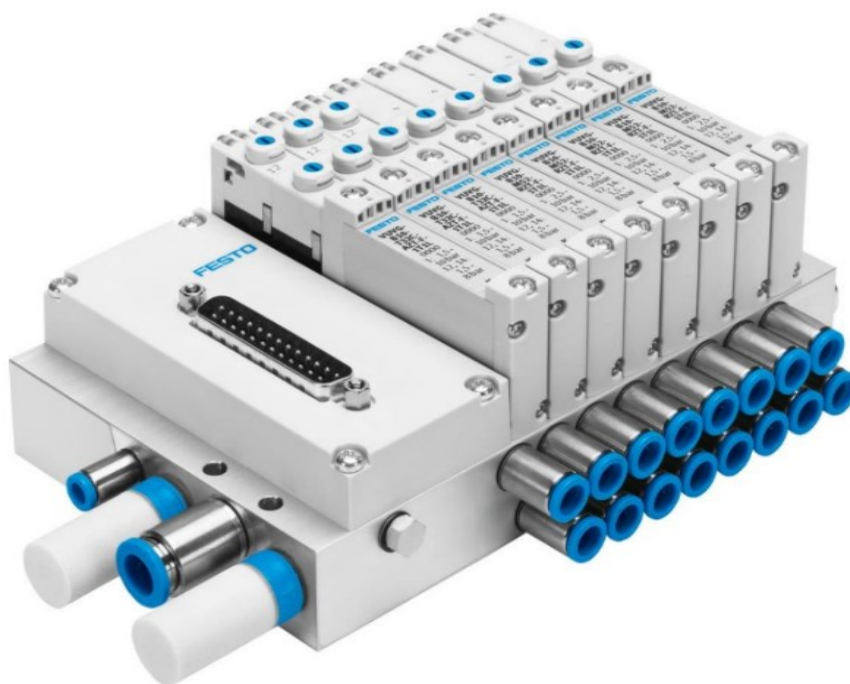


Slika 16. Upravljački uređaj CONTROLLINO MAXI Automation [12]

Ventilski razvodnik VTUG

Ventilski razvodnik VTUG podržava rad do 24 ventila s mogućnošću odabira električne veze. U ovom radu koristit će se multipolski konektor kao na slici 17, no moguće je naručiti model s I/O link-om, CANopen-om, PROFIBUS-om itd. Maksimalni protok za model koji će se koristiti u ovom projektu jest 330 l/min, a nominalno on iznosi 130 l/min -1150 l/min, ovisno o varijanti modela. Razvodnik radi s tlakom u rasponu od -0,9 do 10 bar [13].

U domeni ovog projekta, svi pneumatski ventili koji će se koristiti bit će ugrađeni na jednom ventilskom bloku. Usporedno s trenutnim načinom kako je stanica konstruirana, prednosti koje ćemo s tim dobiti su kompaktnost i organiziranost ožičenja.

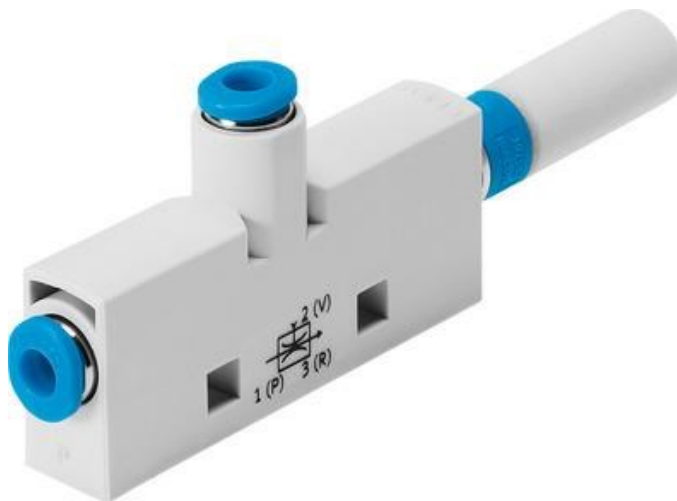


Slika 17. Primjer VTUG ventilskog razvodnika s 8 ventila [13]

Potpuni naziv modela koji će se naručiti glasi: VTUG-10-MSDRB1T-25V20-Q8L-UL-Q6S-11AK-M1. Razvodnik koji će biti korišten na stanici imat će jedanaest 5/2 monostabilnih ventila i dva 3/2 monostabilna ventila. Devet od jedanaest 5/2 ventila mijenjat će stare ventile iz prethodnog poglavlja, a ostala dva će se koristiti kod upravljanja protokom za vakuumske hvataljke.

Generator vakuuma VN-05-H-T3-PQ2-VQ2-RO1

S ovom komponentom zamjenjujemo stari generator vakuuma opisanog u prethodnom poglavlju. Radi s ulaznim tlakom u rasponu 1–8 bar i generira maksimalan vakuum u iznosu od -0,9 bar. Još jedna razlika naspram starog generatora vakuuma, osim u veličini, jest u tome što se ovdje koriste utični priključci za vodove promjera 6 mm [14].



Slika 18. Generator vakuuma VN-05-H-T3-PQ2-VQ2-RO1 [14]

Senzor vakuuma SDE5-V1-FP-Q6-P-M8

S ovim senzorom dobivamo povratnu informaciju o tome da li je predmet prihvaćen s vakuumskom hvataljkom. Na senzoru postoji tipka s kojom se može programirati modul rada. Za konekciju se koristi 3-pinski kabel (puno ime: NEBU-M8G3-K-5-LE3) kao na slici 19.



Slika 19. Senzor vakuuma i njegov konektor [15]

Vakuumska hvataljka VAS-15-1/8-NBR

Stare i dotrajale vakuumske hvataljke potrebno je zamijeniti novima. Dimenzije novih su iste, a jedina razlika je u materijalu izrade. Novi model je izrađen od nitrilne gume (skraćeno: NBR).



Slika 20. Vakuumska hvataljka [16]

Induktivni senzor SME-8M-DS-24V-K-2,5-OE

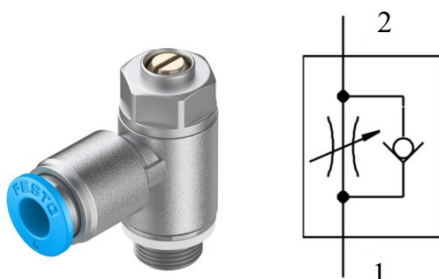
Kako dva induktivna senzora ne rade ispravno, potrebno je naručiti nove. Uz njega naručen je i montažni komplet SMBR-8-16 (slika 21, desno).



Slika 21. Induktivni senzor i njegov držač [17]

Prigušno nepovratni ventili

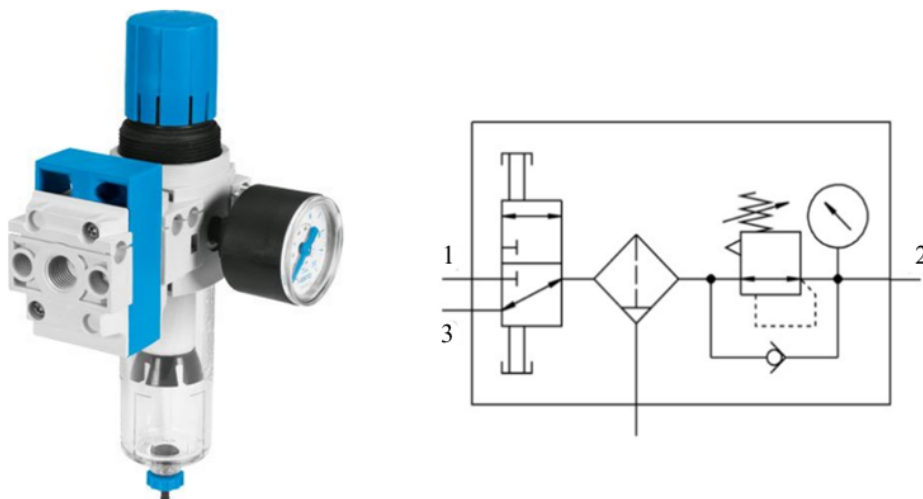
Postojeći ventili na radnoj stanici koriste zastarjele načine spajanja vodova i uz to svaki od njih prima različite promjere crijeva. S ovom narudžbom riješit će se oba problema. Naručeni su modeli: GRLA-1/8-QS-6-D, GRLA-M5-QS-6-D, GRLA-3/8-QS-6-D.



Slika 22. Novi prigušno nepovratni ventil [18]

Pripremna grupa LFR-1/4-DB-7-MINI-KC

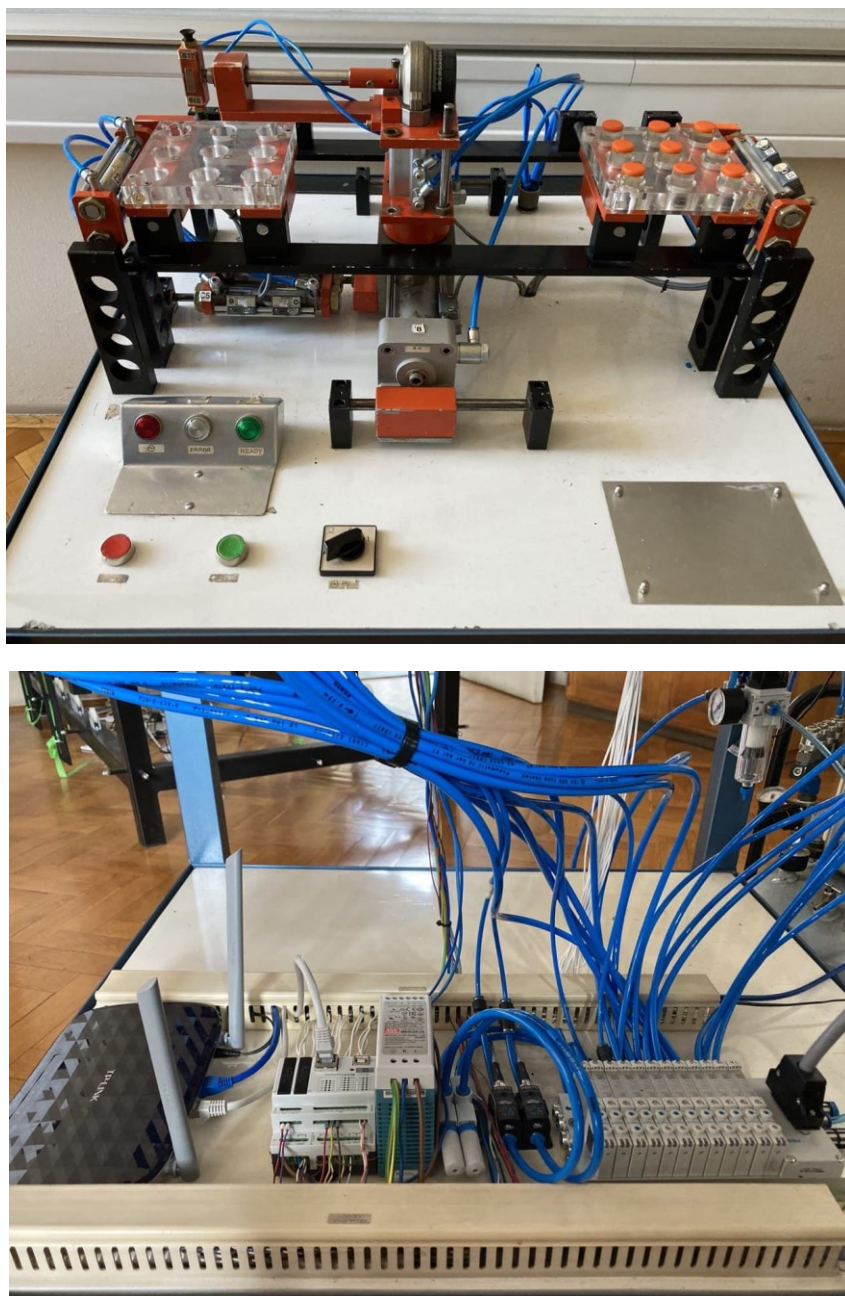
Pripremna grupa za zrak je kombinacija filtra, zauljivača i regulatora u jednom. Moguće je regulirati tlak zraka u rasponu od 0,5 do 7 bar-a. Dotok zraka se aktivira i deaktivira ručno, a nazivni protok iznosi 1900 l/min. [19]



Slika 23. Pripremna grupa LFR-1/4-DB-7-MINI-KC [19]

3.4. Stanje nakon ugradnje novih komponenti

Na slici 24 prikazano je stanje na radnoj stanici nakon što se ugrade novi dijelovi. Na slici se još vidi 24V napajanje i ruter o kojem će se kasnije biti riječ.



Slika 24. Stanje radne stanice nakon ugradnje novih komponenti

4. PROTOKOL ZA PODATKOVNU KOMUNIKACIJU MODBUS TCP/IP

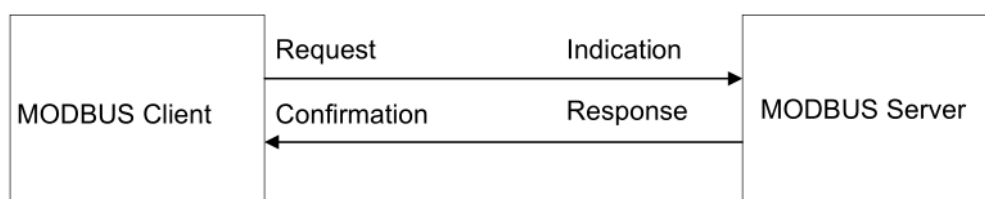
Modbus je komunikacijski protokol za prijenos informacija između elektroničkih uređaja putem serijske veze ili preko Etherneta. U upotrebi je više od 40 godina, a svoj uspjeh i rasprostranjenost duguje ponajviše svojoj jednostavnosti, robusnosti i činjenici da je besplatan za korištenje. Svoju primjenu pronalazi ponajviše u području industrijske automatizacije.

Cijela komunikacija temelji se na Master/Slave odnosu između uređaja. U takvom odnosu komunikacija se uvijek odvija u paru - jedan uređaj inicira zahtjev i čeka odgovor. Modbus uređaj koji inicira zahtjeve zove se *Master* (ili *Server* kod Modbus TCP/IP) i on je odgovoran za započinjanje svake interakcije.

Postoje četiri tipa poruka koji se mogu izmijeniti unutar Modbus komunikacije:

- MODBUS Request - poruka koju šalje klijent da bi inicirao izmjenu
- MODBUS Response - odgovor koji šalje server (npr. glavni PLC) klijentu
- MODBUS Indication - poruka koju je primio server
- MODBUS Confirmation - poruka koju je primio klijent

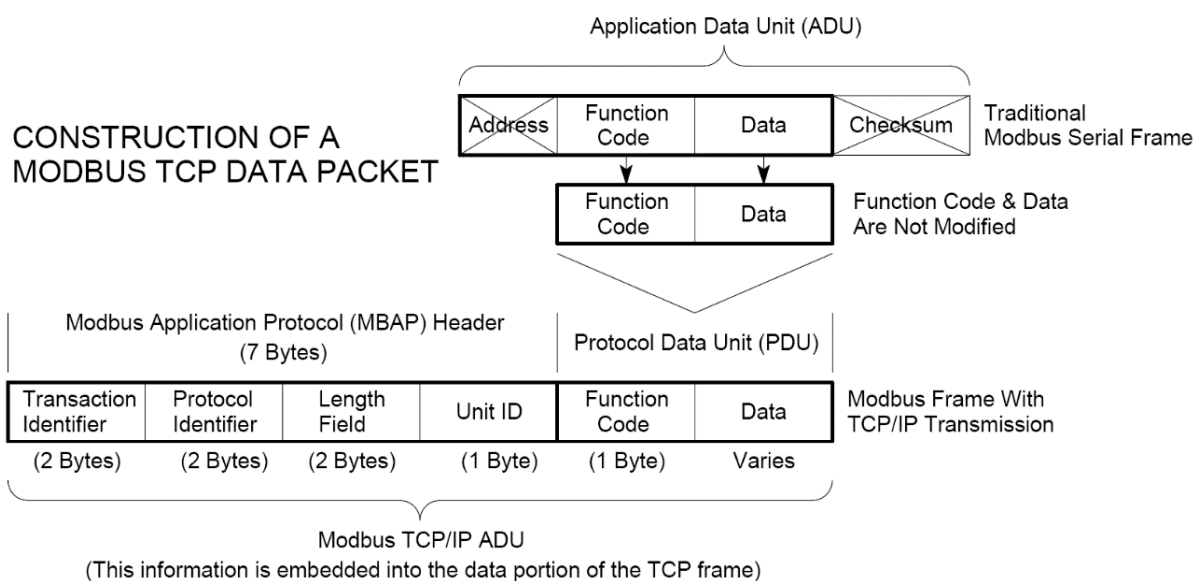
Redoslijed izmjena poruka prikazan je vizualno na slici 25:



Slika 25. Modbus *Client/Server* izmjena poruka [20]

Modbus je izvorno kreiran kao jedan protokol namijenjen samo za serijske veze. Kako je rasla potražnja za Modbusom, javila se potreba za promjenom u jezgri protokola koja bi omogućila implementaciju ostalih komunikacijskih veza. Stoga sad postoje dva sloja u Modbus poruci: Protocol data unit (PDU) i, onaj mrežni sloj, Application data unit (ADU). PDU se sastoji od 1-bitnog funkcijskog koda koji je popraćen sa 252-bitnim podatkom. ADU dio se odnosi na cjelokupnu poruku tj. to je PDU koji u zaglavlju još ima i dio za TCP/IP komunikaciju. Razlika u strukturi ADU-a naspram serijskog Modbus RTU (engl. Remote Terminal Unit) je u tome što

je izbačen dio za provjeru integriteta poruke i u tome što se za adresiranje sad koriste IP adrese, kao što je prikazano na slici 26.



Slika 26. Struktura jednog Modbus TCP/IP paketa [21]

Svaki Modbus *Slave/Client* uređaj koristi tablice kako bi upravljao ulazno/izlaznim podacima. Podaci se mogu spremati u jedan od četiri moguća tipa tablica kao što je to prikazano u tablici 2. Na jednoj adresi stane 2 bajta podataka, točnije jedan pozitivan decimalni broj u rasponu od 0 do 65536. U slučaju da želimo poslati više od 2 bajta podataka, npr. neki broj puno veći od 65536, potrebno je zauzeti dvije adrese za taj podatak.

Tablica 2. Tipovi Modbus tablica

Ime tablice	Tip	Adresni rang	Veličina
Diskretni izlazi	Čitanje/pisanje	00001 - 09999	1 bit
Diskretni ulazi	Samo čitanje	10001 - 19999	1 bit
Analogni ulazni registri	Samo čitanje	30001 - 39999	16 bita
Analogni izlazni registri	Čitanje/pisanje	40001 - 49999	16 bita

Da bi se manipuliralo podacima u tablici potrebno je najprije *Slave/Client* uređaju poslati neki od funkcijskih kodova. Definicija svakog funkcijskog koda određena je standardom. Tablica 3. prikazuje najčešće korištene funkcijske kodove.

Tablica 3. Najbitniji funkcijski kodovi

Tip funkcije		Ime funkcije	Kod
1-bit pristup	Fizički diskretni ulaz	Čitanje statusa ulaza	2
	Interni bitovi	Čitanje statusa bita	1
		Postavljanje jednog bita	5
		Postavljanje više bitova	15
16-bit pristup	Fizički ulazni registri	Čitanje ulaznog registra	4
	Interni registri	Čitanje <i>Holding</i> registra	3
		Postavljanje jednog registra	6
		Postavljanje više registara odjednom	16

U sklopu ovog diplomskog rada korištene su samo funkcije za postavljanje i čitanje *holding* registara, dakle funkcijski kodovi 3 i 6.

TCP (engl. Transmission Control Protocol) je komunikacijski standard koji omogućuje izmjenu podataka između uređaja na mreži. TCP definira da se, nakon uspostave konekcije između servera i klijenta, poruka najprije podijeli u manje pakete kako bi se izbjegla situacija u kojoj se cijela poruka mora ispočetka poslati u slučaju greške u prijenosu. IP (engl. Internet Protocol) protokol se često spominje zajedno uz TCP. On je odgovoran za ispravno adresiranje poruka i usmjeravanje poruka između uređaja.

TCP i IP su protokoli za prijenos podataka, međutim oni ne određuju kako će se ti isti podaci interpretirati. To je zadatak aplikacijskog protokola, u našem slučaju Modbus-a. Stoga, Modbus TCP/IP koristi fizičku mrežu (Ethernet) da bi, zajedno sa mrežnim standardom (TCP/IP), enkapsulirao Modbus komunikaciju.

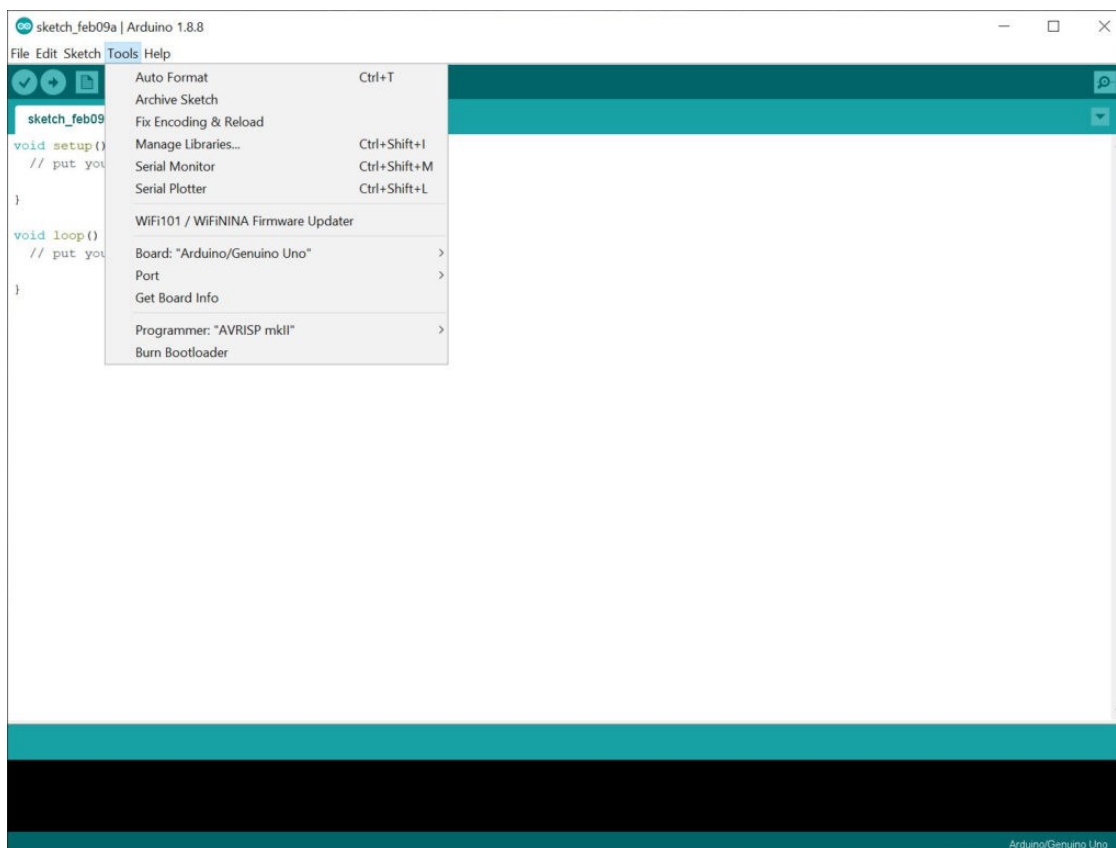
5. UPRAVLJAČKI PROGRAM PNEUMATSKE RADNE STANICE

U ovom poglavlju obradit će se izrada programa koji će izvoditi na CONTROLLINO PLC-u.

5.1. Arduino programski jezik

CONTROLLINO PLC, kao što je već spomenuto, se programira u Arduino jeziku koji se temelji na C++ jeziku. Koristi se ograničen broj standardnih C++ biblioteka tako da bi program bio kompatibilan s Arduino uređajima koji sami po sebi imaju malo radne memorije i ograničene procesorske mogućnosti.

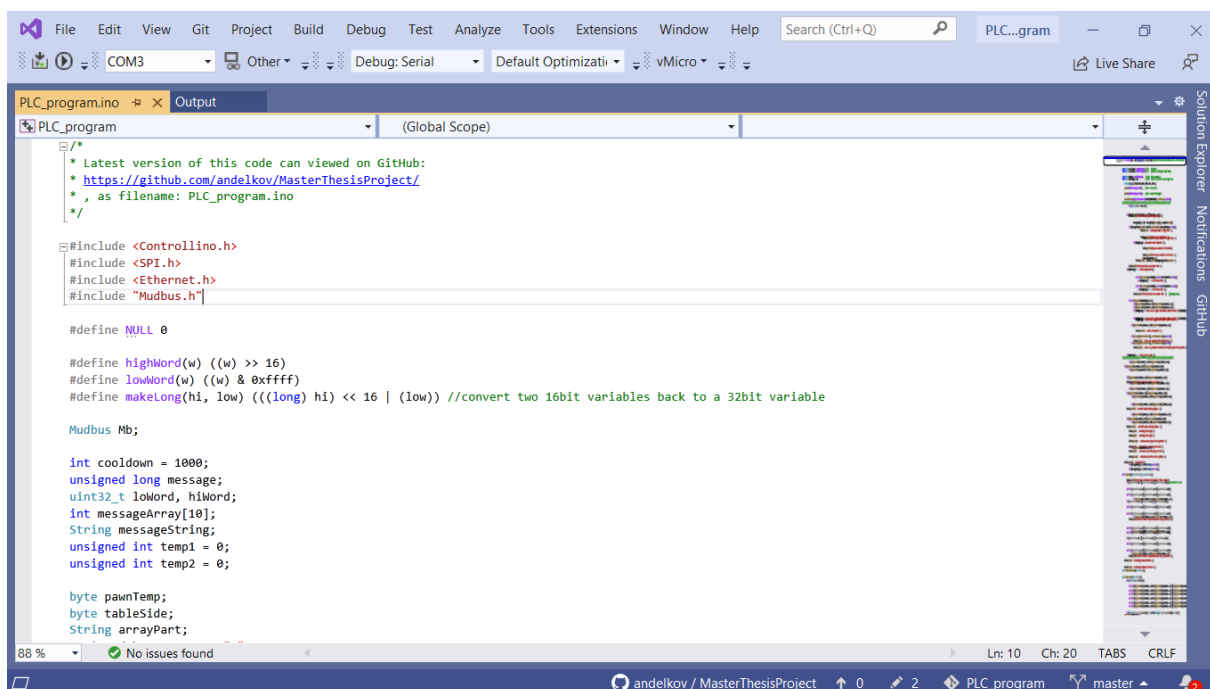
Arduino IDE (engl. Integrated Development Enviroment) je program otvorenog kôda koji služi za pisanje i učitavanje kôda na Arduino kompatibilnim pločicama. Navedeno sučelje je više nego dovoljno opremljeno za pisanje jednostavnih programskih rješenja. Međutim, kako složenost programa raste, javlja se potreba za korištenjem naprednijih razvojnih okruženja koji nude dodatne mogućnosti.



Slika 27. Arduino IDE sučelje

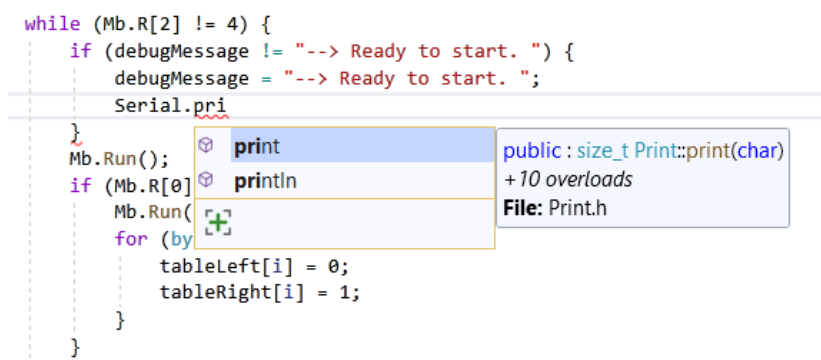
5.2. Razvojno okruženje Visual Studio 2019

Programski kôd stanice napisan je u programu Visual Studio 2019. Proizveden i održavan od strane Microsoft-a, Microsoft Visual Studio ili kraće VS je integrirano razvojno okruženje koje se koristi za razvoj računalnih programa, web stranica, web aplikacija, web servisa, mobilnih aplikacija itd. Sučelje nije vezano za jedan jezik kao što je to npr. Arduino IDE (engl. Integrated Development Enviroment) pa se stoga može koristiti za pisanje kôda u C, C++, C#, Python, JavaScript i za mnoge druge programske jezike.



Slika 28. Visual Studio 2019 sučelje

Visual Studio, kao i većina naprednih IDE-a, ima ugrađeni alat za inteligentno dovršavanje kôda imena *IntelliSense*. Dok unosimo kod, *IntelliSense* pokušava preko različitih algoritama predvidjeti što želimo upisati i potom nam nudi prijedloge u iskočnom prozoru (slika 29). Time se značajno smanjuju tipografske i sintaksne pogreške prilikom programiranja.



Slika 29. *IntelliSense* prijedlog prilikom tipkanja

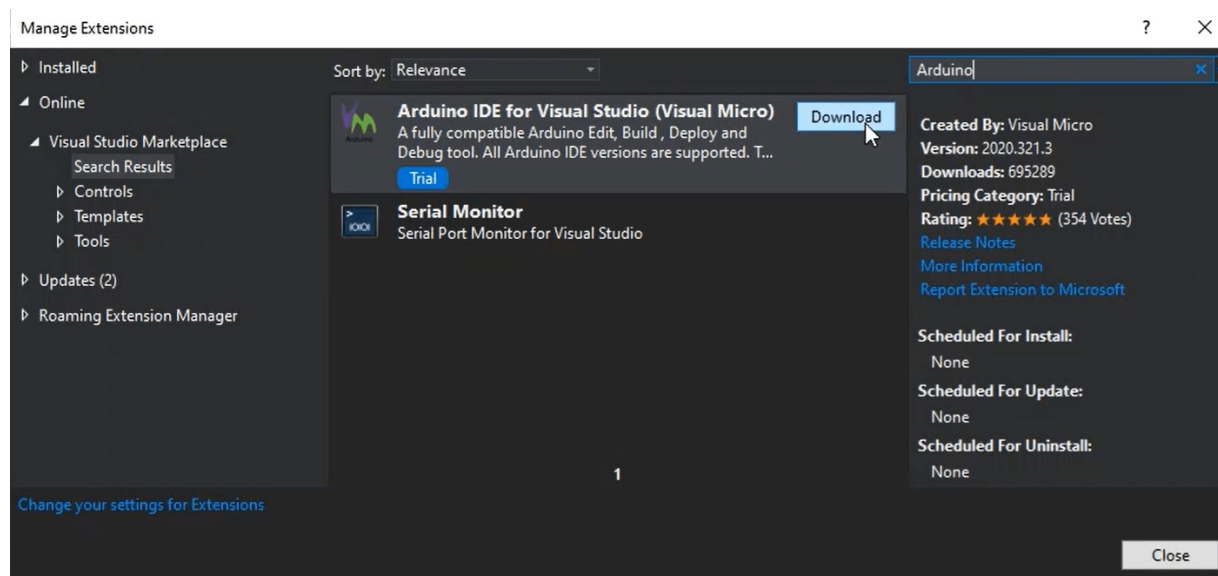
Nadalje, ako se program u bilo kojem trenutku ne izvodi očekivano, moguće je iznova pokrenuti program u tzv. *debugging* način rada. U tom načinu rada moguće je npr. pauzirati program kada izvršavanje dođe do određene linije koda te na taj način ući u trag *bug*-u. Također, prilikom izrade rada korišten je *git* sustav kontrole verzije preko proširenja „GitHub“ kojeg je potrebno zasebno instalirati.

Ove i mnoge druge mogućnosti nisu dostupne u programu Arduino IDE. Program Visual Studio sam po sebi ne podržava Arduino jezik, stoga je potrebno instalirati dodatak Visual Micro. Osim što s ovim proširenjem dobijemo podršku pisanja za Arduino jezik, taj dodatak je neophodan ako želimo kôd kompilirati i poslati na PLC ili Arduino pločicu.

5.3. Konfiguriranje razvojnog okruženja

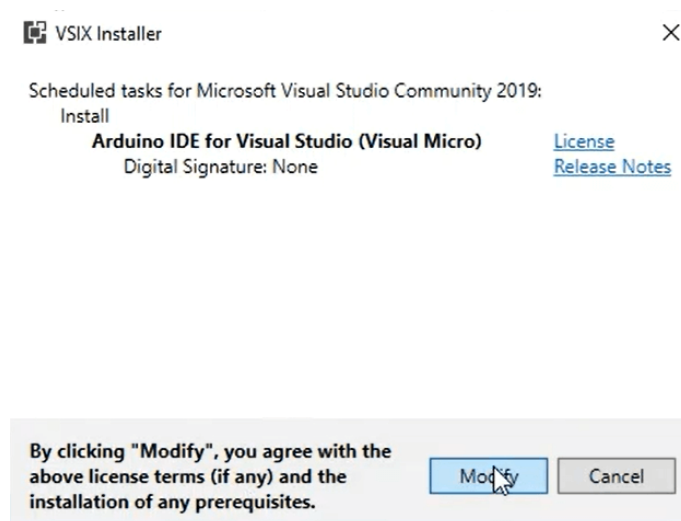
Potrebno je najprije program Visual Studio 2019 preuzeti s Interneta. Na raspolaganju se nude tri inačice: Visual Studio 2019 *Community*, *Professional* i *Enterprise*. *Community* verzija je dovoljno opremljena značajkama za potrebe ovog rada i besplatna je ako se prijavimo s Microsoft računom.

Nakon instalacije, potrebno je preuzeti dodatak VisualMicro. Jedan od lakših načina za preuzeti taj dodatak jest da koristimo upravitelj za proširenja unutar programa. To činimo tako da na glavnoj izbornoj traci odaberemo: *Extensions > Manage Extension*, u tražilicu unesemo „Arduino“ i potom odaberemo „Arduino IDE for Visual Studio (Visual Micro)“.



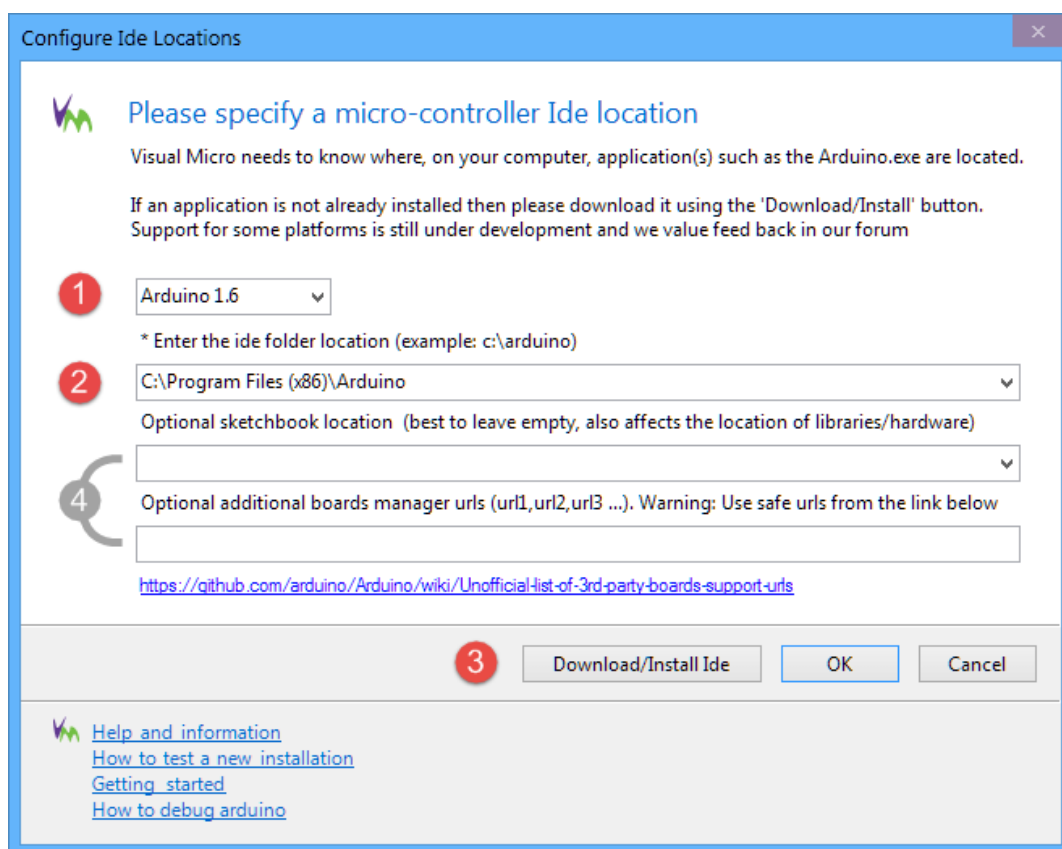
Slika 30. Preuzimanje proširenja Visual Micro

Klikom na *Download* dočekać će nas prozor kao na slici 31. Instalaciju možemo započeti odabirom *Modify*. Nakon uspješne instalacije potrebno je ponovno pokrenuti Visual Studio.



Slika 31. Instaliranje dodatka Visual Micro

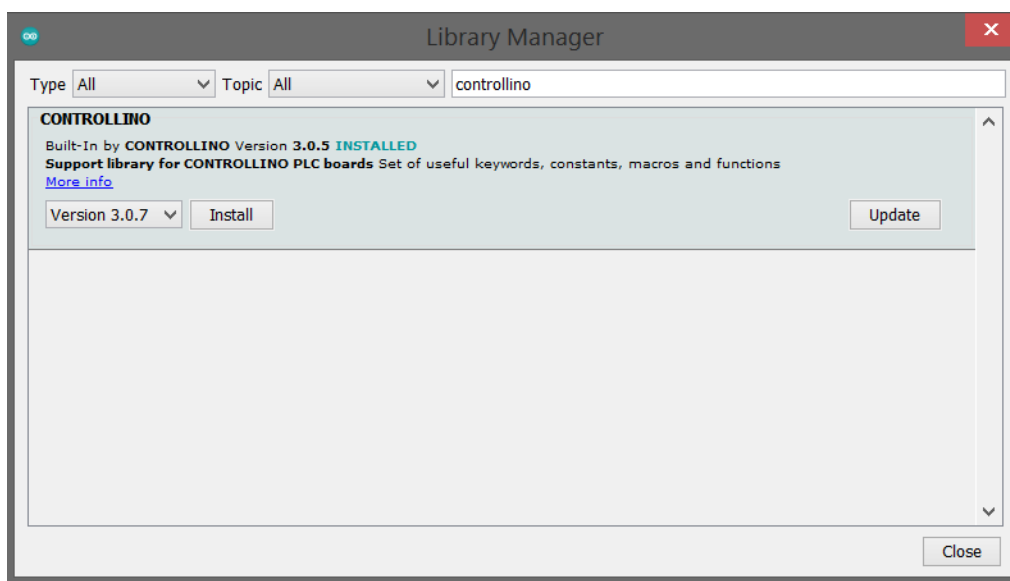
Prilikom prvog pokretanja nakon instalacije dočekać će nas prozor *Configuration Manager* gdje je potrebno definirati mapu u kojoj se nalazi instalacija Arduino IDE-a. Ako on ne postoji, moguće ga je preuzeti i instalirati odabirom tipke *Download/Install Ide*.



Slika 32. Definiranje Arduino IDE instalacijske mape

Polje *Optional sketchbook location* definira adresu mape u kojoj će Visual Micro potražiti korisničke biblioteke. Ako se ovo polje ostavi prazno, Visual Micro će koristiti istu lokaciju kao što ju koristi Arduino IDE.

Da bi mogli programirati PLC potrebno je najprije instalirati službene CONTROLLINO biblioteke za Arduino. Ovdje će se pokazati instalacija preko programa Arduino IDE. Nakon pokretanja programa, u glavnoj izbornoj traci potrebno je odabrati *Sketch>Include Library>Manage Libraries*. Pojavit će se prozor kao na slici 33. U tražilicu upišemo „controllino“ i odabiremo *Install* na prvom rezultatu.



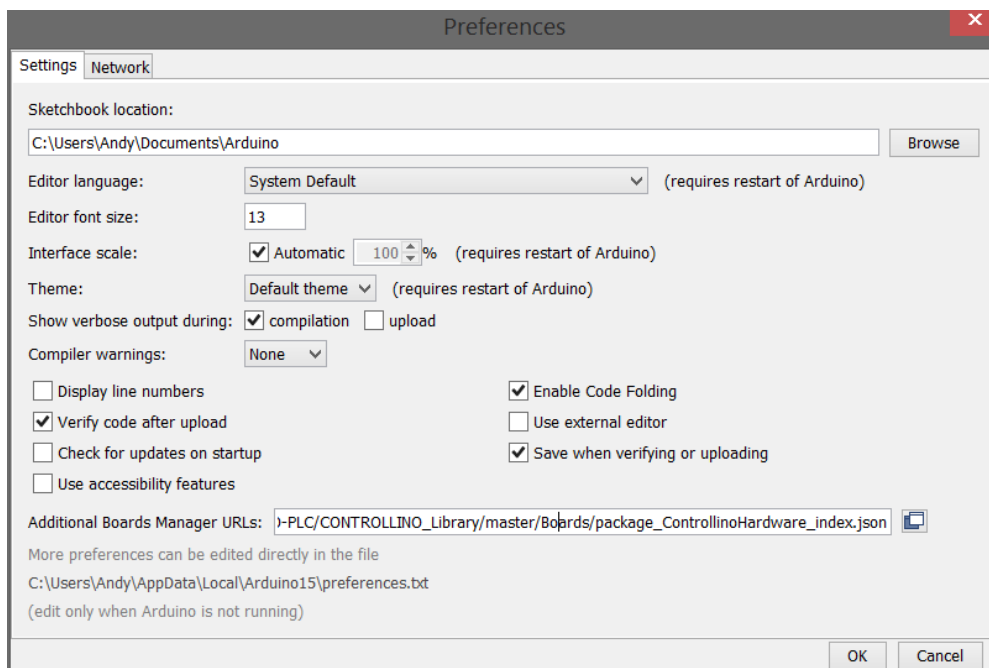
Slika 33. Instaliranje CONTROLLINO biblioteka

Uspješnost instalacije možemo provjeriti tako da u glavnoj alatnoj traci pod *Sketch>Include* vidimo da li postoji biblioteka CONTROLLINO. Na isti način potrebno je provjeriti da je instalirana biblioteka „Ethernet“ i „SPI“. Za implementaciju Modbus TCP/IP protokola korištena je biblioteka „Mudbus.h“, koja se ručno instalira tako da se kopira u mapu *libraries* koja se nalazi unutar instalacijske mape Arduina, a ona se može preuzeti s poveznice:

<https://github.com/luizcantoni/mudbus>

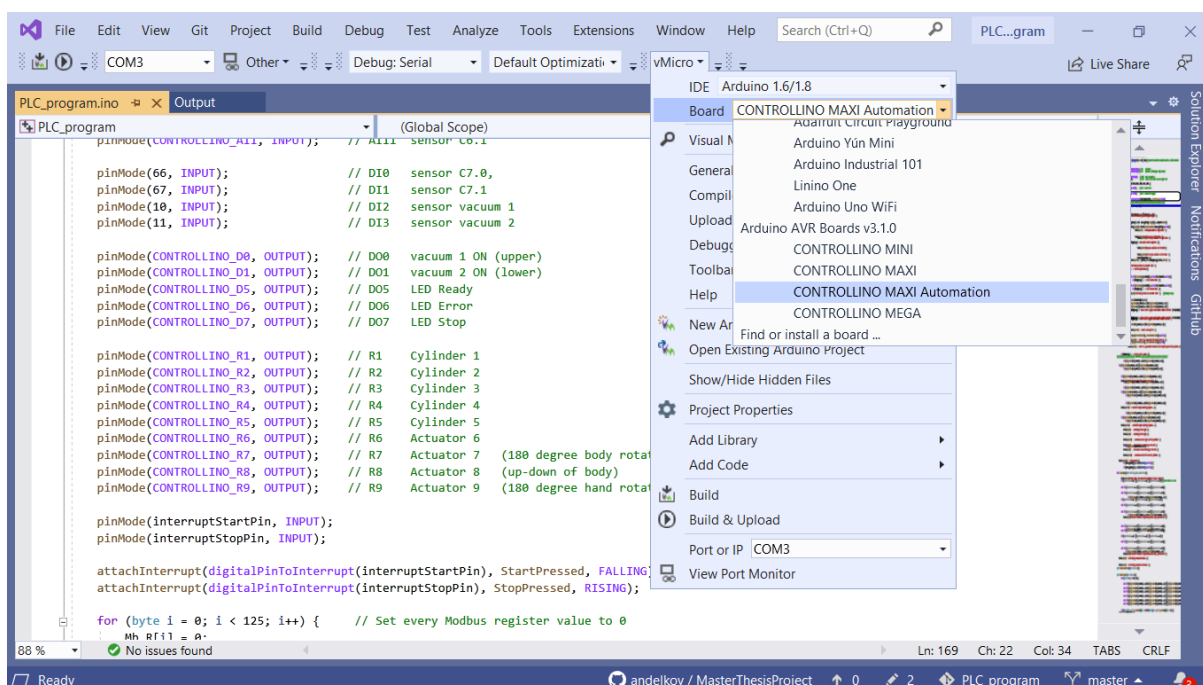
U posljednjem koraku konfiguracije potrebno je još da u postavkama navedemo web poveznicu koja nosi dodatne informacije o CONTROLLINO pločama. Poveznica se unosi u postavkama pod *File>Preferences* u polju *Additional Boards Manager URLs* kao što je prikazano na slici 34. Poveznica koju je potrebno unijeti glasi [22]:

https://raw.githubusercontent.com/CONTROLLINO-PLC/CONTROLLINO_Library/master/Boards/package_ControllinoHardware_index.json



Slika 34. Unos web poveznice u Arduino IDE-u

CONTROLLINO PLC, kao i bilo koja druga Arduino pločica, se s računalom može povezati pomoću USB kabela. Nakon povezivanja još preostaje odabrati *port* i vrstu uređaja kojeg programiramo pod *Extensions>vMicro>Board* i pod *Extensions>vMicro>Port or IP* kao što je prikazano na slici 35.



Slika 35. Instaliranje CONTROLLINO biblioteka

5.4. Struktura programa

Programski kôd radne stanice može se podijeliti na četiri dijela:

- U **prvom dijelu** navedene su globalne varijable, definicije konstanti i biblioteke koje se uvoze.
- U **drugom dijelu**, pod funkcijom *setup*, izvršava se inicijalizacija. Ovdje se definiraju mrežne postavke za PLC, uspostavlja serijska komunikacija i definiraju ulazni/izlazni pinovi. Ovaj dio kôda izvršava se jedanput i to prilikom pokretanja PLC-a.
- U **trećem dijelu**, pod funkcijom *loop*, nalazi se dio kôda koji će se izvršavati u petlji dokle god je PLC uključen.
- U **četvrtom dijelu**, ispod funkcije *loop*, nalaze se funkcije koje će se pozivati unutar *loop*-a. Pisanjem funkcija značajno skraćujemo kôd i olakšavamo njegovo čitanje.

5.4.1. Inicijalizacijski dio programa

U prvom dijelu pozivamo biblioteke sa ključnom riječi *#include* kao u primjeru na slici 36.

```
#include <Controllino.h>
#include <SPI.h>
#include <Ethernet.h>
#include "Mudbus.h"

#define NULL 0

#define highWord(w) ((w) >> 16)
#define lowWord(w) ((w) & 0xffff)
#define makeLong(hi, low) (((long) hi) << 16 | (low)) //convert two 16bit variables back to a 32bit variable

Mudbus Mb;
```

Slika 36. Uvodni dio kôda radne stanice

Biblioteka *Controllino.h* olakšava posao pisanja kôda time što za pozivanje pinova je potrebno samo koristiti ključne riječi, umjesto da se pamti broj za svaki pin. Primjer korištenja bit će vidljiv kasnije. Biblioteka *SPI.h* (skraćeno Serial Peripheral Interface) omogućuje komunikaciju s perifernim uređajima. U našem slučaju ova biblioteka je potrebna da bi mogli koristiti Ethernet modul na CONTROLLINO PLC-u. *Ethernet.h* biblioteka omogućava spajanje Arduina s Internetom. U ovom radu PLC će biti konfiguriran sa statičnom IP adresom, mada je moguće postaviti da se Arduino spaja u lokalnu mrežu s DHCP-om. Biblioteka *Mudbus.h* implementira Modbus TCP/IP protokol za *Slave/Client* uređaje.

Ispod biblioteka, sa ključnom riječi *#define* odredili smo da riječ *NULL* označava brojčanu vrijednost nula i ispod toga još tri, uvjetno rečeno, funkcije. S tim „funkcijama“ moguće je zaobići jedno ograničenje koje nosi Modbus, a to je činjenica da je on dizajniran da prenosi 16-bitne podatke. Jedan od načina da se doskoči problemu jest da se 32-bitni broj najprije podjeli na dva 16-bitna broja i da se oni pošalju u dva registra. Program će pročitati te registre, te uz pomoć *bitwise* operacija rekonstruirati originalni broj.

Nadalje, sa dva *array*-a spremamo aktualne pozicije valjčića na stolovima. Svaki stol ima po 9 pozicija, ali odabran je *array* s 10 članova tako da bi se olakšalo čitanje koda pošto numeriranje članova počinje od 0 (nula). Iz tog razloga je dodan *NULL* na prvom mjestu, odnosno 0. Ti *array*-i, kao i varijable ispod njih, globalne su varijable što znači da one mogu pozvati i modificirati u bilo kojem dijelu programa.

U drugom dijelu programa, unutar vitičastih zagrada ispred *setup()* stavlja se kôd koji će se izvoditi jedanput, prilikom pokretanja Arduina. U ovom dijelu konfigurišu se Internet postavke. MAC adresa je jedinstveni identifikacijski broj uređaja na mreži. Nije neuobičajeno da se ona definira od strane korisnika programatski kao što je to pokazano na slici 37. Ispod toga definira se IP adresa uređaja. Radna stanica će imati svoju statičnu IP adresu, međutim nju će koristiti ruter koji će potom promet preusmjeravati na lokalnu IP adresu PLC koju smo definirali ovdje. *Gateway* je IP adresa čvorišne točke koja stoji na putu između jedne mreže i druge. Ovdje je definirana čvorišna točka jedne mreže na Fakultetu strojarstva i brodogradnje. Subnet adresu nije nužno unijeti, ali je korisna ako se podesi, pošto pomaže da poruka kraće putuje do mrežnog odredišta.

```
void setup() {  
    uint8_t mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };  
    uint8_t ip[] = { 192, 168, 0, 160 };  
    uint8_t gateway[] = { 161, 53, 116, 1 };  
    uint8_t subnet[] = { 255, 255, 252, 0 };  
    Ethernet.begin(mac, ip, subnet);  
  
    delay(2000);  
  
    Serial.begin(9600);  
    Serial.print("Started. ");  
    Serial.print("My IP address: ");  
    Serial.println(Ethernet.localIP());  
}
```

Slika 37. Isječak iz *setup()* dijela kôda

Unutar *setup()* je također potrebno explicitno definirati način rada za svaki *pin* koji ćemo koristiti. Umjesto da pišemo broj *pin*-a, koristit ćemo ključne riječi koje nudi **CONTROLINNO** biblioteka za Arduino pa tako, umjesto pisanja broja 29, može se napisati samo „**CONTROLLINO_R9**“.

Kako bi se mogle koristiti start i stop tipke na stanici, potrebno je koristiti *attachInterrupt()* funkciju. U slučaju da korisnik pritisne neku od tipki, program na glavnoj petlji se zaustavlja i nastavlja se izvoditi unutar funkcije koju smo sami definirali. Na slici 38 se vidi da su za to definirane funkcije *StartPressed* i *StopPressed*.

Zadnji korak unutar *setup()* jest postaviti sve Modbus adrese u nulu, osim registra 5 i 2. Registar 2 služi kao statusni registar i upisivanjem vrijednosti 2 u njega javljamo *Masteru/Serveru* da se stanica nalazi u pripravnom stanju. Registar 5 i 6 primaju naredbe o pozicioniranju valjića i, pošto vrijednost 0 (nula) ima značenje, ovdje su oni postavljeni u negativnu vrijednost.

```
pinMode(CONTROLLINO_R7, OUTPUT); // R7 Actuator 7 (180 degree body rotate)
pinMode(CONTROLLINO_R8, OUTPUT); // R8 Actuator 8 (up-down of body)
pinMode(CONTROLLINO_R9, OUTPUT); // R9 Actuator 9 (180 degree hand rotate)

pinMode(interruptStartPin, INPUT);
pinMode(interruptStopPin, INPUT);

attachInterrupt(digitalPinToInterrupt(interruptStartPin), StartPressed, FALLING);
attachInterrupt(digitalPinToInterrupt(interruptStopPin), StopPressed, RISING);

for (byte i = 0; i < 125; i++) { // Set every Modbus register value to 0
    Mb.R[i] = 0;
}
Mb.R[5] = -1; // For Auto mode
Mb.R[6] = -1; // For Auto mode
Mb.R[2] = 2; // Set status to "Power on"
Serial.println("Modbus registers set to 0.");
}
```

Slika 38. Isječak iz *setup()* dijela kôda

5.4.2. Izvršni dio programa

Program stranice ima tri modula rada:

- Automatski
- *Point-to-point*
- Ručno upravljanje

O svakom načinu rada bit će zasebno napisano kako funkcionira. Struktura izvršnog dijela programa prikazana je na slici 39.

```

//////////////////////////////// MAIN //////////////////////////////////
void loop() {
    while (Mb.R[2] != 4) {
        if (debugMessage != "--> Ready to start. ") {
            debugMessage = "--> Ready to start. ";
            Serial.println(debugMessage);
        }
        Mb.Run();
        if (Mb.R[0] == 2) { ... }

        digitalWrite(LED_Start, 0);
        delay(500);

        digitalWrite(LED_Start, 1);
        delay(500);
        Mb.Run();
    }

    Mb.Run();

    switch (Mb.R[3]) {
        case 1:
            ...
            break;
        case 2:
            ...
            break;
        case 3:
            ...
            break;
        default:
            if (modeMessage != "--> Please select a mode.") { ... }
            break;
    }
}

```

Slika 39. Pregled *loop()* petlje

Ulaskom programa u *loop()* prvo se ispiše poruka u terminalu. Program potom čeka da se vrijednost statusnog registra 2 promjeni u 4 što bi označavalo da program nije više u stanju mirovanja. Dok se ona ne promijeni, moguće je definirati koji stol je popunjen s valjčićima preko registra 0. Na mjestima gdje se kôd vrti u petlji, dodana je naredba „Mb.Run()“ koja služi za osvježavanje Modbus tablice i slanje povratnih poruka *Server*-u. Nakon izlaska iz stanja mirovanja, sljedeća naredba koju očekuje program jest odabir modula rada. Za to se koristi registar 3 i Arduino naredba *switch*.

5.4.2.1. Automatski način

Ideja ovog načina rada jest da se sa jednom naredbom od 9 binarnih znamenki definira raspored i brojučano stanje valjčića na pojedinom stolu. Odabrano je da naredba mora imati 9 znamenki pošto je to broj mjesta za valjčiće na svakom stolu. Broj „1“ označava popunjeno, a „0“ prazno

polje. Pa tako naredba „111000111“ govori da želimo da prvi i treći red na stolu budu popunjeni. Prije slanja te naredbe potrebno je najprije odabrati stol koji želimo popuniti ili isprazniti tako što ispunimo registar 7 s vrijednošću „1“ (lijevi stol) ili „2“ (desni stol).

Budući da za spremanje broja s 9 znamenki potrebna 32-bitna varijabla, zadaća je *Server/Master* uređaja da ispravno ispuni dva registra s 16-bitnim podacima. Na slici 40 prikazan je dio koji se bavi zaprimanjem 32-bitnog broja.

```
case 1:
    if (modeMessage != "--> Auto-mode selected.") {
        modeMessage = "--> Auto-mode selected.";
        Serial.println(modeMessage);
    }

    temp1 = Mb.R[6];
    temp2 = Mb.R[5];

    if ((Mb.R[6] >= 0) && ((Mb.R[7] == 1) || (Mb.R[7] == 2))) {

        message = makeLong(temp1, temp2);
        Serial.print("Received Mb.R[5] and Mb.R[6] messages: ");
        Serial.print(Mb.R[5]);
        Serial.print(", ");
        Serial.println(Mb.R[6]);
        Serial.print("Coververted Mb.R[5] and Mb.R[6] messages: ");
        Serial.print(temp1);
        Serial.print(", ");
        Serial.println(temp2);

        messageString = String(message);
```

Slika 40. Zaprimanje 32-bitnog broja

Nadalje, program provjerava je li zaprimljeni broj kraći od 10 znamenki i sadrži li samo binarne znamenke. Potom se, ovisno o odabranoj strani stola, izvršava algoritam za raspoređivanje valjčića za tu stranu. Na slici 41 prikazan je dio tog algoritma.

```
if ((messageString.length() < 10) && (onlyBinaryNumbers(messageString) == true)) {
    if (Mb.R[7] == 1) {
        Serial.println("--> Filling table 1. ");
        for (byte j = 1; j < 10; j++) {
            Mb.Run();
            Serial.print("--> Doing j number: ");
            Serial.println(j);
            if (messageArray[j] == 1) {

                if (tableLeft[j] != 1) {
                    pawnTemp = findAvailablePawn(2);
                    Serial.print("--> Found pawn number at right table: ");
                    Serial.println(pawnTemp);
                    if (pawnTemp == 0) {
                        Serial.println("No pawns available at right table");
                    }
                    else {
                        rotateRight();
                        goTo(2, pawnTemp);
                        pawnPickUp();
                        tableRight[pawnTemp] = 0;

                        rotateLeft();
                        goTo(1, j);

                        pawnDrop();
                        tableLeft[j] = 1;
                    }
                }
            }
            else if (messageArray[j] == 0) {
```

Slika 41. Isječak algoritma za automatsko slaganje valjčića

Algoritam se sastoji od *for* petlje koja broji od 1 do 9. Taj broj koristi se da bi se pristupilo vrijednostima u *array*-u. Ako je za zadanu iteraciju pronađena vrijednost „1“, algoritam će potražiti figuru na suprotnom stolu, uzet ju i potom ostaviti na traženi stol u broj pozicije jednak broju iteracije. Ako je za zadanu iteraciju pronađena vrijednost „0“, algoritam će potražiti prvo slobodno mjesto na drugom stolu i, ako ga ima, odnijet će valjčić tamo. Nakon što sve iteracije završe, algoritam će registre 5 i 6 postaviti u negativnu vrijednost tako da se modul zaustavi od ponovnog izvršavanja i čekat će sljedeći unos korisnika.

5.4.2.2. Point-to-point

U ovom modulu rada korisnik odabire jedan valjčić koji će biti prihvaćen i potom definira mjesto na kojem će se on ostaviti. Strana stola s kojeg će uzeti ili ostaviti valjčić se definira u registru 9. Ako se taj registar ostavi praznim, program će automatski definirat stranu stola preko senzora za poziciju stola. Koristeći Arduino petlju *do,while* spriječeno je da korisnik promijeni modul programa dok je valjčić u zraku (slika 42).

```
case 2:
    if (modeMessage != "--> Point-to-point mode.") {
        modeMessage = "--> Point-to-point mode.";
        Serial.println(modeMessage);
    }

    if (Mb.R[9] == 0) {
        Mb.R[9] = currentTableSide();
    }

    do {
        Mb.Run();

        if ((Mb.R[9] == 1) || (Mb.R[9] == 2)) {
            if ((Mb.R[10] >= 1) && (Mb.R[10] <= 9)) {
                if (Mb.R[9] == 1) { ... }
                else if (Mb.R[9] == 2) { ... }
            }
            Mb.R[10] = 0;
        }
    } while (isHandFull == true);
    break;
```

Slika 42. Struktura kôda drugog modula

5.4.2.3. Ručno upravljanje

U modulu ručnog upravljanja tj. *Jog mode* moguće je, kao što ime sugerira, manualno upravljati pokretima stolova i manipulatora. Naredbe za stol koje je moguće koristiti su pomicanje gore, dolje, lijevo ili desno. Naredbe za manipulator koje je moguće koristiti su rotacija tijela manipulatora lijevo ili desno, uzimanje i ispuštanje valjčića. Prije korištenja nekih od naredbi, potrebno je definirati koji stol želimo upravljati. Ako ovaj registar ostavimo praznim, algoritam će koristiti informaciju senzora pozicije manipulatora da bi odredio stranu. Slično kao i u prošlom modulu, *do*, *while* petlja zaustavlja korisnika da mijenja modul rada dok je valjčić u zraku.

```
case 3:
    if (modeMessage != "--> Jog mode.") {
        modeMessage = "--> Jog mode.";
        Serial.println(modeMessage);

        if (digitalRead(handIsLeft) == 1) {
            Mb.R[20] = 1;
        }
        else if (digitalRead(handIsRight) == 1) {
            Mb.R[20] = 2;
        }
    }

    if (Mb.R[20] == 0) {
        Mb.R[20] = currentTableSide();
    }
    do {
        Mb.Run();
        //go up
        if (Mb.R[22] == 1) { ... }
        //go down
        if (Mb.R[21] == 1) { ... }
        //go left
        if (Mb.R[23] == 1) { ... }
        //go right
        if (Mb.R[24] == 1) { ... }
        //rotate left or right
        if ((Mb.R[25] == 1) || (Mb.R[25] == 2)) { ... }
        // pick up the pawn
        if (Mb.R[26] == 1) { ... }
        // drop the pawn
        if (Mb.R[27] == 1) { ... }
    } while (isHandFull == true);
    break;
```

Slika 43. Struktura kôda trećeg modula rada programa

6. PROGRAMSKO RJEŠENJE ZA UPRAVLJANJE I NADZOR PREKO WEB-A

Za upravljanje sustavom koristit će se platforma *CyberHydraulic* kreirana od strane asistenta J. Benića. Također, ista platforma koristit će se i za video nadzor stanice. Prije nego se platforma može koristiti za upravljanje i nadzor, potrebno je najprije konfigurirati internet postavke na ruteru koji se nalazi na stanici.

6.1. Konfiguriranje Internet postavki rutera

Model rutera koji se koristi jest *TP-Link Archer C20 AC750*, iako je moguće koristiti praktički bilo koji ruter, jer ih većina ima osnovne funkcionalnosti koje se ovdje traže. Da bi se pristupilo postavkama rutera, potrebno je preko UTP kabla spojiti računalo na jedan od četiri *switch* porta na ruteru. Računalo treba u TCP/IP postavkama imati omogućeno automatsko dobavljanje IP adresa. Nakon uspješno dobivene IP adrese, moguće je spojiti se na ruter preko Internet pretraživača tako se u polje za URL adrese unese 192.168.0.1. Ako je sve u redu, dočekat će nas stranica za prijavljivanje gdje je potrebno unijeti korisničko ime i lozinku.



Slika 44. Ruter *TP-Link Archer C20 AC750* [23]

Za povezivanje na internet koristit će se statička IP adresa. Stoga prva stvar koju treba postaviti jest tip konekcije na ruteru. Pod *Network > WAN* u postavkama u padajućem izborniku *Connection Type*: odabiremo *Static IP* kao na slici 45. U poljima ispod unosimo statičku IP adresu, *gateway* i ostale unaprijed od fakulteta definirane adrese.

TP-LINK®

Status
Quick Setup
Network
- WAN
- LAN
- IPTV
- MAC Clone
Dual Band Selection
Wireless 2.4GHz
Wireless 5GHz
Guest Network
DHCP
USB Settings
NAT
Forwarding

WAN Settings

Connection Type: Static IP

IP Address: 161.53.117.212

Subnet Mask: 255.255.252.0

Gateway: 161.53.116.1

Primary DNS Server: 161.53.116.8

Secondary DNS Server: 161.53.2.70 (optional)

Slika 45. Postavljanje statičke IP adrese

Uređaji koji se povežu na ruter nalaze se unutar LAN mreže. IP adrese koje će im dodijeliti DHCP nalaze se u rasponu od 192.168.0.100 do 192.168.0.199. Kako bi osigurali da PLC i IP kamera svaki put dobiju istu IP adresu, potrebno je rezervirati adrese za njih. Na slici 46 prikazane su MAC adrese uređaja i IP adrese koje su vezane uz njih.

TP-LINK®

Status
Quick Setup
Network
Dual Band Selection
Wireless 2.4GHz
Wireless 5GHz
Guest Network
DHCP
- DHCP Settings
- DHCP Clients List
- Address Reservation
USB Settings
NAT
Forwarding

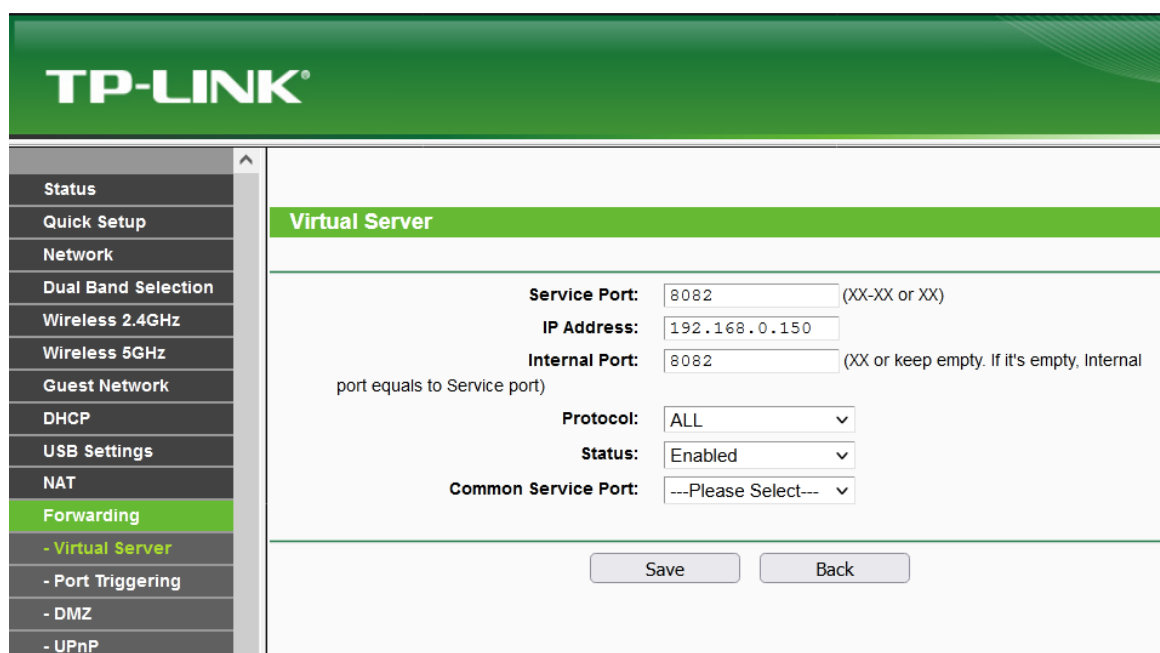
DHCP Address Reservation

This page displays the static IP address assigned by the DHCP Server and allows you to adjust these configurations by clicking the corresponding fields.

☐	MAC Address	IP Address	Status	Edit
☐	B8:27:EB:A5:A1:7B	192.168.0.150	Enabled	Edit
☐	90:A2:DA:00:51:06	192.168.0.160	Enabled	Edit

Slika 46. Rezervacija lokalnih IP adresa

Da bi se izvana komuniciralo s uređajima koji se nalaze unutar LAN mreže rutera, potrebno je uspostaviti *port forwarding*. Port koji će koristiti IP kamera je 8082, a za PLC glasi 502 što je uobičajeno za Modbus uređaje.



Slika 47. Postavljanje *Port-forwarding*-a za IP kameru

U sljedećem poglavlju je prikazano kako se može pristupiti postavkama IP kamere koristeći *port forwarding*.

6.2. Konfiguriranje video nadzora

Za realizaciju video nadzora korištena je web kamera Logitech C170 i računalo Raspberry Pi 4 model B prikazani na slici 48.

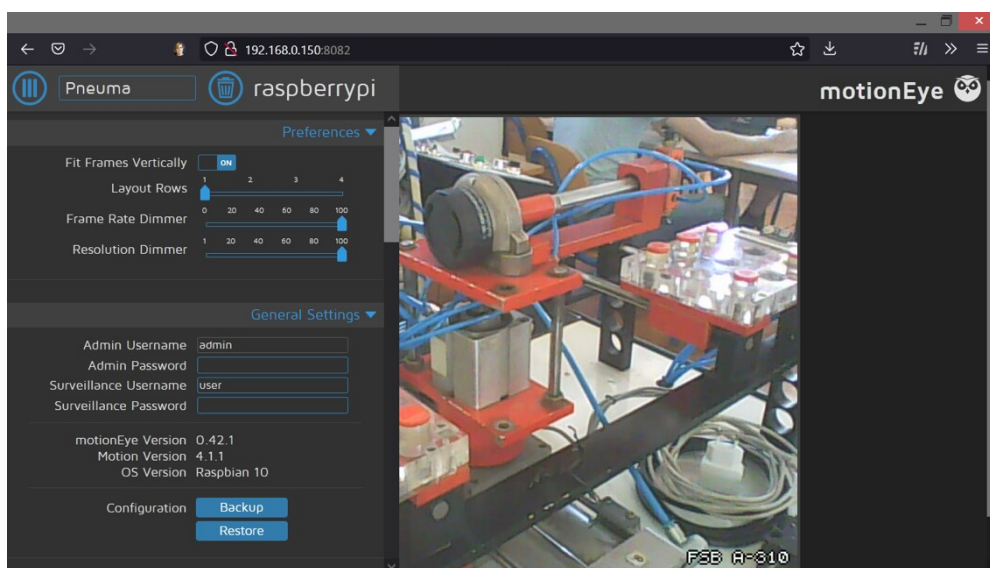


Slika 48. Raspberry Pi 4 model B (lijevo) [24]; Logitech C170 web kamera (desno) [25]

Raspberry Pi je moguće spojiti na internet preko RJ45 utora, a sama kamera spaja se na jedan od četiri USB utora. Prije nego se Raspberry Pi može koristiti kao računalo, potrebno je instalirati operativni sustav na njega. Korišten je operativni sustav *Rasbian* tj. *Raspberry Pi OS*. Operativni sustav se najprije instalira na MicroSD karticu koja se potom umetne u Raspberry Pi. Navedeni operativni sustav ne koristi GUI tj. grafičko sučelje, pa tako za instalaciju programa *MotionEye* potrebno je koristiti komandne linije.

Spajanjem Raspberry Pi-a na ruter on dobiva adresu koja je prethodno rezervirana za njega. Postoje dva načina za pristup postavkama. Ako smo spojeni na LAN mrežu rutera, postavkama pristupamo sa adresom: <http://192.168.0.150:8082>. Ako želimo pristupiti postavkama izvana onda koristimo statičku IP adresu rutera i broj *port*-a koji je konfiguriran u *port forwarding*-u. U našem slučaju to je *port* 8082.

Sučelje postavki za video nadzor prikazane su na slici 49. Unutar njih moguće je namještati postavke za osvjjetljenje, rezoluciju, *frame rate*, detekciju pokreta, pohranjivanje snimki itd.

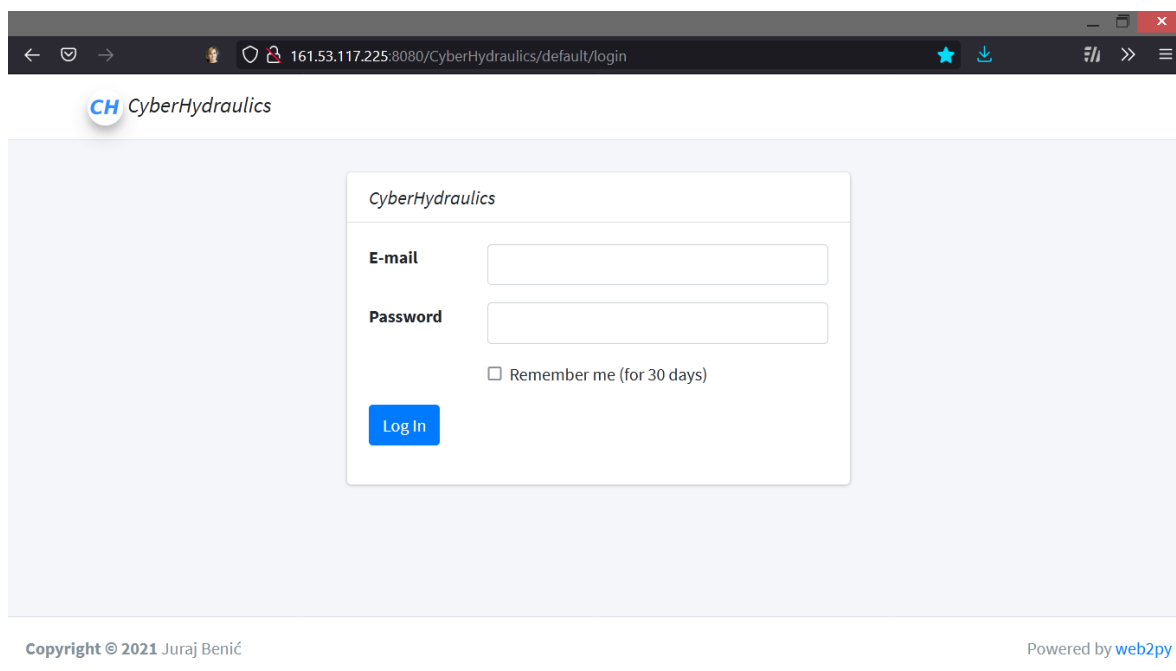


Slika 49. Postavke unutar programa MotionEye

U sljedećem poglavlju bit će opisano postavljanje i korištenje aplikacije *CyberHydraulic*.

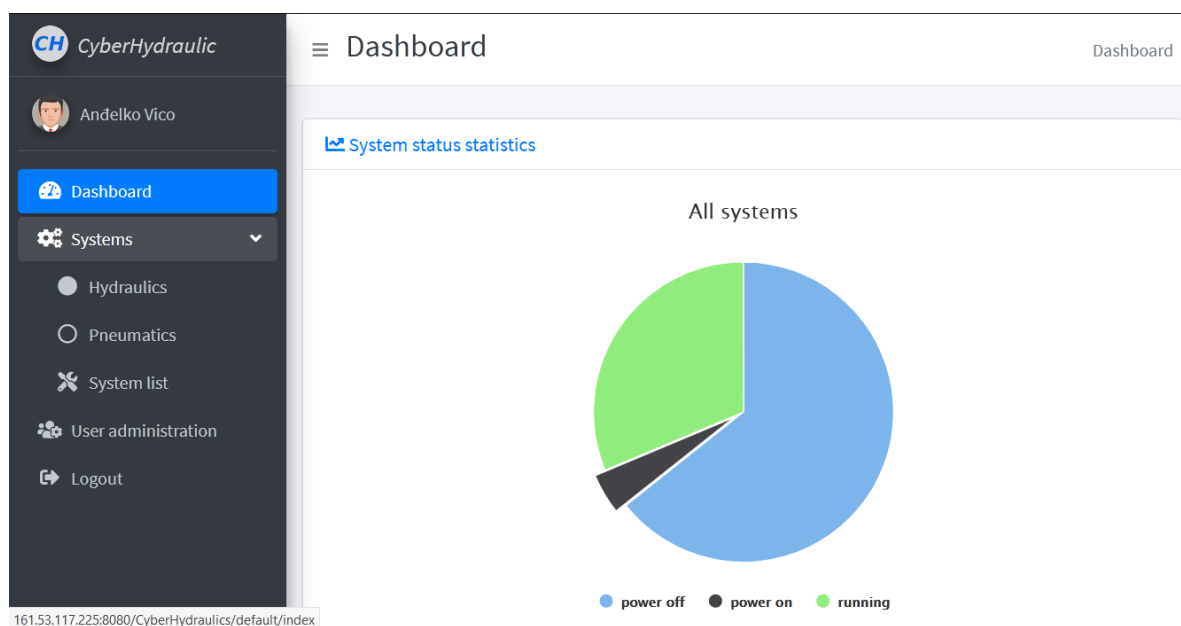
6.3. Korištenje sučelja aplikacije *CyberHydraulic*

Ove web aplikacije služe kao platforma za upravljanje i nadzor nad hidrauličkim i/ili pneumatskim sustavima. Također, putem iste moguće je uspostaviti i prikupljanje podataka. Prvi prvom pokretanju stranice dočekat će nas *login screen* kao na slici 50. Pristup je moguć tek nakon uspješne autorizacije s e-mail adresom i lozinkom.



Slika 50. Prikaz aplikacije *CyberHydraulic*

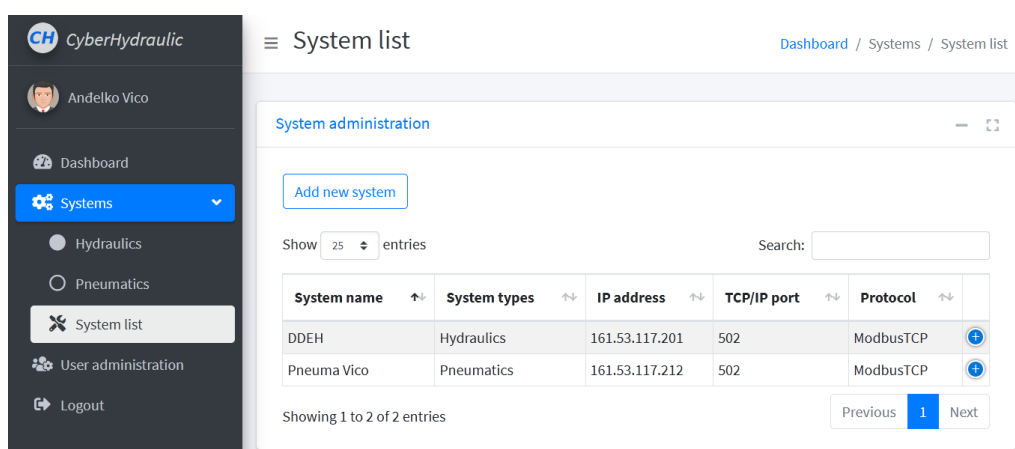
Nakon prijave dolazimo na *Dashboard*, odnosno početnu stranicu gdje se nalazi vizualan pregled stanja svih sustava.



Slika 51. *Dashboard* aplikacije *CyberHydraulic*

S lijeve strane nalazi se izbornik iz kojeg je moguće odabrati *Dashboard*, *Systems*, *User administration* i naposljetku *Logout*. Odabirom *Dashboard* vraćamo se na početnu stranicu. Pod *Systems* padajućim izbornikom nalaze se kategorija sustava *Hydraulics* i *Pneumatics*, a pod *Systems list* moguće je dodati nove ili izmijeniti postojeće sustave. Unutar *User administration* moguće je dodati nove korisnike aplikacije i mijenjati njihova ovlaštenja. Odabirom *Logout* prekida se trenutna korisnikova sesija.

Odabirom *Systems>System list* možemo vidjeti sve sustave koje pokriva aplikacija, kao što je to prikazano na slici 52.

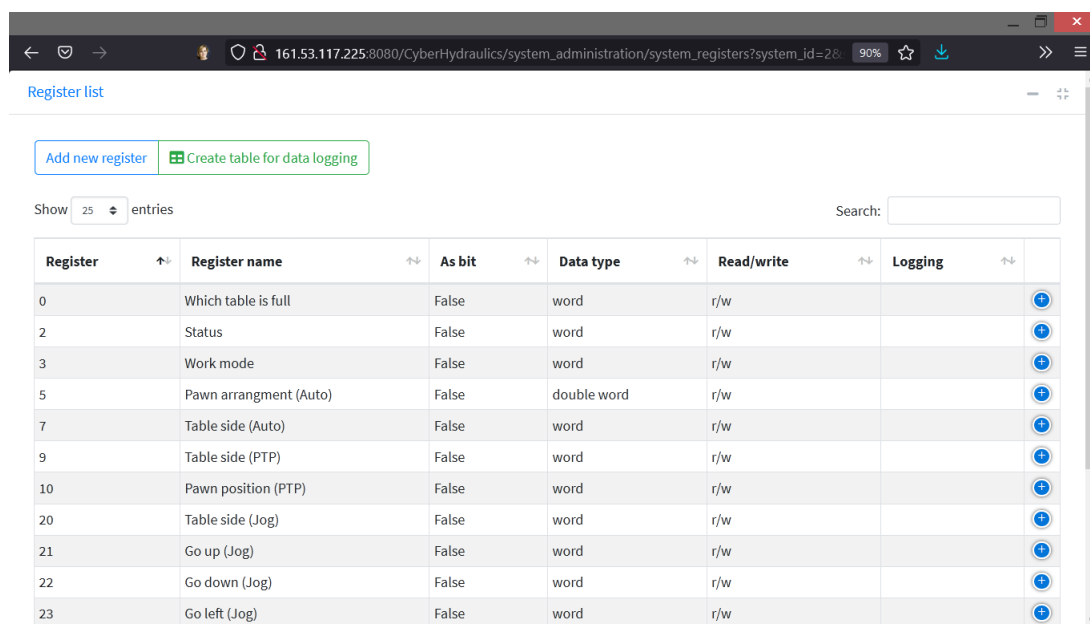


Slika 52. Izbornik *System list*

Nova pneumatska radna stanica dodaje se pod *Systems>System list* te klikom na *Add new system* nakon čega nas dočeka iskočni prozor kao na slici 53.

Slika 53. Prozor za dodavanje novog sustava

Za novo dodani sustav potrebno je još sastaviti tablicu registara s kojim će se upravljati putem Modbus TCP protokola. Tablica za pneumatski sustav prikazana je na slici 54. Za svaki registar nužno je unijeti njegov broj, proizvoljno ime, *True/False* za svojstvo registra ako je bit, tip podatka i pristup čitanja i/ili pisanja. Tipovi podataka koji se mogu unijeti su *word*, *double word*, *signed word* i *float*. Moguće je, ali ne i nužno, aktivirati zapisivanje podataka u bazu te dodavanje kratkog opisa tog registra.



Register list

[Add new register](#) [Create table for data logging](#)

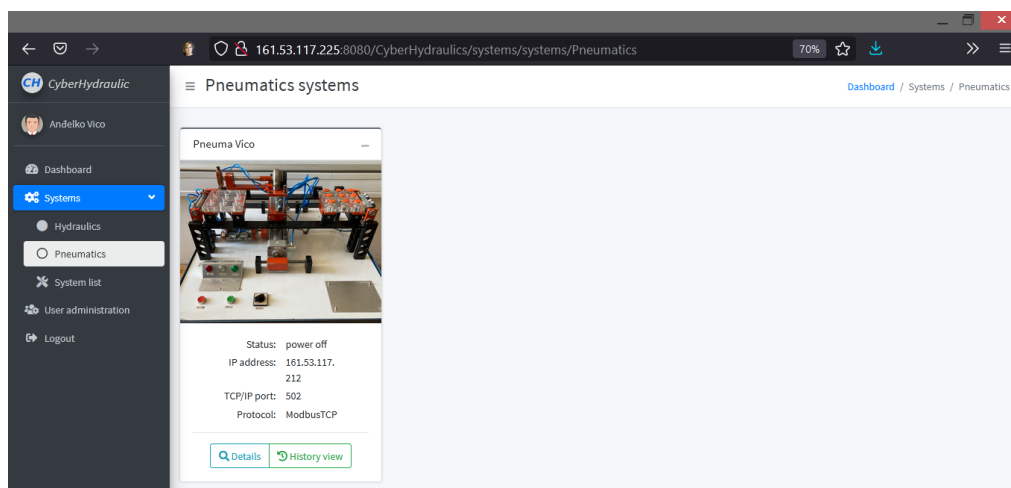
Show entries

Search:

Register	Register name	As bit	Data type	Read/write	Logging
0	Which table is full	False	word	r/w	
2	Status	False	word	r/w	
3	Work mode	False	word	r/w	
5	Pawn arrangment (Auto)	False	double word	r/w	
7	Table side (Auto)	False	word	r/w	
9	Table side (PTP)	False	word	r/w	
10	Pawn position (PTP)	False	word	r/w	
20	Table side (Jog)	False	word	r/w	
21	Go up (Jog)	False	word	r/w	
22	Go down (Jog)	False	word	r/w	
23	Go left (Jog)	False	word	r/w	

Slika 54. Tablica registara sustava

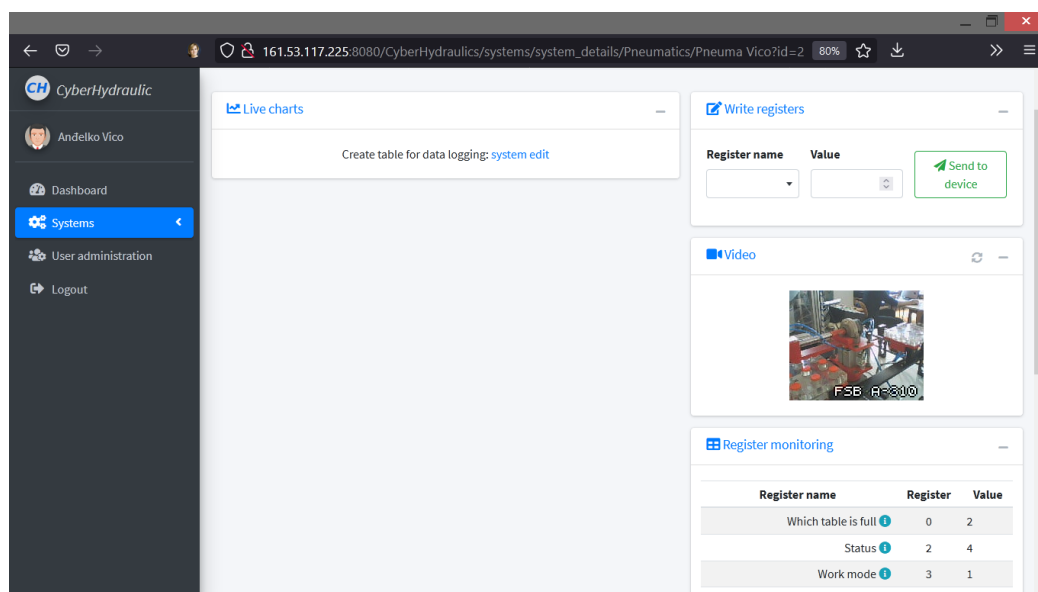
Nakon postavljanja svih parametara, sustav je sad moguće vidjeti pod *Systems>Pneumatics* kao na slici 55.



Slika 55. Pregled pneumatskih sustava

Klikom na *Details* aplikacija će nas odvesti na kontrolnu ploču kao na slici 56. S desne strane nalazi se padajući izbornik sa svim registrima i polje za unos vrijednosti. Ispod toga nalazi se

video prikaz stanice koju pruža Raspberry Pi zajedno sa programom *MotionEye*. Pri dnu desne strane nalazi se tablica vrijednosti svakog registra u stvarnom vremenu.



Slika 56. Kontrolna ploča sustava

Program se pokreće tako što se u padajućem izborniku odabere registar *Status* i postavi u vrijednost 4. Nakon unosa nužno je kliknuti na *Send to device*. Ako kod slanja bilo koje poruke dobijemo potvrđan odgovor od *Slave/Client* uređaja, na ekranu će pojaviti poruka *Sent successful*. Prije postavljanja statusnog registra moguće je definirati stol na kojem se nalaze valjčići. Ako korisnik ništa ne unese, podrazumijeva se da je unio vrijednost 1 odnosno da se valjčići nalaze na lijevom stolu. Potom se od korisnika očekuje da odredi modul rada unosom vrijednost 1, 2 ili 3 u registar *Work mode*.

Vrijednost „1“ predstavlja Automatski način rada. U tom modulu korisnik najprije treba s registrom *Table side (Auto)* odrediti s kojom stranom stola želi raditi. Potom s registrom *Pawn arrangement (Auto)* postavlja raspored tako što unosi 9-znamenasti binarni broj.

Vrijednost „2“ predstavlja modul *Point-to-point*. S registrom *Pawn position (PTP)* korisnik bira valjčić koji želi uzeti tako da unese neki broj od 1 do 9. Stol s kojeg želimo uzeti valjčić definiramo s registrom *Table side (PTP)*. Međutim, modul se može koristiti i bez unosa u to polje, jer program sam definira stranu ovisno gdje se nalazi ruka manipulatora. Nakon što se valjčić pokupi, program automatski postavlja stranu stola na suprotnu od one odakle je uzeo, a valjčić se ispušta tako što ponovno unesemo neku vrijednost u registar *Pawn position (PTP)*.

Vrijednost „3“ označava modul ručnog upravljanja. U ovom modulu korisnik može ručno upravljati gibanjem stola, micati manipulator, okretati ruku manipulatora i manipulirati valjčićima. Program automatski prepoznaje stranu stola, a može ju i korisnik ručno definirati.

7. ZAKLJUČAK

Zadatak ovog projekta bio je zastarjelu industrijsku opremu zamijeniti suvremenim rješenjima. Pojedini dijelovi edukacijske radne stanice zahtijevali su zamjenu oštećenih elemenata, obnovu ožičenja i druge oblike popravka.

Nakon montaže novih dijelova osmišljena je nova funkcija stanice. Funkcija tj. program stanice sastoji se od manipulacije valjčićima koji se nalaze na dvama razdvojenim stolovima. Program se nalazi na CONTROLLINO PLC-u, ima tri modula rada, a isprogramiran je u cijelosti u Arduino jeziku. Unutar programa implementiran je Modbus TCP/IP, a PLC je preko rutera spojen na Internet. Ruteru je dodijeljena statička IP adresa. Za upravljanje i nadzor korištena je web platforma *CyberHydraulic* koju je razvio asistent Juraj Benić. Platforma omogućuje, na vizualno pregledan način, upravljanje Modbus tablicom te video nadzor nad pneumatskom stanicom.

Prijedlozi za unaprjeđenje uključuju zamjenu drugih dotrajalih dijelova, proširenje funkcionalnosti kôda, poboljšanje stabilnosti kôda te dodavanje novih mogućnosti na platformi *CyberHydraulic*. Na ovom radu pokazano je kako stara oprema može dobiti novi dah života ako se u nju ugradi jedan dio silnih mogućnosti Interneta stvari koji, u svakom slučaju, nam donosi zanimljivu budućnost.

LITERATURA

- [1] Definicija Interneta stvari: Jednostavno objašnjenje, <https://leastwastefulcities.com/sigurnost-na-internetu/definicija-interneta-stvari-jednostavno-objanjenje>, pristupljeno 23.6.2021.
- [2] F. Lambert, Tesla vehicles can now diagnose themselves and even pre-order parts for service, <https://electrek.co/2019/05/06/tesla-diagnose-pre-order-parts-service>, pristupljeno 23.6.2021.
- [3] A. Lücke, Festo FPC 202C Program Logic Controller: Operating manual, Esslingen: FESTO Electronic, 1988.
- [4] Pneumatski ventil, <https://www.festo.com/us/en/a/download-document/datasheet/9982>, pristupljeno 24.6.2021.
- [5] Pneumatski ventil, <https://www.festo.com/us/en/a/download-document/datasheet/4450>, pristupljeno 24.6.2021.
- [6] Elektromag. zavojnica, <https://www.festo.com/us/en/a/download-document/datasheet/34411>, pristupljeno 24.6.2021.
- [7] Pneumatski ventil, <https://www.festo.com/media/pim/275/D15000100123275.PDF>, pristupljeno 24.6.2021.
- [8] Vakuumska sklopka, https://www.festo.com/net/SupportPortal/Files/339903/VUV_1982-06_219816F3.pdf, pristupljeno 24.6.2021.
- [9] Element LC-3-1/8, <https://www.festo.com/net/SupportPortal/Files/44034/LC-3-18.pdf>, pristupljeno 24.6.2021.
- [10] <https://www.facebook.com/Controllino>, pristupljeno 24.6.2021.
- [11] CONTROLLINO product comparison, <https://www.controllino.biz/controllino-product-comparison>, pristupljeno 24.6.2021.
- [12] CONTROLLINO Maxi Automation, <https://www.controllino.com/product/controllino-maxi-automation>, pristupljeno 25.6.2021.
- [13] Ventilski razvodnik, https://www.festo.com/cat/hr_hr/data/doc_engb/PDF/EN/VTUG-G_EN.PDF, pristupljeno 25.6.2021.
- [14] Novi generator vakuuma, <https://www.festo.com/us/en/a/download-document/datasheet/193488>, pristupljeno 25.6.2021.
- [15] Senzor vakuuma, <https://www.festo.com/media/pim/034/D15000100123034.PDF>, pristupljeno 25.6.2021.

- [16] Hvataljka, <https://www.festo.com/media/pim/034/D15000100123034.PDF>, pristupljeno 25.6.2021.
- [17] Induktivni senzor, <https://www.festo.com/us/en/a/download-document/datasheet/543862>, pristupljeno 25.6.2021.
- [18] Priključak, <https://www.festo.com/us/en/a/download-document/datasheet/193144>, pristupljeno 25.6.2021.
- [19] Pripremna grupa, <https://www.festo.com/us/en/a/download-document/datasheet/8002799>, pristupljeno 25.6.2021.
- [20] Modbus messaging implementation guide v1.0b, https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf, pristupljeno 27.6.2021.
- [21] Modbus TCP, <https://www.throughput.co.za/protocols/modbus-tcp-protocols.html>, pristupljeno 27.6.2021.
- [22] Arduino board library setup, <https://www.controllino.com/knowledge-base/board-library-setup-in-arduino-ide/>, pristupljeno 27.6.2021.
- [23] Ruter TP-LINK, <https://www.tp-link.com/us/home-networking/wifi-router/archer-c20/>, pristupljeno 28.6.2021.
- [24] Ruter TP-LINK, <https://www.tp-link.com/us/home-networking/wifi-router/archer-c20/>, pristupljeno 28.6.2021.
- [25] Raspberry Pi, https://bs.wikipedia.org/wiki/Raspberry_Pi, pristupljeno 28.6.2021.
- [26] Arduino in IoT, <https://www.youtube.com/watch?v=dQw4w9WgXcQ>, pristupljeno 29.6.2021.

PRILOZI

I. Programski kôd PLC-a

```

/*
 * Latest version of this code can viewed on GitHub:
 * https://github.com/andelkov/MasterThesisProject/
 * , as filename: PLC_program.ino
 */

#include <Controllino.h>
#include <SPI.h>
#include <Ethernet.h>
#include "Mudbus.h"

#define NULL 0

#define highWord(w) ((w) >> 16)
#define lowWord(w) ((w) & 0xffff)
#define makeLong(hi, low) (((long) hi) << 16 | (low)) //convert two 16bit variables
back to a 32bit variable

Mudbus Mb;

int cooldown = 1000;
unsigned long message;
uint32_t lowWord, hiWord;
int messageArray[10];
String messageString;
unsigned int temp1 = 0;
unsigned int temp2 = 0;

byte pawnTemp;
byte tableSide;
String arrayPart;
String debugMessage = " ";
String modeMessage = "";

bool isUp = false;
bool isMoving = false;
bool isHandFull = false;

byte tableLeft[10] = { NULL, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
byte tableRight[10] = { NULL, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

//Pin configuration:
const byte C1_cilindar = CONTROLLINO_R1; // R1      Cylinder 1
const byte C2_cilindar = CONTROLLINO_R2; // R2      Cylinder 2

const byte C3_cilindar = CONTROLLINO_R3; // R3      Cylinder 3
const byte C4_cilindar = CONTROLLINO_R4; // R4      Cylinder 4

const byte C5_cilindar = CONTROLLINO_R5; // R5      Cylinder 5
const byte C6_cilindar = CONTROLLINO_R6; // R6      Cylinder 6

const byte C7_cilindar = CONTROLLINO_R7; // R7      Cylinder 7 (180 degree body rotate)
const byte C8_cilindar = CONTROLLINO_R8; // R8      Cylinder 8 (up-down of body)
const byte C9_cilindar = CONTROLLINO_R9; // R9      Cylinder 9 (180 degree hand rotate)

const byte C1_izvucen = CONTROLLINO_A0; // AI0     sensor C1.0
const byte C1_uvucen = CONTROLLINO_A1; // AI1     sensor C1.1
const byte C2_uvucen = CONTROLLINO_A2; // AI2     sensor C2.0
const byte C2_izvucen = CONTROLLINO_A3; // AI3     sensor C2.1

```

```

60
61 const byte C3_izvucen = CONTROLLINO_A4; // AI4 sensor C3.0
62 const byte C3_uvucen = CONTROLLINO_A5; // AI5 sensor C3.1
63 const byte C4_uvucen = CONTROLLINO_A6; // AI6 sensor C4.0
64 const byte C4_izvucen = CONTROLLINO_A7; // AI7 sensor C4.1
65
66 const byte C5_izvucen = CONTROLLINO_A8; // AI8 sensor C5.0
67 const byte C5_uvucen = CONTROLLINO_A9; // AI9 sensor C5.1
68 const byte C6_uvucen = CONTROLLINO_A10; // AI10 sensor C6.0
69 const byte C6_izvucen = CONTROLLINO_A11; // AI11 sensor C6.1
70
71 const byte Vacuum_1 = CONTROLLINO_D0; // D00 turn on vacuum 1
72 const byte Vacuum_2 = CONTROLLINO_D1; // D01 turn on vacuum 2
73
74 const byte LED_Start = CONTROLLINO_D5; // D05 light Start/ON
75 const byte LED_Error = CONTROLLINO_D6; // D06 light Error
76 const byte LED_Stop = CONTROLLINO_D7; // D07 light Stop
77
78 const byte handIsRight = 66; // DI0 sensor C7.0, hand is now at right
  side
79 const byte handIsLeft = 67; // DI1 sensor C7.1, hand is now at left
  side
80
81 const byte pawnGrabbed_V1 = 10; // DI2 sensor vacuum 1
82 const byte pawnGrabbed_V2 = 11; // DI3 sensor vacuum 2
83
84 const byte interruptStartPin = 18; // IN0 interrupt input, Start button
85 volatile byte startPressed = LOW;
86
87 const byte interruptStopPin = 19; // IN1 interrupt input, Stop button
88 volatile byte stopPressed = LOW;
89
90 void setup() {
91   uint8_t mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
92   uint8_t ip[] = { 192, 168, 0, 160 };
93   uint8_t gateway[] = { 161, 53, 116, 1 };
94   uint8_t subnet[] = { 255, 255, 252, 0 };
95   Ethernet.begin(mac, ip, subnet);
96
97   delay(2000);
98
99   Serial.begin(9600);
100  Serial.print("Started. ");
101  Serial.print("My IP address: ");
102  Serial.println(Ethernet.localIP());
103
104  pinMode(CONTROLLINO_IN0, INPUT); // IN0 button Start
105  pinMode(CONTROLLINO_IN1, INPUT); // IN1 button Stop
106
107  pinMode(CONTROLLINO_A0, INPUT); // AI0 sensor C1.0
108  pinMode(CONTROLLINO_A1, INPUT); // AI1 sensor C1.1
109  pinMode(CONTROLLINO_A2, INPUT); // AI2 sensor C2.0
110  pinMode(CONTROLLINO_A3, INPUT); // AI3 sensor C2.1
111  pinMode(CONTROLLINO_A4, INPUT); // AI4 sensor C3.0
112  pinMode(CONTROLLINO_A5, INPUT); // AI5 sensor C3.1
113
114  pinMode(CONTROLLINO_A6, INPUT); // AI6 sensor C4.0
115  pinMode(CONTROLLINO_A7, INPUT); // AI7 sensor C4.1
116  pinMode(CONTROLLINO_A8, INPUT); // AI8 sensor C5.0
117  pinMode(CONTROLLINO_A9, INPUT); // AI9 sensor C5.1

```

```

118 pinMode(CONTROLLINO_A10, INPUT); // AI10 sensor C6.0
119 pinMode(CONTROLLINO_A11, INPUT); // AI11 sensor C6.1
120
121 pinMode(66, INPUT); // DI0 sensor C7.0,
122 pinMode(67, INPUT); // DI1 sensor C7.1
123 pinMode(10, INPUT); // DI2 sensor vacuum 1
124 pinMode(11, INPUT); // DI3 sensor vacuum 2
125
126 pinMode(CONTROLLINO_D0, OUTPUT); // D00 vacuum 1 ON (upper)
127 pinMode(CONTROLLINO_D1, OUTPUT); // D01 vacuum 2 ON (lower)
128 pinMode(CONTROLLINO_D5, OUTPUT); // D05 LED Ready
129 pinMode(CONTROLLINO_D6, OUTPUT); // D06 LED Error
130 pinMode(CONTROLLINO_D7, OUTPUT); // D07 LED Stop
131
132 pinMode(CONTROLLINO_R1, OUTPUT); // R1 Cylinder 1
133 pinMode(CONTROLLINO_R2, OUTPUT); // R2 Cylinder 2
134 pinMode(CONTROLLINO_R3, OUTPUT); // R3 Cylinder 3
135 pinMode(CONTROLLINO_R4, OUTPUT); // R4 Cylinder 4
136 pinMode(CONTROLLINO_R5, OUTPUT); // R5 Cylinder 5
137 pinMode(CONTROLLINO_R6, OUTPUT); // R6 Actuator 6
138 pinMode(CONTROLLINO_R7, OUTPUT); // R7 Actuator 7 (180 degree body rotate)
139 pinMode(CONTROLLINO_R8, OUTPUT); // R8 Actuator 8 (up-down of body)
140 pinMode(CONTROLLINO_R9, OUTPUT); // R9 Actuator 9 (180 degree hand rotate)
141
142 pinMode(interruptStartPin, INPUT);
143 pinMode(interruptStopPin, INPUT);
144
145 attachInterrupt(digitalPinToInterrupt(interruptStartPin), StartPressed, FALLING);
146 attachInterrupt(digitalPinToInterrupt(interruptStopPin), StopPressed, RISING);
147
148 for (byte i = 0; i < 125; i++) { // Set every Modbus register value to 0
149     Mb.R[i] = 0;
150 }
151 Mb.R[5] = -1; // For Auto mode
152 Mb.R[6] = -1; // For Auto mode
153 Mb.R[2] = 2; // Set status to "Power on"
154 Serial.println("Modbus registers set to 0.");
155 }
156
157 ////////////////////////////////////////////////// MAIN //////////////////////////////////////
158 void loop() {
159     while (Mb.R[2] != 4) {
160         if (debugMessage != "--> Ready to start. ") {
161             debugMessage = "--> Ready to start. ";
162             Serial.println(debugMessage);
163         }
164         Mb.Run();
165         if (Mb.R[0] == 2) {
166             Mb.Run();
167             for (byte i = 1; i < 10; i++) {
168                 tableLeft[i] = 0;
169                 tableRight[i] = 1;
170             }
171         }
172     }
173
174     digitalWrite(LED_Start, 0);
175     delay(500);
176
177     digitalWrite(LED_Start, 1);

```

```

178     delay(500);
179     Mb.Run();
180 }
181
182 Mb.Run();
183
184 switch (Mb.R[3]) {
185 case 1:
186     if (modeMessage != "--> Auto-mode selected.") {
187         modeMessage = "--> Auto-mode selected.";
188         Serial.println(modeMessage);
189     }
190
191     temp1 = Mb.R[6];
192     temp2 = Mb.R[5];
193
194     if ((Mb.R[6] >= 0) && ((Mb.R[7] == 1) || (Mb.R[7] == 2))) {
195
196         message = makeLong(temp1, temp2);
197         Serial.print("Received Mb.R[5] and Mb.R[6] messages: ");
198         Serial.print(Mb.R[5]);
199         Serial.print(", ");
200         Serial.println(Mb.R[6]);
201         Serial.print("Coververted Mb.R[5] and Mb.R[6] messages: ");
202         Serial.print(temp1);
203         Serial.print(", ");
204         Serial.println(temp2);
205
206         messageString = String(message);
207
208         Serial.print(" The re-converted received message is: ");
209         Serial.println(messageString);
210         Serial.println(onlyBinaryNumbers(messageString));
211
212         if (messageString.length() == 9) {
213             for (char i = 0; i < 9; i++) {
214                 arrayPart = messageString.charAt(i);
215                 messageArray[i + 1] = arrayPart.toInt();
216             }
217             messageArray[0] = 0;
218         }
219         else if (messageString.length() < 9) {
220             for (char i = 0; i < messageString.length(); i++) {
221                 arrayPart = messageString.charAt(i);
222                 messageArray[i + (10 - messageString.length())] = arrayPart.toInt();
223             }
224             for (char i = 1; i < (10 - messageString.length()); i++) {
225                 messageArray[i] = 0;
226             }
227             messageArray[0] = NULL;
228         }
229
230         Serial.print("Our message Int array is: ");
231         for (int i = 1; i < 10; i++)
232         {
233             Serial.print(messageArray[i]);
234             Serial.println(" ");
235
236             if ((messageString.length() < 10) && (onlyBinaryNumbers(messageString) == true)
{

```

```

237     if (Mb.R[7] == 1) {
238         Serial.println("--> Filling table 1. ");
239         for (byte j = 1; j < 10; j++) {
240             Mb.Run();
241             Serial.print("--> Doing j number: ");
242             Serial.println(j);
243             if (messageArray[j] == 1) {
244
245                 if (tableLeft[j] != 1) {
246                     pawnTemp = findAvailablePawn(2);
247                     Serial.print("--> Found pawn number at right table: ");
248                     Serial.println(pawnTemp);
249                     if (pawnTemp == 0) {
250                         Serial.println("No pawns available at right table");
251                     }
252                     else {
253                         rotateRight();
254                         goTo(2, pawnTemp);
255                         pawnPickUp();
256                         tableRight[pawnTemp] = 0;
257
258                         rotateLeft();
259                         goTo(1, j);
260
261                         pawnDrop();
262                         tableLeft[j] = 1;
263                     }
264                 }
265             }
266             else if (messageArray[j] == 0) {
267                 if (tableLeft[j] == 1) {
268                     pawnTemp = findFreeSpot(2);
269                     Serial.print("Found free spot at right table: ");
270                     Serial.println(pawnTemp);
271                     if (pawnTemp == 0) {
272                         Serial.println("No free spot available at right table. ");
273                     }
274                     else {
275                         rotateLeft();
276                         goTo(1, j);
277                         pawnPickUp();
278                         tableLeft[j] = 0;
279
280                         rotateRight();
281                         goTo(2, pawnTemp);
282                         pawnDrop();
283                         tableRight[pawnTemp] = 1;
284                     }
285                 }
286             }
287         }
288         if (debugMessage != "Auto-mode transfer completed. ") {
289             debugMessage = "Auto-mode transfer completed. ";
290             Serial.println(" ");
291             Serial.println(debugMessage);
292         }
293     }
294     else if (Mb.R[7] == 2) {
295         Serial.println("--> Filling table 2. ");
296         for (byte j = 1; j < 10; j++) {

```

```

297 Mb.Run();
298 Serial.print("--> Doing j number: ");
299 Serial.println(j);
300 if (messageArray[j] == 1) {
301     if (tableRight[j] != 1) {
302
303         pawnTemp = findAvailablePawn(1);
304         Serial.print("--> Found pawn at left table: ");
305         Serial.println(pawnTemp);
306
307         if (pawnTemp == 0) {
308             Serial.println("No pawns available at left table");
309
310         }
311         else {
312             rotateLeft();
313             goTo(1, pawnTemp);
314             pawnPickUp();
315             tableLeft[pawnTemp] = 0;
316
317             rotateRight();
318             goTo(2, j);
319             pawnDrop();
320             tableRight[j] = 1;
321         }
322     }
323 }
324 else if (messageArray[j] == 0) {
325     if (tableRight[j] == 1) {
326         pawnTemp = findFreeSpot(1);
327         Serial.print("Found free spot at left table: ");
328         Serial.println(pawnTemp);
329         if (pawnTemp == 0) {
330             Serial.println("No free spot available at left table. ");
331         }
332         else {
333             rotateRight();
334             goTo(2, j);
335             pawnPickUp();
336             tableRight[j] = 0;
337
338             rotateLeft();
339             goTo(1, pawnTemp);
340             pawnDrop();
341             tableLeft[pawnTemp] = 1;
342         }
343     }
344 }
345 }
346 if (debugMessage != "Auto-mode transfer completed.") {
347     debugMessage = "Auto-mode transfer completed.";
348     Serial.println(" ");Serial.println(debugMessage);Serial.println(" ");
349 }
350 }
351 else {
352     Serial.println("Wrong table side choosen. Check input.");
353 }
354 }
355 else {
356     Mb.R[5] = -1;

```

```

357     Mb.R[6] = -1;
358 }
359
360 Serial.print("Our table positions are on LEFT table: ");
361 for (int i = 1; i < 10; i++)
362 {
363     Serial.print(tableLeft[i]);
364 } Serial.println(" ");
365
366 Serial.print("Our table positions are on RIGHT table: ");
367 for (int i = 1; i < 10; i++)
368 {
369     Serial.print(tableRight[i]);
370 } Serial.println(" ");
371
372 Mb.R[5] = -1;
373 Mb.R[6] = -1;
374 }
375
376 break;
377 case 2:
378     if (modeMessage != "--> Point-to-point mode.") {
379         modeMessage = "--> Point-to-point mode.";
380         Serial.println(modeMessage);
381     }
382
383     if (Mb.R[9] == 0) {
384         Mb.R[9] = currentTableSide();
385     }
386
387     do {
388         Mb.Run();
389
390         if ((Mb.R[9] == 1) || (Mb.R[9] == 2)) {
391             if ((Mb.R[10] >= 1) && (Mb.R[10] <= 9)) {
392                 if (Mb.R[9] == 1) {
393
394                     if ((isHandFull == true) && (isSpotEmpty(1, Mb.R[10]) == true)) {
395                         rotateLeft();
396                         goTo(1, Mb.R[10]);
397                         pawnDrop();
398                         tableLeft[Mb.R[10]] = 1;
399
400                         Mb.R[9] = 2;
401                         isHandFull == false;
402                     }
403                     else if ((isSpotEmpty(1, Mb.R[10]) == false) && (isHandFull == false)) {
404                         rotateLeft();
405                         goTo(1, Mb.R[10]);
406                         pawnPickUp();
407                         tableLeft[Mb.R[10]] = 0;
408
409                         Mb.R[9] = 2;
410                         isHandFull = true;
411                     }
412                 else {
413                     if (debugMessage != "--> Can't do that move. ") {
414                         debugMessage = "--> Can't do that move. ";
415                         Serial.println(debugMessage);
416                         flashError();

```

```

417     }
418
419     }
420 }
421 else if (Mb.R[9] == 2) {
422     if ((isHandFull == true) && (isSpotEmpty(2, Mb.R[10]) == true)) {
423         rotateRight();
424         goTo(2, Mb.R[10]);
425         pawnDrop();
426         tableRight[Mb.R[10]] = 1;
427
428         Mb.R[9] = 1;
429         isHandFull == false;
430     }
431     else if ((isSpotEmpty(2, Mb.R[10]) == false) && (isHandFull == false)) {
432         rotateRight();
433         goTo(2, Mb.R[10]);
434         pawnPickUp();
435         tableRight[Mb.R[10]] = 0;
436
437         Mb.R[9] = 1;
438         isHandFull = true;
439     }
440     else {
441         if (debugMessage != "--> Can't do that move. ") {
442             debugMessage = "--> Can't do that move. ";
443             Serial.println(debugMessage);
444             flashError();
445         }
446     }
447 }
448
449 debugMessage = "Auto-movement fullfiled. ";
450 Serial.println(debugMessage);
451
452 Serial.print("Our table positions are on LEFT table: ");    //printanje
453 arraya
454 for (int i = 1; i < 10; i++)
455 {
456     Serial.print(tableLeft[i]);
457 } Serial.println(" ");
458
459 Serial.print("Our table positions are on RIGHT table: ");    //printanje
460 arraya
461 for (int i = 1; i < 10; i++)
462 {
463     Serial.print(tableRight[i]);
464 } Serial.println(" ");
465
466 }
467 Mb.R[10] = 0;
468 } while (isHandFull == true);
469 break;
470 case 3:
471     if (modeMessage != "--> Jog mode.") {
472         modeMessage = "--> Jog mode.";
473         Serial.println(modeMessage);
474

```

```

475     if (digitalRead(handIsLeft) == 1) {
476         Mb.R[20] = 1;
477     }
478     else if (digitalRead(handIsRight) == 1) {
479         Mb.R[20] = 2;
480     }
481 }
482
483 if (Mb.R[20] == 0) {
484     Mb.R[20] = currentTableSide();
485 }
486 do {
487     Mb.Run();
488     //go up
489     if (Mb.R[22] == 1) {
490         if (Mb.R[20] == 1) {
491             if ((digitalRead(C1_uvucen) == 1) && (digitalRead(C2_uvucen) == 1)) {
492                 tableGoCenterY(1);
493             }
494             else {
495                 tableGoUp(1); //
496             }
497
498             Mb.R[22] = 0;
499         }
500         else if (Mb.R[20] == 2) {
501             if ((digitalRead(C3_uvucen) == 1) && (digitalRead(C4_uvucen) == 1)) {
502                 tableGoCenterY(2);
503             }
504             else {
505                 tableGoUp(2); //
506             }
507
508             Mb.R[22] = 0;
509         }
510         else {
511             if ((debugMessage != "Please select a proper table number. Number received:
512 ") && (Mb.R[20] != 0)) {
513                 debugMessage = "Please select a proper table number. Number received: ";
514                 Serial.print(debugMessage);
515                 Serial.println(Mb.R[22]);
516             }
517         }
518         //go down
519         if (Mb.R[21] == 1) {
520             if (Mb.R[20] == 1) {
521                 if ((digitalRead(C1_izvucen) == 1) && (digitalRead(C2_izvucen) == 1)) {
522                     tableGoCenterY(1);
523                 }
524                 else {
525                     tableGoDown(1); //
526                 }
527                 Mb.R[21] = 0;
528             }
529             else if (Mb.R[20] == 2) {
530                 if ((digitalRead(C3_izvucen) == 1) && (digitalRead(C4_izvucen) == 1)) {
531                     tableGoCenterY(2);
532                 }
533                 else {

```

```

534         tableGoDown(2); //
535     }
536
537     Mb.R[21] = 0;
538 }
539 else {
540     if ((debugMessage != "Please select a proper table number. Number received:
541 ") && (Mb.R[20] != 0)) {
542         debugMessage = "Please select a proper table number. Number received: ";
543         Serial.print(debugMessage);
544         Serial.println(Mb.R[21]);
545         flashError();
546     }
547 }
548 //go left
549 if (Mb.R[23] == 1) {
550     if ((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_izvucen) == 1)) {
551         tableGoCenterX();
552     }
553     else {
554         tableGoLeft();
555     }
556     Mb.R[23] = 0;
557 }
558 //go right
559 if (Mb.R[24] == 1) {
560     if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
561         tableGoCenterX();
562     }
563     else {
564         tableGoRight();
565     }
566     Mb.R[24] = 0;
567 }
568 //rotate left or right
569 if ((Mb.R[25] == 1) || (Mb.R[25] == 2)) {
570
571     if (Mb.R[25] == 1) {
572         if (digitalRead(handIsLeft) == 0) {
573             rotateLeft();
574             Mb.R[20] = 1;
575         }
576         else {
577             Serial.println("--> Hand is already left. ");
578             flashError();
579         }
580     }
581
582     else if (Mb.R[25] == 2) {
583         if (digitalRead(handIsRight) == 0) {
584             rotateRight();
585             Mb.R[20] = 2;
586         }
587         else {
588             Serial.println("--> Hand is already right. ");
589             flashError();
590         }
591     }
592 }

```

```

593
594     Mb.R[25] = 0;
595 }
596 // pick up the pawn
597 if (Mb.R[26] == 1) {
598
599     if (isSpotEmpty(currentTableSide(), currentPawnPosition()) == false) {
600         pawnPickUp();
601
602         if (currentTableSide() == 1) {
603             tableLeft[currentPawnPosition()] = 0;
604         }
605         else if (currentTableSide() == 2) {
606             tableRight[currentPawnPosition()] = 0;
607         }
608
609         Serial.print("--> Picked pawn from position: ");
610         Serial.println(currentPawnPosition());
611     }
612     else {
613         Serial.println("--> There is no pawn under the hand to pick up. ");
614         flashError();
615     }
616
617     Mb.R[26] = 0;
618 }
619 // drop the pawn
620 if (Mb.R[27] == 1) {
621     if (isSpotEmpty(currentTableSide(), currentPawnPosition()) == true) {
622         pawnDrop();
623
624         if (currentTableSide() == 1) {
625             tableLeft[currentPawnPosition()] = 1;
626         }
627         else if (currentTableSide() == 2) {
628             tableRight[currentPawnPosition()] = 1;
629         }
630
631         Serial.print("--> Picked pawn from position: ");
632         Serial.println(currentPawnPosition());
633     }
634     else {
635         Serial.println("--> There is a pawn under the hand. Can't drop the pawn in
636 this position. ");
637         flashError();
638     }
639
640     Mb.R[27] = 0;
641 }
642 } while (isHandFull == true);
643
644 break;
645 case 4:
646     //For testing:
647
648     Serial.print(Mb.R[5]);
649     Serial.print(Mb.R[6]);
650
651     message = makeLong(Mb.R[6], Mb.R[5]);
652     Serial.print("Message string is: ");

```

```

652     messageString = String(message);
653     Serial.print(messageString);
654
655
656
657     Mb.R[3] = 5;
658     break;
659 default:
660     if (modeMessage != "--> Please select a mode.") {
661         modeMessage = "--> Please select a mode.";
662         Serial.println(modeMessage);
663     }
664     break;
665 }
666 }
667 ////////////////////////////////////////////////// FUNCTIONS
668 //////////////////////////////////////////////////
669 void tableGoRight() {
670
671     if ((digitalRead(C5_izvucen) == 0) || (digitalRead(C6_izvucen) == 0)) {
672         isMoving = true;
673         Serial.print("--> Going right. ");
674
675         digitalWrite(CONTROLLINO_R6, HIGH);
676         digitalWrite(CONTROLLINO_R5, HIGH);
677         delay(cooldown);
678
679         while (isMoving == true) {
680             Serial.print("Waiting for input from sensors C5 and C6... ");
681             if ((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_izvucen) == 1)) {
682                 isMoving = false;
683                 Serial.println("Move completed. ");
684             }
685         }
686     }
687     else {
688         Serial.println("--> Can't go anymore right.");
689         digitalWrite(LED_Error, 1);
690         delay(500);
691         digitalWrite(LED_Error, 0);
692     }
693 }
694
695 void tableGoLeft() {
696     if ((digitalRead(C5_uvucen) == 0) || (digitalRead(C6_uvucen) == 0)) {
697
698         isMoving = true;
699         Serial.print("--> Going left. ");
700
701         digitalWrite(CONTROLLINO_R6, LOW);
702         digitalWrite(CONTROLLINO_R5, LOW);
703         delay(cooldown);
704
705         Serial.print("Waiting for input from sensors C5 and C6. ");
706         while (isMoving == true) {
707
708             if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
709                 isMoving = false;
710                 Serial.println("Move completed.");

```

```

711     }
712 }
713 }
714 else {
715     Serial.println("--> Can't go anymore left.");
716     digitalWrite(LED_Error, 1);
717     delay(500);
718     digitalWrite(LED_Error, 0);
719 }
720 }
721
722 void tableGoCenterX() {
723
724     Serial.print("---> Going to the center in the X direction. ");
725     isMoving == true;
726     digitalWrite(CONTROLLINO_R6, LOW);
727     digitalWrite(CONTROLLINO_R5, HIGH);
728     delay(cooldown);
729
730     while (isMoving == true) {
731         Serial.print("Waiting for input from sensors C5 and C6. ");
732         if ((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
733             isMoving = false;
734             Serial.println("Move completed. Table in X centre. ");
735         }
736     }
737 }
738
739 void tableGoCenterY(char tableSide) {
740
741     if (tableSide == 1) {
742
743         isMoving = true;
744         Serial.print("---> Moving left table to the Y centre. ");
745
746         digitalWrite(C1_cilindar, HIGH);
747         digitalWrite(C2_cilindar, LOW);
748         delay(cooldown);
749
750         Serial.print("Waiting for input from sensors on C1 and C2. ");
751         while (isMoving == true) {
752
753             if ((digitalRead(C1_izvucen) == 1) && (digitalRead(C2_uvucen) == 1)) {
754                 isMoving = false;
755                 Serial.println("Move completed. ");
756             }
757         }
758     }
759     else if (tableSide == 2) {
760
761         isMoving = true;
762         Serial.print("---> Moving right table to the Y centre. ");
763
764         digitalWrite(C3_cilindar, HIGH);
765         digitalWrite(C4_cilindar, LOW);
766         delay(cooldown);
767
768         Serial.print("Waiting for input from sensors on C3 and C4. ");
769         while (isMoving == true) {
770             if ((digitalRead(C3_izvucen) == 1) && (digitalRead(C4_uvucen) == 1)) {

```

```

771     isMoving = false;
772     Serial.println("Move completed. ");
773 }
774 }
775
776 }
777 else {
778     Serial.println("--> Check input number for table positon.");
779 }
780
781 }
782
783 void tableGoUp(char tableSide) {
784     if (tableSide == 2) {
785         if ((digitalRead(C3_izvucen) == 0) || (digitalRead(C4_izvucen) == 0)) {
786             isMoving = true;
787             Serial.print("---> Moving right table up. ");
788
789             digitalWrite(C3_cilindar, HIGH);
790             digitalWrite(C4_cilindar, HIGH);
791             delay(cooldown);
792
793             Serial.print("Waiting for input from sensors on C3 and C4. ");
794             while (isMoving == true) {
795
796                 if ((digitalRead(C3_izvucen) == 1) && (digitalRead(C4_izvucen) == 1)) {
797                     isMoving = false;
798                     Serial.println("Move completed.");
799                 }
800             }
801         }
802         else {
803             Serial.println("--> Can't go up anymore.");
804             digitalWrite(LED_Error, 1);
805             delay(500);
806             digitalWrite(LED_Error, 0);
807         }
808     }
809     else if (tableSide == 1) {
810         if ((digitalRead(C1_izvucen) == 0) || (digitalRead(C2_izvucen) == 0)) {
811             isMoving = true;
812             Serial.print("---> Moving left table up. ");
813
814             digitalWrite(C1_cilindar, HIGH);
815             digitalWrite(C2_cilindar, HIGH);
816             delay(cooldown);
817
818             Serial.print("Waiting for input from sensors on C1 and C2. ");
819             while (isMoving == true) {
820
821                 if ((digitalRead(C1_izvucen) == 1) && (digitalRead(C2_izvucen) == 1)) {
822                     isMoving = false;
823                     Serial.println("Move completed.");
824                 }
825             }
826         }
827         else {
828             Serial.println("--> Can't go up anymore.");
829             digitalWrite(LED_Error, 1);
830             delay(500);

```

```

831     digitalWrite(LED_Error, 0);
832 }
833 }
834 else
835 {
836     Serial.println("--> Incorrect input. Aborting sequence...");
837 }
838
839 }
840
841 void tableGoDown(char tableSide) {
842     if (tableSide == 2) {
843         if ((digitalRead(C3_uvucen) == 0) || (digitalRead(C4_uvucen) == 0)) {
844             isMoving = true;
845             Serial.print("---> Moving right table down. ");
846
847             digitalWrite(C3_cilindar, LOW);
848             digitalWrite(C4_cilindar, LOW);
849             delay(cooldown);
850
851             Serial.print("Waiting for input from sensors on C3 and C4. ");
852             while (isMoving == true) {
853
854                 if ((digitalRead(C3_uvucen) == 1) && (digitalRead(C4_uvucen) == 1)) {
855                     isMoving = false;
856                     Serial.println("Move completed.");
857                 }
858             }
859         }
860         else {
861             Serial.println("--> Can't go further down.");
862             digitalWrite(LED_Error, 1);
863             delay(500);
864             digitalWrite(LED_Error, 0);
865         }
866     }
867     else if (tableSide == 1) {
868         if ((digitalRead(C1_uvucen) == 0) || (digitalRead(C2_uvucen) == 0)) {
869             isMoving = true;
870             Serial.print("---> Moving left table down. ");
871
872             digitalWrite(C1_cilindar, LOW);
873             digitalWrite(C2_cilindar, LOW);
874             delay(cooldown);
875
876             Serial.print("Waiting for input from sensors on C1 and C2. ");
877             while (isMoving == true) {
878
879                 if ((digitalRead(C1_uvucen) == 1) && (digitalRead(C2_uvucen) == 1)) {
880                     isMoving = false;
881                     Serial.println("Move completed.");
882                 }
883             }
884         }
885         else {
886             Serial.println("--> Can't go further down.");
887             digitalWrite(LED_Error, 1);
888             delay(500);
889             digitalWrite(LED_Error, 0);
890         }
891     }
892 }

```

```

891 }
892 else
893 {
894     Serial.println("---> Incorrect input. Aborting sequence...");
895 }
896 }
897
898 void rotateRight() {
899     if (digitalRead(handIsRight) == 1) {
900         Serial.println("---> Table is already right. ");
901         digitalWrite(LED_Error, 1);
902         delay(500);
903         digitalWrite(LED_Error, 0);
904     }
905     else {
906         if (isUp == true)
907         {
908             isMoving = true;
909             Serial.print("---> Rotating to the right.");
910             digitalWrite(C7_cilindar, HIGH);
911             delay(cooldown);
912
913             while (isMoving == true) {
914
915                 if (digitalRead(handIsRight) == 1) {
916                     isMoving = false;
917                     Serial.println("Move completed.");
918                 }
919             }
920         }
921     }
922     else if (isUp == false)
923     {
924         isMoving = true;
925         Serial.print("---> Rotating to the right.");
926         digitalWrite(C8_cilindar, LOW);
927         delay(cooldown);
928
929         isUp = true;
930
931         digitalWrite(C7_cilindar, HIGH);
932         delay(cooldown);
933
934         while (isMoving == true) {
935
936             if (digitalRead(handIsRight) == 1) {
937                 isMoving = false;
938                 Serial.println("Move completed.");
939             }
940         }
941     }
942     else
943     {
944         Serial.println("---> Rotation to the right can't be perfomed. ");
945     }
946 }
947 }
948
949 void rotateLeft() {
950     if (digitalRead(handIsLeft) == 1) {

```

```

951     Serial.println("---> Table is already left. ");
952     digitalWrite(LED_Error, 1);
953     delay(500);
954     digitalWrite(LED_Error, 0);
955 }
956 else {
957     if (isUp == true)
958     {
959         isMoving = true;
960         Serial.print("---> The hand is up. Rotating to the left.");
961
962         digitalWrite(C7_cilindar, LOW);
963         delay(cooldown);
964
965         while (isMoving == true) {
966             if (digitalRead(handIsLeft) == 1) {
967                 isMoving = false;
968                 Serial.println("Move completed.");
969             }
970         }
971     }
972 }
973 else if (isUp == false)
974 {
975     isMoving = true;
976     Serial.print("---> The hand is down. Rotating to the left.");
977
978     digitalWrite(C8_cilindar, LOW);
979     delay(cooldown);
980
981     isUp = true;
982
983     digitalWrite(C7_cilindar, LOW);
984     delay(cooldown);
985
986     while (isMoving == true) {
987         if (digitalRead(handIsLeft) == 1) {
988             isMoving = false;
989             Serial.println("Move completed.");
990         }
991     }
992 }
993 else
994 {
995     Serial.println("---> Rotation to the left can't be perfomed. ");
996 }
997 }
998 }
999
1000 void StartPressed() {
1001     Serial.println("--> Start pressed.");
1002     startPressed = HIGH;
1003
1004     digitalWrite(LED_Start, LOW);
1005     digitalWrite(LED_Stop, LOW);
1006
1007     Mb.R[2] = 4;
1008
1009 }
1010

```

```

1011 void StopPressed() {
1012     Serial.println("--> Stop pressed.");
1013     Mb.R[2] = 3;
1014     digitalWrite(LED_Stop, HIGH);
1015 }
1016
1017 bool isEmptyTable(byte tableSide) {
1018     bool notEmpty = false;
1019     if (tableSide == 1) {
1020
1021         for (char i = 1; i < 10; i++) {
1022             if ((messageArray[i] == 1) && (notEmpty == false)) {
1023                 notEmpty = true;
1024             }
1025         }
1026         if (notEmpty == true) {
1027             return true;
1028         }
1029         else {
1030             return false;
1031         }
1032     }
1033 }
1034 else if (tableSide == 2) {
1035     for (char i = 1; i < 10; i++) {
1036
1037         if ((messageArray[i] == 1) && (notEmpty == false)) {
1038             notEmpty = true;
1039         }
1040
1041     }
1042     if (notEmpty == true) {
1043         return true;
1044     }
1045     else {
1046         return false;
1047     }
1048 }
1049 else {
1050     return false;
1051 }
1052 }
1053
1054 bool isEmptySpot(byte tableSide, byte pawnPosition) {
1055     //treba testirat ovo
1056     if (tableSide == 1) {
1057         if (tableLeft[pawnPosition] == 1) {
1058             return false;
1059         }
1060         else {
1061             return true;
1062         }
1063     }
1064     else if (tableSide == 2) {
1065         if (tableRight[pawnPosition] == 1) {
1066             return false;
1067         }
1068         else {
1069             return true;
1070         }
1071     }
1072 }

```

```

1071 }
1072 else {
1073     Serial.println("Wrong table side choosen for checking if spot is empty.");
1074 }
1075 }
1076
1077 void goTo(byte tableSide, byte pawnPosition) {
1078
1079     if (tableSide == 1) {
1080
1081         if ((pawnPosition == 1) || (pawnPosition == 4) || (pawnPosition == 7)) { // move
table in x-axis
1082             if (digitalRead(C5_uvucen) == 0 || digitalRead(C6_uvucen) == 0) {
1083
1084                 isMoving = true;
1085                 Serial.print("Going left. ");
1086
1087                 digitalWrite(CONTROLLINO_R6, LOW);
1088                 digitalWrite(CONTROLLINO_R5, LOW);
1089                 delay(cooldown);
1090
1091                 Serial.print("Waiting for input from sensors C5 and C6. ");
1092                 while (isMoving == true) {
1093
1094                     if (digitalRead(C5_uvucen) == 1 && digitalRead(C6_uvucen) == 1) {
1095                         isMoving = false;
1096                         Serial.println("Move completed.");
1097                     }
1098                 }
1099             }
1100         }
1101     }
1102     else if ((pawnPosition == 2) || (pawnPosition == 5) || (pawnPosition == 8)) {
1103         if (digitalRead(C5_izvucen) == 0 || digitalRead(C6_uvucen) == 0) {
1104
1105             isMoving = true;
1106             Serial.print("Going right. ");
1107
1108             digitalWrite(CONTROLLINO_R6, LOW);
1109             digitalWrite(CONTROLLINO_R5, HIGH);
1110             delay(cooldown);
1111
1112             while (isMoving == true) {
1113                 Serial.print("Waiting for input from sensors C5 and C6... ");
1114                 if (digitalRead(C5_izvucen) == 1 && digitalRead(C6_uvucen) == 1) {
1115                     isMoving = false;
1116                     Serial.println("Move completed.");
1117                 }
1118             }
1119         }
1120     };
1121 }
1122 else if ((pawnPosition == 3) || (pawnPosition == 6) || (pawnPosition == 9)) {
1123     if (digitalRead(C5_izvucen) == 0 || digitalRead(C6_izvucen) == 0) {
1124
1125         isMoving = true;
1126         Serial.print("Going right. ");
1127
1128         digitalWrite(CONTROLLINO_R6, HIGH);
1129         digitalWrite(CONTROLLINO_R5, HIGH);

```

```

1130     delay(cooldown);
1131
1132     while (isMoving == true) {
1133         Serial.print("Waiting for input from sensors C5 and C6... ");
1134         if (digitalRead(C5_izvucen) == 1 && digitalRead(C6_izvucen) == 1) {
1135             isMoving = false;
1136             Serial.println("Move completed.");
1137         }
1138     }
1139
1140 };
1141 }
1142
1143 if ((pawnPosition == 1) || (pawnPosition == 2) || (pawnPosition == 3)) {
1144     if (digitalRead(C1_uvucen) == 0 || digitalRead(C2_uvucen) == 0) {
1145         isMoving = true;
1146         Serial.print("Moving left table up. ");
1147
1148         digitalWrite(C1_cilindar, LOW);
1149         digitalWrite(C2_cilindar, LOW);
1150         delay(cooldown);
1151
1152         Serial.print("Waiting for input from sensors on C1 and C2. ");
1153         while (isMoving == true) {
1154
1155             if (digitalRead(C1_uvucen) == 1 && digitalRead(C2_uvucen) == 1) {
1156                 isMoving = false;
1157                 Serial.println("Move completed.");
1158             }
1159         }
1160     }
1161 }
1162 else if ((pawnPosition == 4) || (pawnPosition == 5) || (pawnPosition == 6)) {
1163     if (digitalRead(C1_izvucen) != 1 || digitalRead(C2_uvucen) != 1) {
1164         isMoving = true;
1165         Serial.print("Moving left table to centre. ");
1166
1167         digitalWrite(C1_cilindar, HIGH);
1168         digitalWrite(C2_cilindar, LOW);
1169         delay(cooldown);
1170
1171         Serial.print("Waiting for input from sensors on C1 and C2. ");
1172         while (isMoving == true) {
1173
1174             if (digitalRead(C1_izvucen) == 1 && digitalRead(C2_uvucen) == 1) {
1175                 isMoving = false;
1176                 Serial.println("Move completed.");
1177             }
1178         }
1179     }
1180 }
1181 else if ((pawnPosition == 7) || (pawnPosition == 8) || (pawnPosition == 9)) {
1182     if (digitalRead(C1_izvucen) == 0 || digitalRead(C2_izvucen) == 0) {
1183         isMoving = true;
1184         Serial.print("Moving left table down. ");
1185
1186         digitalWrite(C1_cilindar, HIGH);
1187         digitalWrite(C2_cilindar, HIGH);
1188         delay(cooldown);
1189

```

```

1190     Serial.print("Waiting for input from sensors on C1 and C2. ");
1191     while (isMoving == true) {
1192         if (digitalRead(C1_izvucen) == 1 && digitalRead(C2_izvucen) == 1) {
1193             isMoving = false;
1194             Serial.println("Move completed.");
1195         }
1196     }
1197 }
1198 }
1199 }
1200 else {
1201     Serial.print("Incorrect format of requested position. Req. position was: ");
1202     for (int i = 0; i < 10; i++)
1203     {
1204         Serial.print(messageArray[i]);
1205     }
1206     Serial.println(" ");
1207 }
1208 }
1209 }
1210 else if (tableSide == 2) {
1211     if ((pawnPosition == 1) || (pawnPosition == 4) || (pawnPosition == 7)) {
1212         isMoving = true;
1213         Serial.print("Going left. ");
1214
1215         digitalWrite(CONTROLLINO_R6, LOW);
1216         digitalWrite(CONTROLLINO_R5, LOW);
1217         delay(cooldown);
1218
1219         Serial.print("Waiting for input from sensors C5 and C6. ");
1220         while (isMoving == true) {
1221             if (digitalRead(C5_uvucen) == 1 && digitalRead(C6_uvucen) == 1) {
1222                 isMoving = false;
1223                 Serial.println("Move completed.");
1224             }
1225         }
1226     }
1227 }
1228 }
1229 }
1230 else if ((pawnPosition == 2) || (pawnPosition == 5) || (pawnPosition == 8)) {
1231     isMoving = true;
1232     Serial.print("Going right. ");
1233
1234     digitalWrite(CONTROLLINO_R6, LOW);
1235     digitalWrite(CONTROLLINO_R5, HIGH);
1236     delay(cooldown);
1237
1238     while (isMoving == true) {
1239         Serial.print("Waiting for input from sensors C5 and C6... ");
1240         if (digitalRead(C5_izvucen) == 1 && digitalRead(C6_uvucen) == 1) {
1241             isMoving = false;
1242             Serial.println("Move completed.");
1243         }
1244     }
1245 }
1246 }
1247 else if ((pawnPosition == 3) || (pawnPosition == 6) || (pawnPosition == 9)) {
1248     isMoving = true;
1249

```

```

1250 Serial.print("Going right. ");
1251
1252 digitalWrite(CONTROLLINO_R6, HIGH);
1253 digitalWrite(CONTROLLINO_R5, HIGH);
1254 delay(cooldown);
1255
1256 while (isMoving == true) {
1257     Serial.print("Waiting for input from sensors C5 and C6... ");
1258     if (digitalRead(C5_izvucen) == 1 && digitalRead(C6_izvucen) == 1) {
1259         isMoving = false;
1260         Serial.println("Move completed.");
1261     }
1262 }
1263 }
1264
1265 if ((pawnPosition == 1) || (pawnPosition == 2) || (pawnPosition == 3)) {
1266
1267     isMoving = true;
1268     Serial.print("Moving right table up. ");
1269
1270     digitalWrite(C3_cilindar, LOW);
1271     digitalWrite(C4_cilindar, LOW);
1272     delay(cooldown);
1273
1274     Serial.print("Waiting for input from sensors on C3 and C4. ");
1275     while (isMoving == true) {
1276
1277         if (digitalRead(C3_uvucen) == 1 && digitalRead(C4_uvucen) == 1) {
1278             isMoving = false;
1279             Serial.println("Move completed.");
1280         }
1281     }
1282 }
1283
1284 else if ((pawnPosition == 4) || (pawnPosition == 5) || (pawnPosition == 6)) {
1285
1286     isMoving = true;
1287     Serial.print("Moving right table to centre. ");
1288
1289     digitalWrite(C3_cilindar, HIGH);
1290     digitalWrite(C4_cilindar, LOW);
1291     delay(cooldown);
1292
1293     Serial.print("Waiting for input from sensors on C3 and C4. ");
1294     while (isMoving == true) {
1295
1296         if (digitalRead(C3_izvucen) == 1 && digitalRead(C4_uvucen) == 1) {
1297             isMoving = false;
1298             Serial.println("Move completed.");
1299         }
1300     }
1301 }
1302
1303 else if ((pawnPosition == 7) || (pawnPosition == 8) || (pawnPosition == 9)) {
1304
1305     isMoving = true;
1306     Serial.print("Moving right table down. ");
1307
1308     digitalWrite(C3_cilindar, HIGH);
1309     digitalWrite(C4_cilindar, HIGH);

```

```

1310     delay(cooldown);
1311
1312     Serial.print("Waiting for input from sensors on C3 and C4. ");
1313     while (isMoving == true) {
1314
1315         if (digitalRead(C3_izvucen) == 1 && digitalRead(C4_izvucen) == 1) {
1316             isMoving = false;
1317             Serial.println("Move completed.");
1318         }
1319     }
1320
1321 }
1322 else {
1323     Serial.print("Incorrect format of requested position. Req. position was: ");
1324     for (int i = 0; i < 10; i++)
1325     {
1326         Serial.print(messageArray[i]);
1327     }
1328     Serial.println(" ");
1329 }
1330 }
1331 else {
1332     Serial.println("Wrong table side number choosen.");
1333 }
1334 }
1335
1336 void pawnPickUp() {
1337     Serial.print("--> Initiating vacuum activation. ");
1338     while (digitalRead(pawnGrabbed_V1) != 1) {
1339         digitalWrite(C8_cilindar, HIGH);
1340         delay(cooldown);
1341         digitalWrite(Vacuum_1, HIGH);
1342         delay(cooldown);
1343         digitalWrite(C8_cilindar, LOW);
1344         delay(cooldown);
1345     }
1346     digitalWrite(C9_cilindar, HIGH);
1347     delay(cooldown);
1348     Serial.print("Pawn is picked up. ");
1349     isHandFull = true;
1350 }
1351
1352 void pawnDrop() {
1353
1354     digitalWrite(C9_cilindar, LOW);
1355     delay(cooldown);
1356     delay(cooldown);
1357     digitalWrite(C8_cilindar, HIGH);
1358     delay(cooldown);
1359
1360     Serial.print("--> Initiating vacuum de-activation. ");
1361     digitalWrite(Vacuum_1, LOW);
1362     digitalWrite(C8_cilindar, LOW);
1363     delay(cooldown);
1364
1365     isHandFull = false;
1366     Serial.println("Pawn dropped. ");
1367 }
1368
1369 byte findAvailablePawn(byte tableSide) {

```

```

1370 byte pawn;
1371 if (tableSide == 1) {
1372     for (byte i = 9; i > 0; i--) {
1373         if (tableLeft[i] == 1) {
1374             pawn = i;
1375             break;
1376         }
1377         pawn = 0;
1378     }
1379 }
1380 else if (tableSide == 2) {
1381     for (byte i = 9; i > 0; i--) {
1382         if (tableRight[i] == 1) {
1383             pawn = i;
1384             break;
1385         }
1386         pawn = 0;
1387     }
1388 }
1389 return pawn;
1390 }
1391
1392 byte findFreeSpot(byte tableSide) {
1393     byte pawn;
1394     if (tableSide == 1) {
1395         for (byte i = 1; i < 10; i++) {
1396             if (tableLeft[i] == 0) {
1397                 pawn = i;
1398                 break;
1399             }
1400             pawn = 0;
1401         }
1402     }
1403     else if (tableSide == 2) {
1404         for (byte i = 1; i < 10; i++) {
1405             if (tableRight[i] == 0) {
1406                 pawn = i;
1407                 break;
1408             }
1409             pawn = 0;
1410         }
1411     }
1412     return pawn;
1413 }
1414
1415 byte currentPawnPosition() {
1416     // function for finding the position under the hand
1417     if (digitalRead(handIsLeft) == 1) {
1418
1419         if ((digitalRead(C1_uvucen) == 1) && (digitalRead(C2_uvucen) == 1)) {
1420
1421             if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1422                 return 1;
1423             }
1424             else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1425 ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1426                 return 2;
1427             }
1428             else {
1429                 return 3;

```

```

1429     }
1430
1431     }
1432     else if (((digitalRead(C1_izvucen) == 1) && (digitalRead(C2_uvucen) == 1)) ||
1433              ((digitalRead(C2_izvucen) == 1) && (digitalRead(C1_uvucen) == 1))) {
1434         if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1435             return 4;
1436         }
1437         else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1438                  ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1439             return 5;
1440         }
1441         else {
1442             return 6;
1443         }
1444     }
1445     else {
1446         if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1447             return 7;
1448         }
1449         else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1450                  ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1451             return 8;
1452         }
1453         else {
1454             return 9;
1455         }
1456     }
1457     else if (digitalRead(handIsRight) == 1) {
1458
1459         if ((digitalRead(C3_uvucen) == 1) && (digitalRead(C4_uvucen) == 1)) {
1460
1461             if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1462                 return 1;
1463             }
1464             else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1465                      ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1466                 return 2;
1467             }
1468             else {
1469                 return 3;
1470             }
1471         }
1472         else if (((digitalRead(C3_izvucen) == 1) && (digitalRead(C4_uvucen) == 1)) ||
1473                  ((digitalRead(C4_izvucen) == 1) && (digitalRead(C3_uvucen) == 1))) {
1474             if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1475                 return 4;
1476             }
1477             else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1478                      ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1479                 return 5;
1480             }
1481             else {
1482                 return 6;
1483             }
1484         }
1485     }

```

```

1483     else {
1484
1485         if ((digitalRead(C5_uvucen) == 1) && (digitalRead(C6_uvucen) == 1)) {
1486             return 7;
1487         }
1488         else if (((digitalRead(C5_izvucen) == 1) && (digitalRead(C6_uvucen) == 1)) ||
1489 ((digitalRead(C6_izvucen) == 1) && (digitalRead(C5_uvucen) == 1))) {
1489             return 8;
1490         }
1491         else {
1492             return 9;
1493         }
1494     }
1495 }
1496 }
1497
1498 byte currentTableSide() {
1499
1500     if (digitalRead(handIsLeft) == 1) {
1501         return 1;
1502     }
1503     else if (digitalRead(handIsRight) == 1) {
1504         return 2;
1505     }
1506 }
1507
1508 bool onlyBinaryNumbers(String messageString) {
1509     for (char i = 0; i < messageString.length(); i++) {
1510         if ((String(messageString.charAt(i)) != "1" && (String(messageString.charAt(i))
1511 != "0"))) {
1512             Serial.println("--> The received message contains non-binary numbers.");
1513             return false;
1514         }
1515     }
1516     return true;
1517 }
1518
1519 void flashError() {
1520     digitalWrite(LED_Error, 1);
1521     delay(500);
1522     digitalWrite(LED_Error, 0);
1523 }

```